Computer Programming

IIE (799-18)

(1) May 6, 2004

Lecturer

Ying CHEN （陳　迎）

Department of Quantum Engineering and System Science

Teaching Assistant

Miti Ruchanurucks

Graduate School of Information Science and Technology

Today

- General course information
- Distribute ECC Account
- Training on ECC System
- Getting started
  Login, Logout, simple Unix, ...

Objectives & Schedule

1-2: Introduction

Guidance to ECC, Basic Unix

3-6: Java programming

Syntax, Object-Oriented programming

7-10: Application

GUI (Graphics and Applets)

Numerical algorithms

- Basic idea of numerical analysis
- Basic techniques: Root of equation, Integration, Random number

Course Calendar

June 3 (Thu.) : No lesson

Course Assessment

- In-class/take-home exercises: 50%
- Take-home final assignment: 50%
- Email to us

  ying@q.t.u-tokyo.ac.jp

  miti@cvl.iis.u-tokyo.ac.jp

References

- Davies, R. *Introductory Java for Scientists and Engineers* (Harlow: Addison-Wesley, 1998)
- W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes -The Art of Scientific Computing (Cambridge University Press, 1992)
- Sun Java tutorial: http://java.sun.com/docs/books/tutorial/
- Unix tutorial for beginners: http://www.ee.surrey.ac.uk/Teaching/Unix/

Course Information

http://gwp01.t.u-tokyo.ac.jp/kouryu/students/classes/ica.htm

- Announcement
- Lecture Notes
  (will be uploaded in 1-2 days after lecture)
- Take-home Exercises
- Final assignment

Requirement

- Be on time
- Don’t talk each other loudly in lesson
- Home work/Assignment must be turned in by the due date
- **Study after class**
  - Unix + Java: Terminal Rooms on campus
  - PC: **download: Java™ 2 Platform for PC**
    http://java.sun.com/j2se/1.4/index.html
- Please ask questions freely in lecture

Important Rules

- **Your account is only for yourself,**
  only for study and research (Don't abuse!)
  If you miss the password, need to
  get a new one (ECC office).

- **No food and beverage in terminal room.**

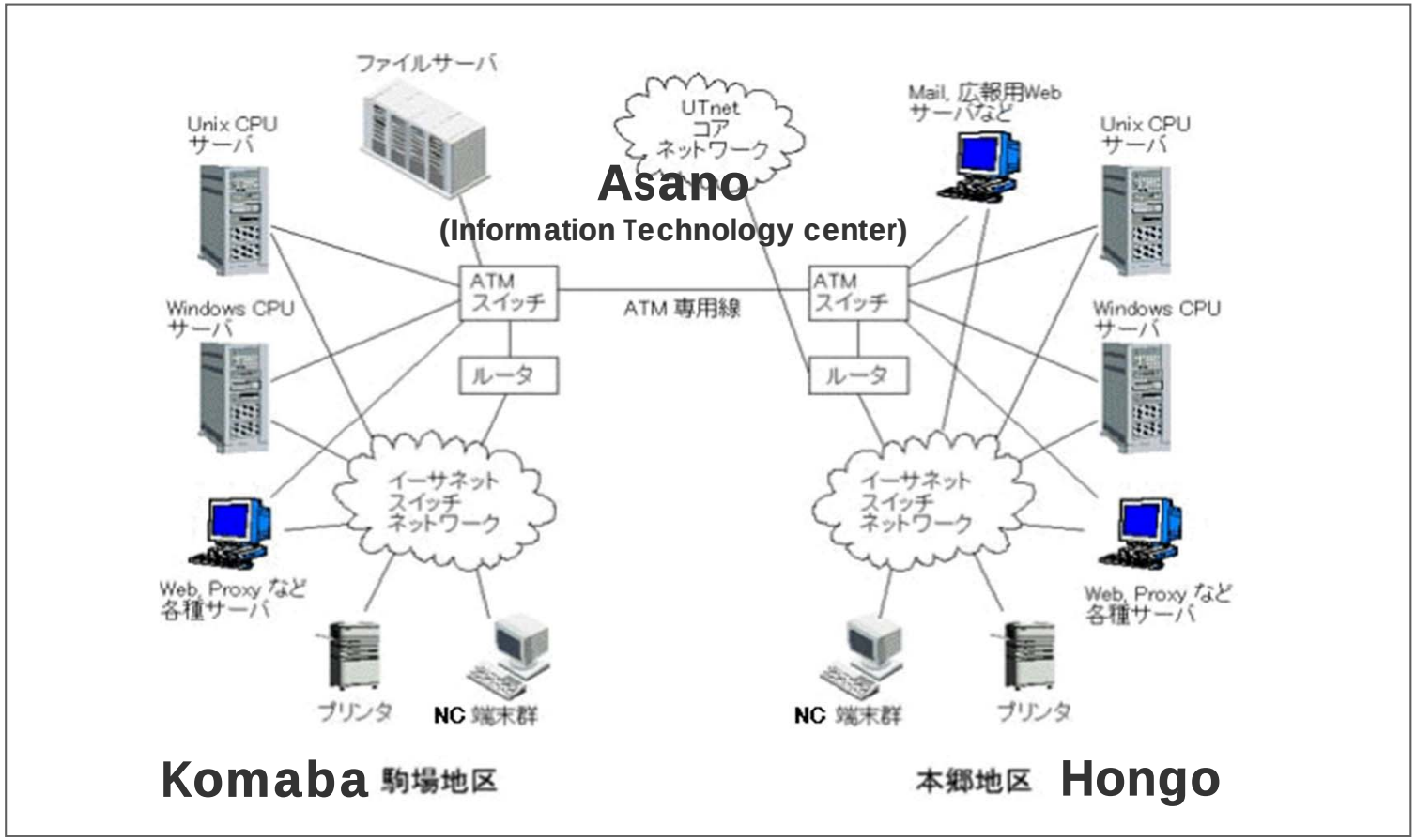
- **Make sure your terminal "Shut down"**
  before leaving.

- **Take care Limitations (Dir: 500MB, Mail: 100MB)**
  https://secure.ecc.u-tokyo.ac.jp/quota.html

Guidance to ECC System

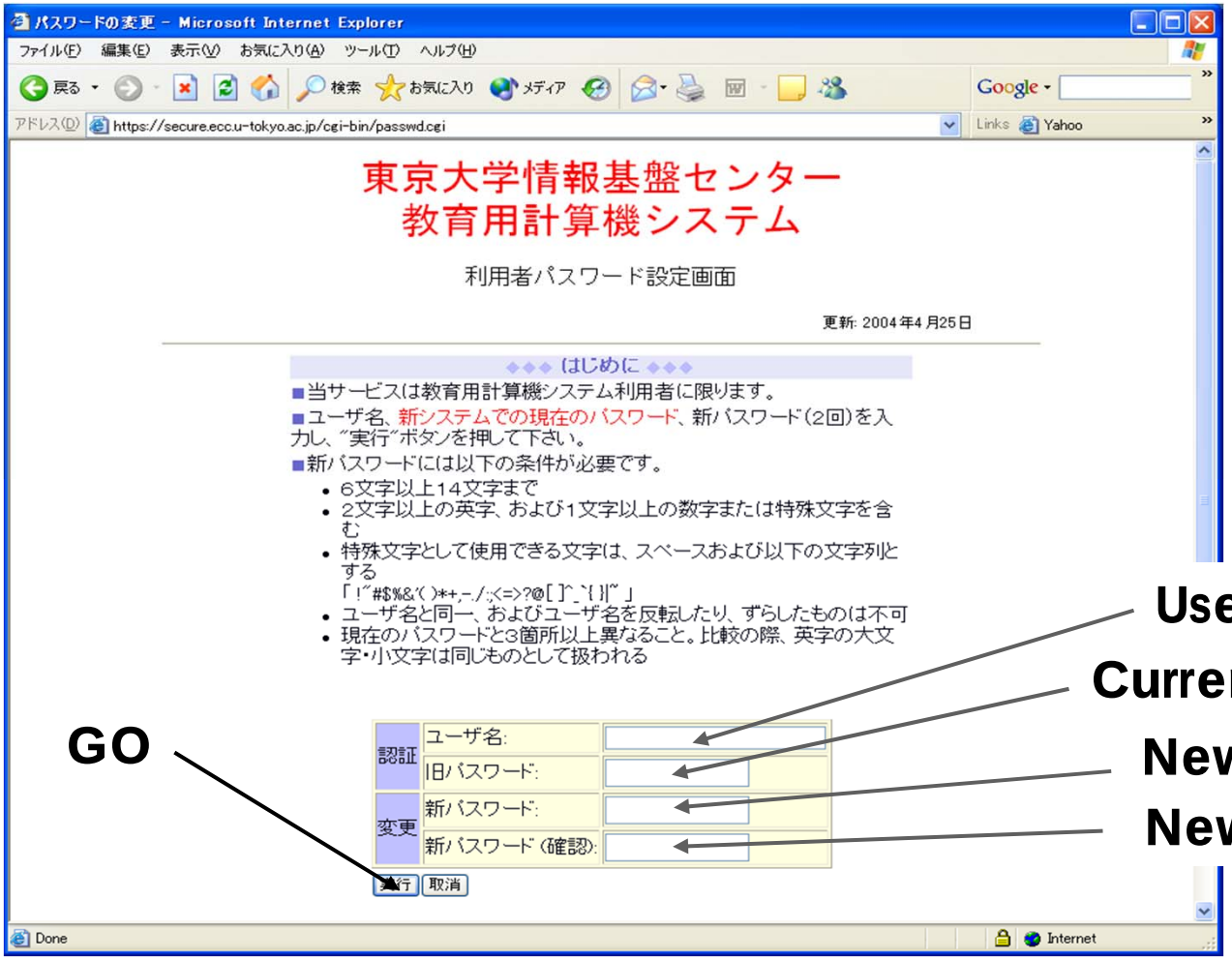
Education Computer Center (ECC) System

http://www.ecc.u-tokyo.ac.jp/

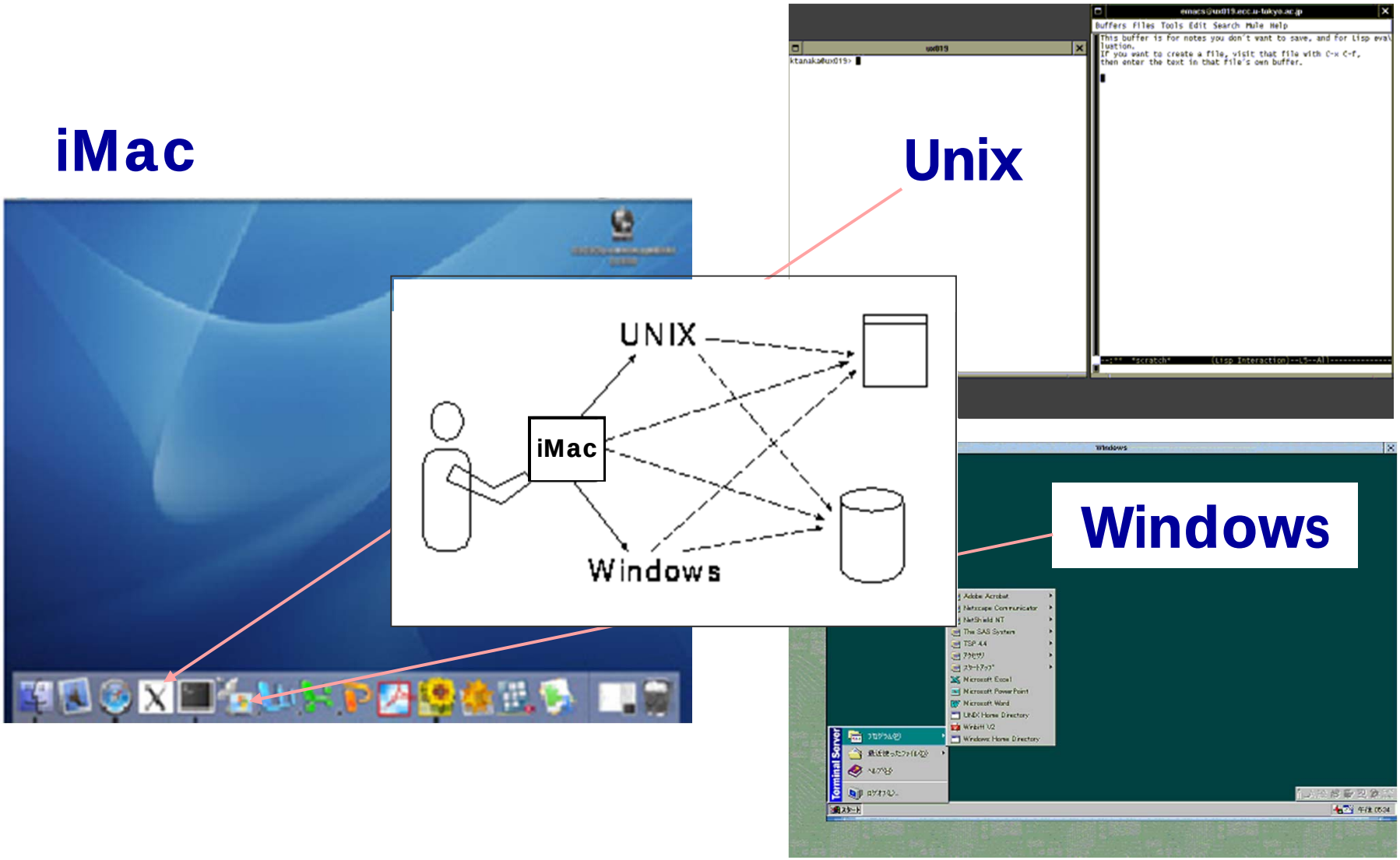


Changing Password

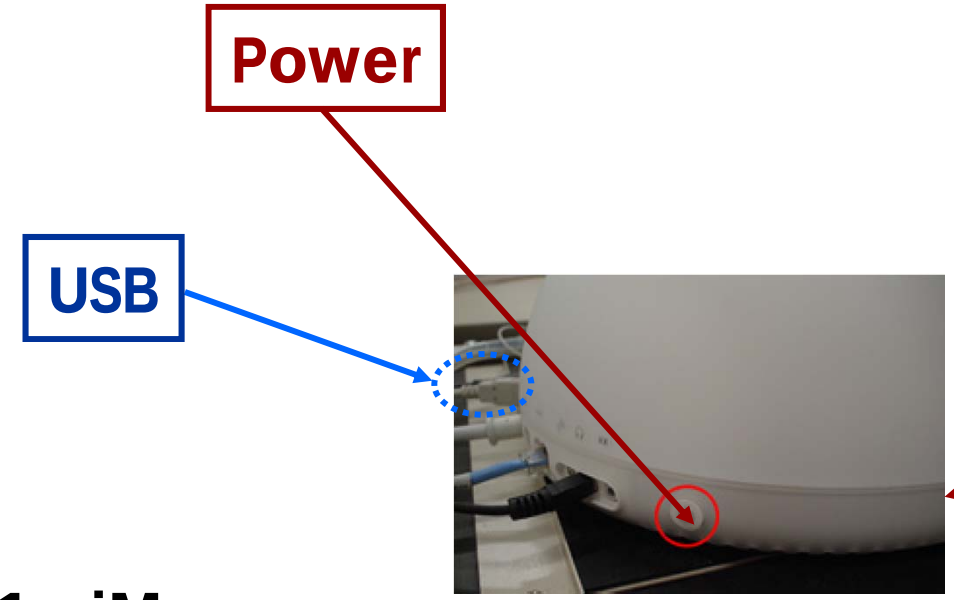
https://secure.ecc.u-tokyo.ac.jp/cgi-bin/passwd.cgi

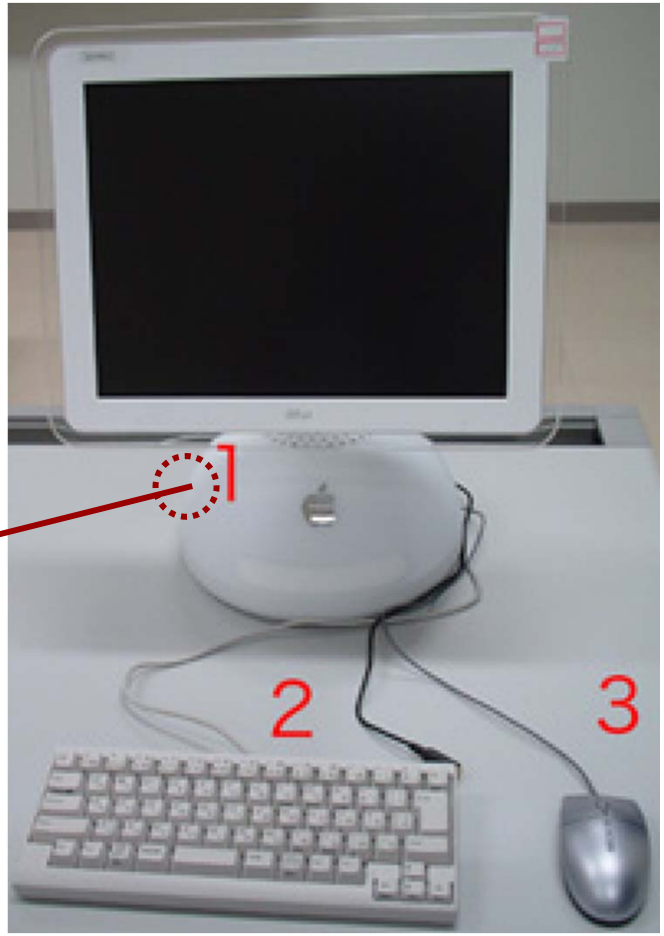


“Three kinds of world”



iMac Terminal



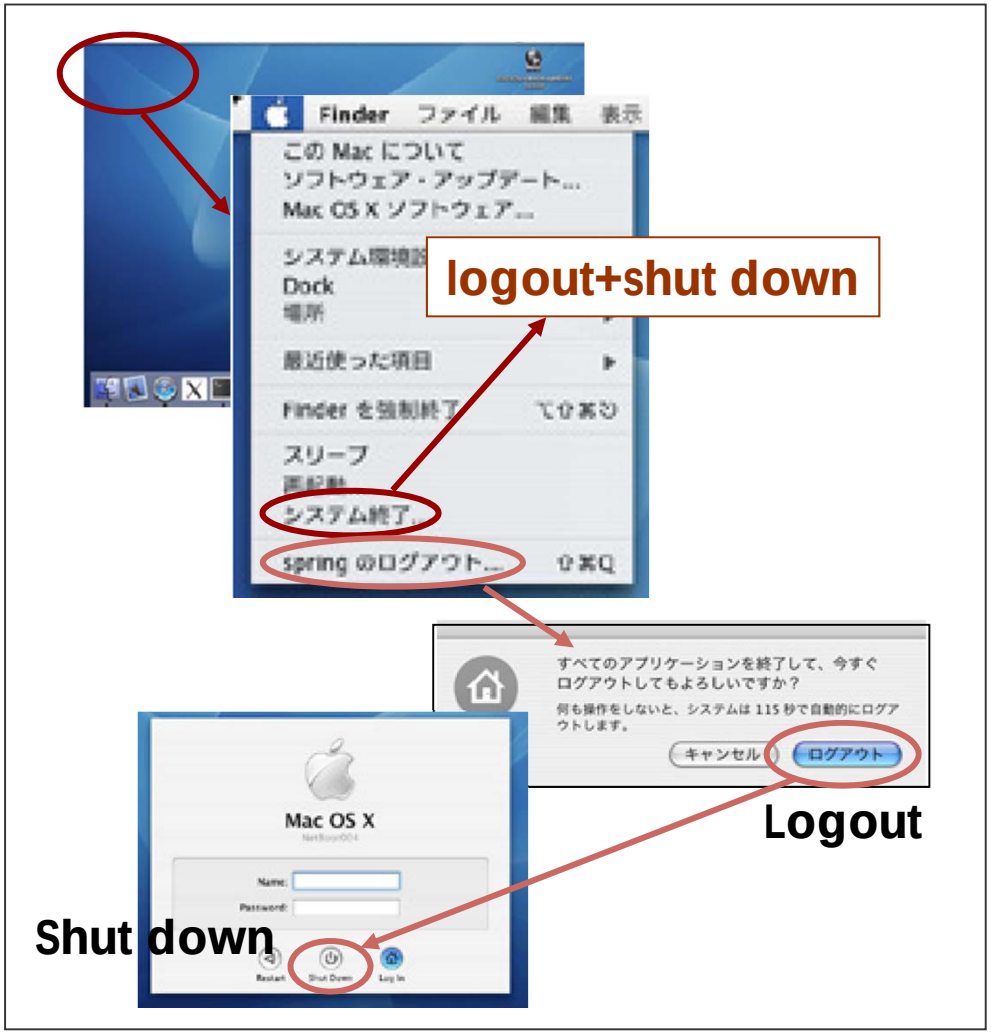


1. iMac
2. Happy Hacking keyboard
3. Mouse

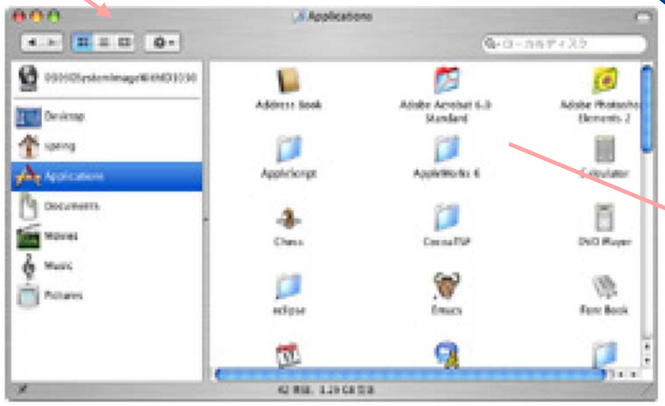
iMac: Login/Logout

Login

Logout



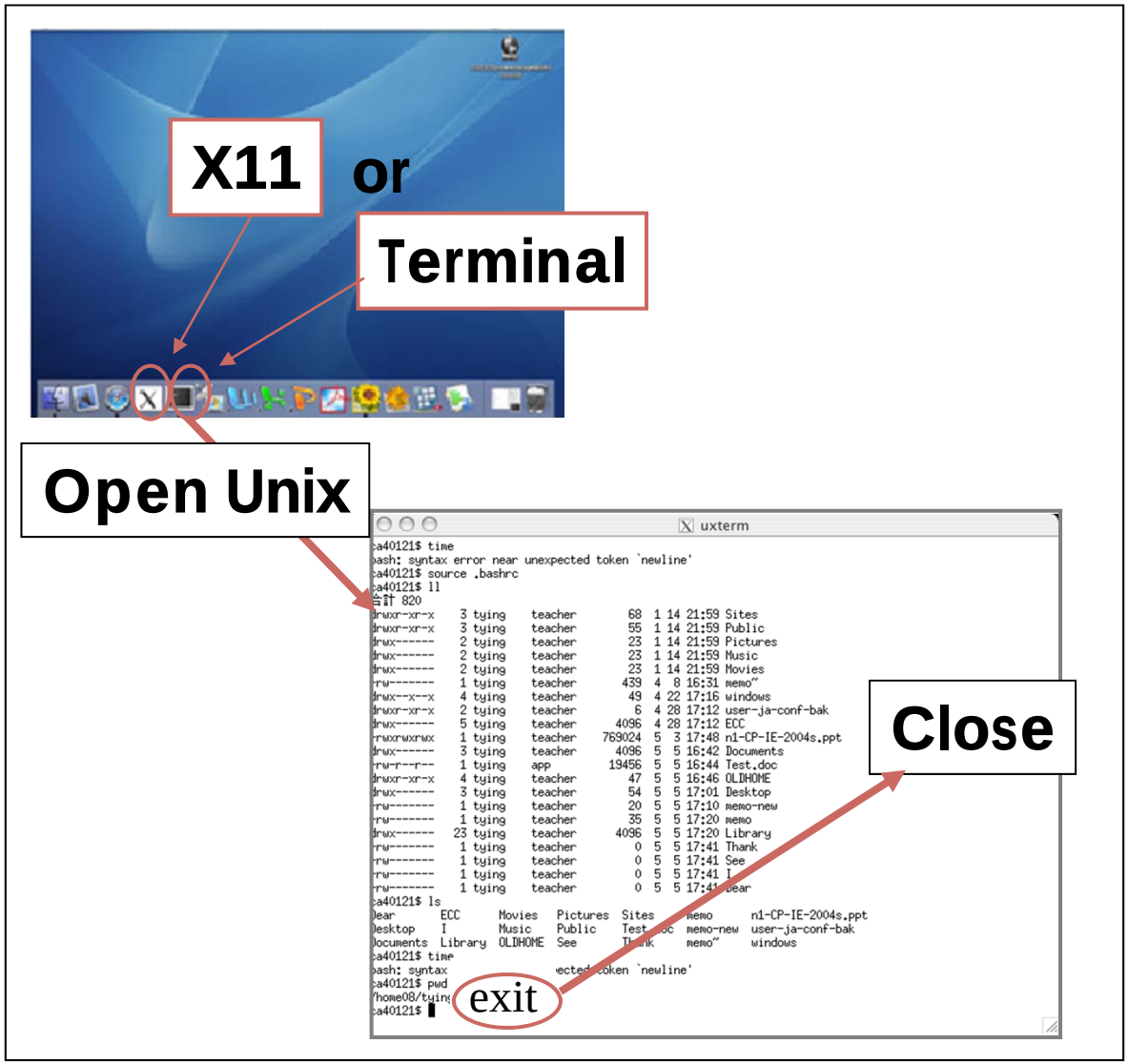
iMac: Applications



- Finder
- Mail
- Safai: Web browser
- X11, Terminal: Unix
- Remote Desktop Connection: Windows environment
- Microspft: Word, excel, Power Point
- Acrobat 6.0
- Photoshop Elements 2.0
- Mathematica 5.0
- STATA

...more...

Unix Environment



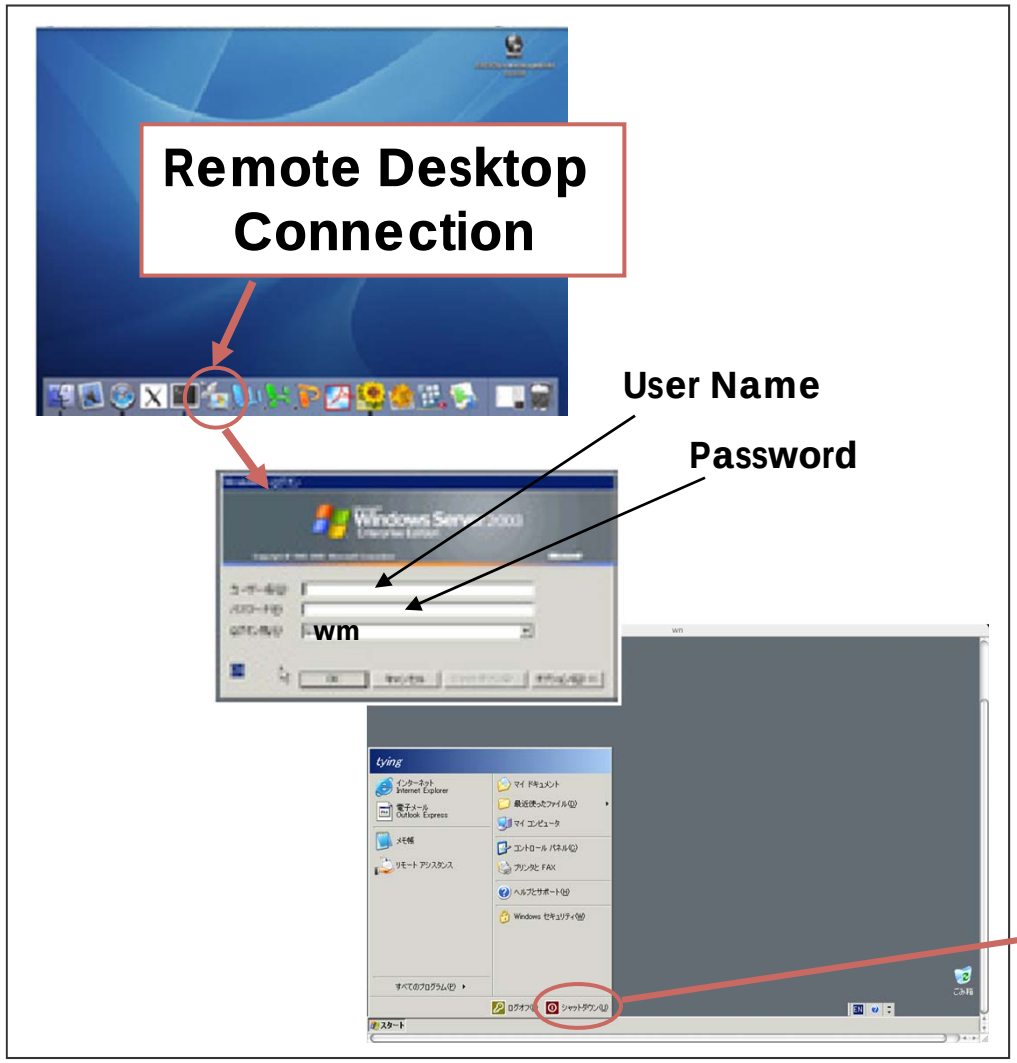
Application

- C, C++
- Fortran, Fortran90
- Pascal
- Java

...

Windows Environment

login



Application

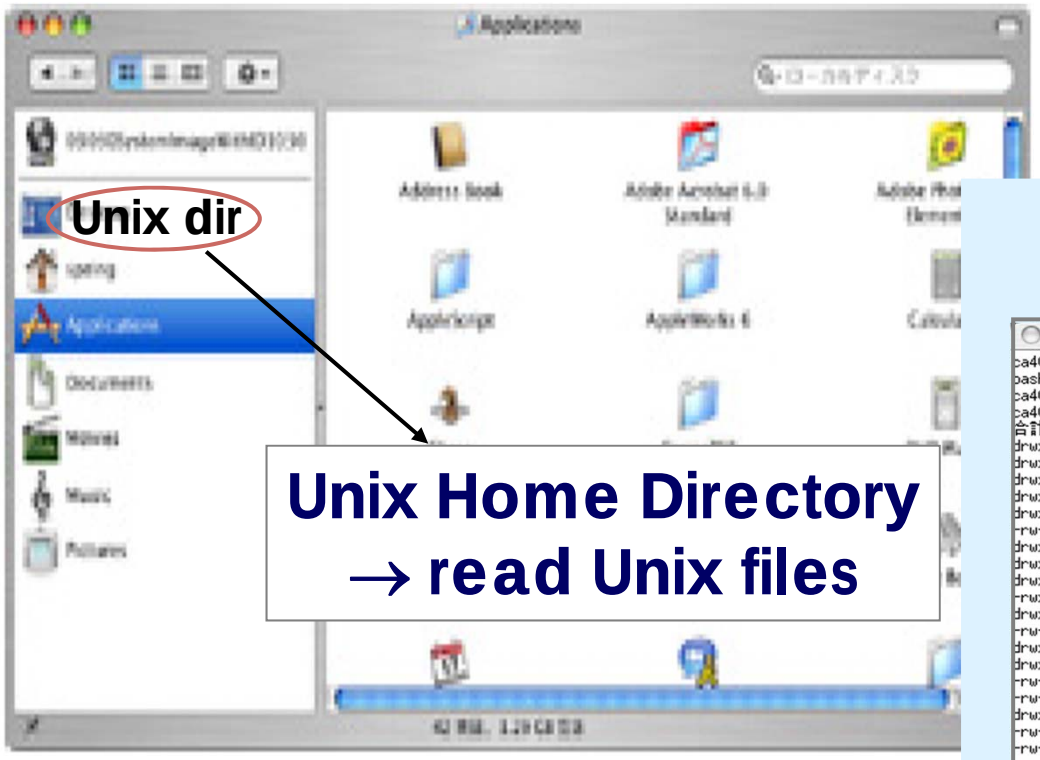
- Givewin2
- Microsoft Office
- Mozilla
- Opera
- SAS (statistics)
- TSP (statistics)
- Adobe Acrobat 6.0 Standard
- Adobe Photoshop Element
- video LAN

...

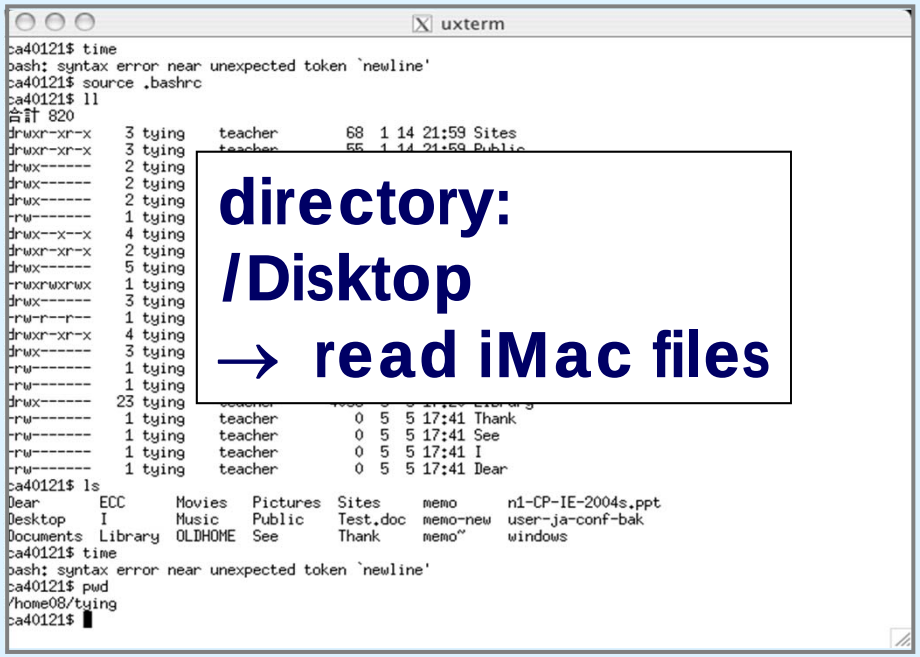
logout

File Sharing Between Unix and iMac

iMac/Finder



Unix/Terminal



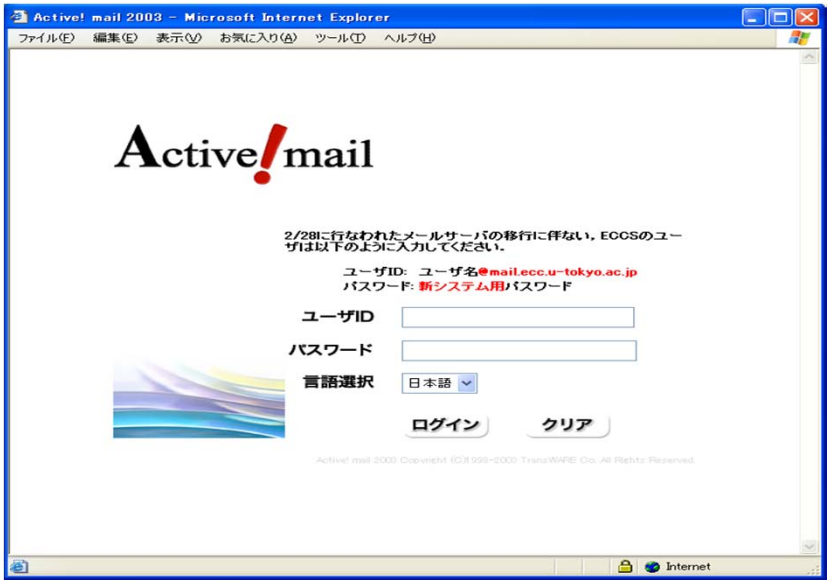
Mail Service

Mail address: *LoginName*@mail.ecc.u-tokyo.ac.jp

Mail client

Mail (need setting)

Web Mail: https://wm.ecc.u-tokyo.ac.jp/



Terminal Rooms

Asano Campus

- Information Technology Center 1F 5F

Komaba Campus

- Information eduacation Center 1F-4F
- Komaba Library 2F

Hongo Campus

- Main Library 2F, 3F
- International Student Center
- School of Eng. Bldg.1, 2F
- Law, Bldg.1, 1F

...

Computer Programming

IIE (799-18)

(2) May13, 2004

Lecturer

Ying CHEN （陳　迎）

Department of Quantum Engineering and System Science

Teaching Assistant

Miti Ruchanurucks

Graduate School of Information Science and Technology

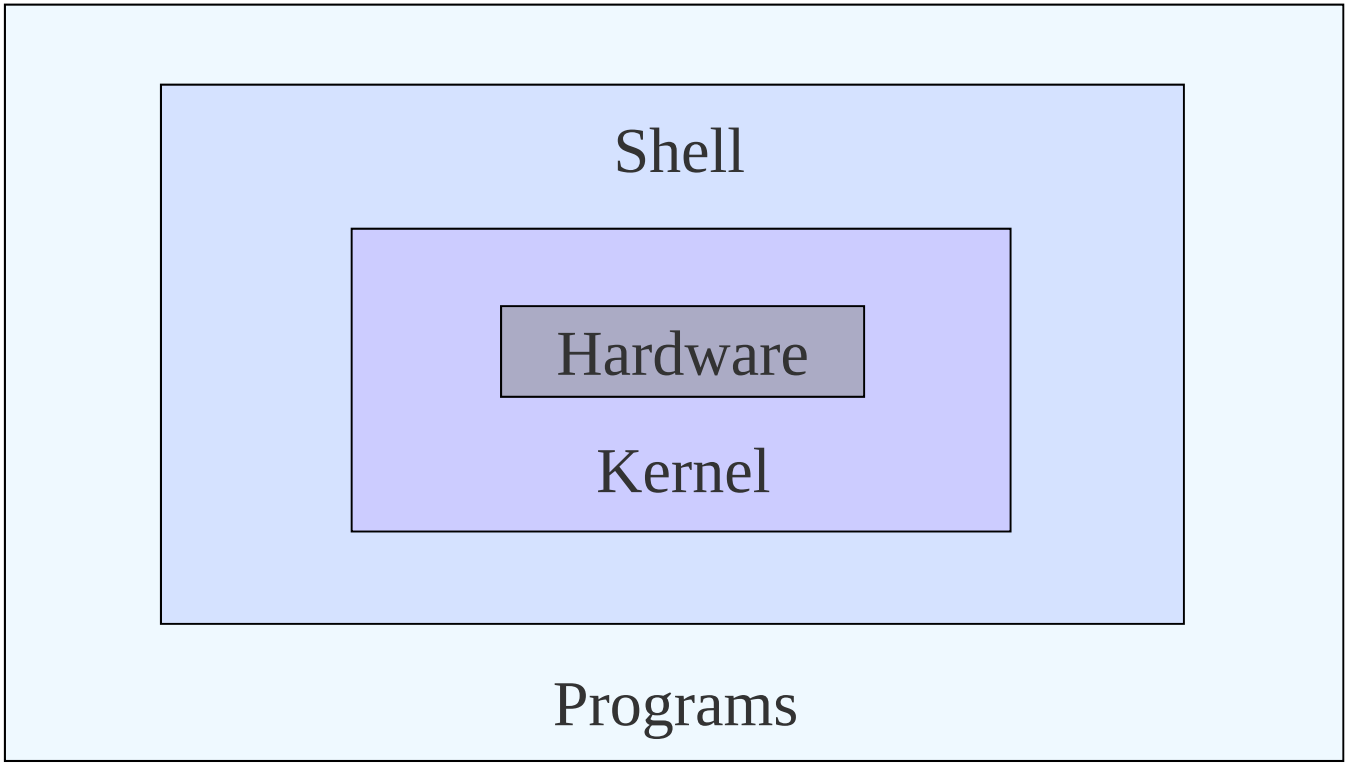
Today

- **Simple Unix commands**
- **Java: Getting started**

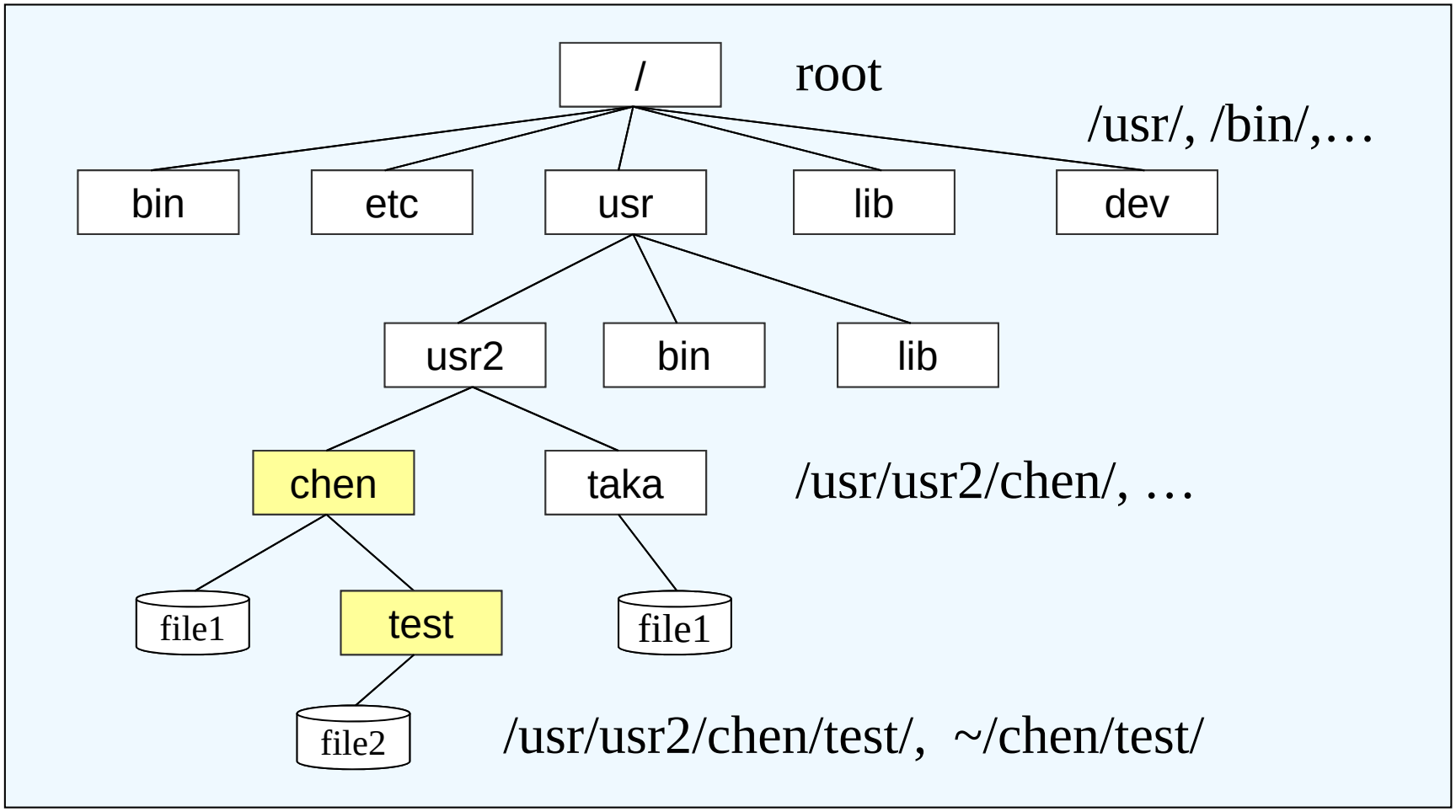
Now, Please login...

and open a X11 window

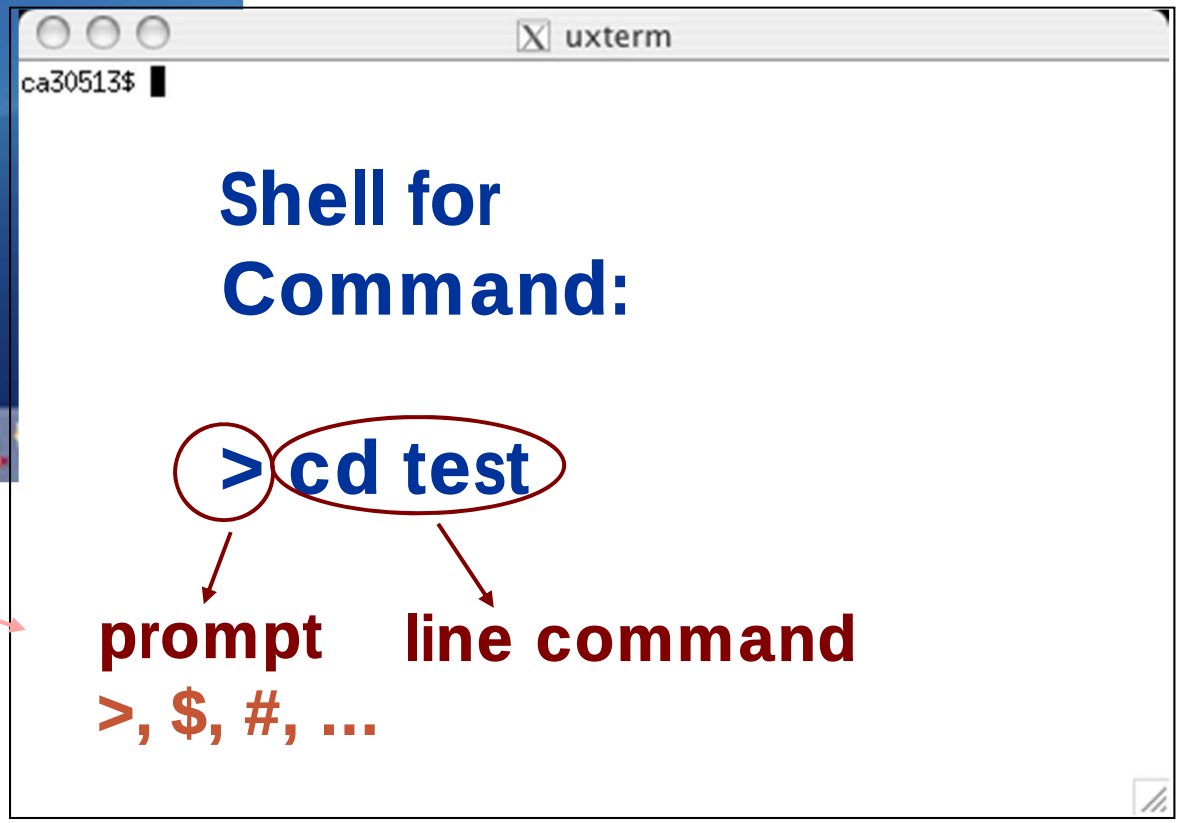
Unix: Operating System of workstation and multi-users server



Unix: File System, Path



Unix Environment (X11)



Simple Unix Commands

Try …, What happens ?

> date

```
tying@as301> date
Sun May 11 22:02:41 JST 2003
```

> pwd

```
tying@as301> pwd
/home/tying
```

> history

```
tying@as301> history
    17  8:41    cd ..
    18  8:41    ls -l
    19  8:41    rm yyy
    20  8:42    mv test2 mytest
    21  8:42    vi mytest
    22  8:42    lf
    23  8:42    ls
    24  8:42    date
    25  8:42    pwd
    26  8:43    history
```

Simple Unix Commands

Try More ... list contents of directory

> ls

```
tying@as301> ls
GNUstep          atokdicts.tar  registry
GNUstep.sun      mytest         test
```

> ls -F

```
tying@as301> ls -F
GNUstep/         atokdicts.tar  registry/
GNUstep.sun/     mytest         test/
```

> ls -l

> ls -a

Simple Unix Commands

Making /Deleting directory

> mkdir test11.6

```
tying@as301> mkdir test 5.13
tying@as301>
tying@as301> ls
GNUstep          atokdicts.tar  registry        test 5.13
GNUstep.sun      mytest         test
tying@as301> ls -F
GNUstep/         atokdicts.tar  registry/       test 5.13/
GNUstep.sun/     mytest         test/
```

> rmdir *dir_name*

Simple Unix Commands

Change directory

> cd test513

```
tying@as301> cd test 5.13
tying@as301>
tying@as301> pwd
/home/tying/test 5.13
```

> cd.. Go one directory up hierarchy

```
tying@as301> cd ..
tying@as301> pwd
/home/tying
```

> cd Return to home directory

Simple Unix Commands

Editing a file: Emacs

```
> emacs
> emacs &
> emacs testfile &
```

different ?

Command: 2 ways

- menu on top
- line command in bottom

Save file: **c-x c-s**

Quit emacs: **c-x c-c**

Open file: **c-x c-f**

……

Practice 1

1) Make some files in your Windows system;

2) Read the windows files from Unix:
   > cd /windows/profile/
   > less *file_name*

3) Read the iMac files from Unix:
   > cd /Desktop
   > less *file_name*

Simple Unix Commands

Displaying a file

> less *file_name*

Deleting a file

> rm *file_name*

> rm -i *file_name*

Copying a file

> cp *file_original* *file_new*

Renaming a file/directory

> mv *file_old* *file_new*

Simple Unix Commands

Redirectory: >

Write output of command into a file

eg.
> ls –l > filelist
> less filelist
> wc filelist

Pipe: |

Output of 1st command as input of 2nd command

eg.
> ls –l | wc

Practice 2

1) Make another directory inside test5.13/ called “practice2”;

2) Make a file “hello” inside test5.13/practice2/;

3) Display the content of file “hello”;

3) Use “ls”, “pwd”, “cd” to explore your file system, Find the full pathname of your file “hello”.

Getting Help: Online Manual

```
> man command
```

```
> man ls
> man cp
> man mv
```

```
> man grep  ...
```

Simple Unix Commands

Changing Unix Environment
Japanese → English

> emacs .bashrc &

in emacs:
add 2 lines to the end of file ".bashrc:"
LANG=C (followed by return)
export=LANG

save .bashrc

>source .bashrc

> man ls → in English?

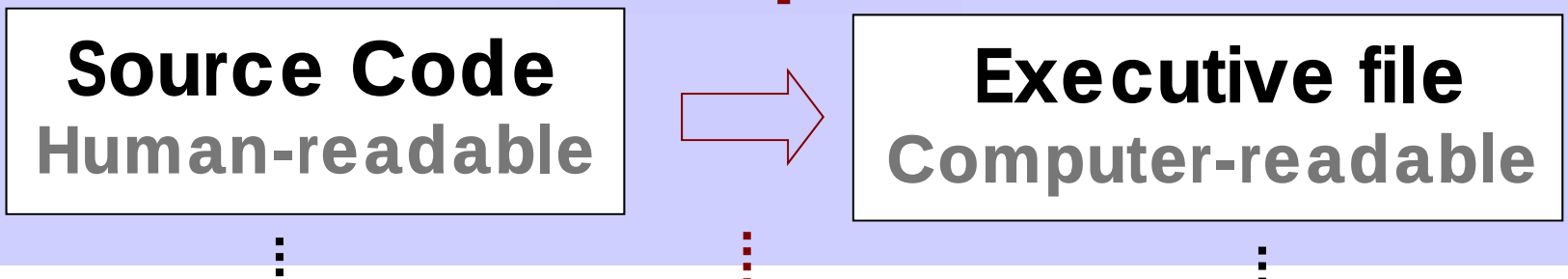
Java: Getting Started – Basic Idea

A program

A list of commands stored in a text file

How does a program work?

Compiler



Create a source file

Compile the source file

Run the program (Executive file)

Practice 3: “Your first cup of Java”

1) Create a java source file

unix

```
> mkdir practice3
> cd practice3
> emacs Hello.java &
```

“*Hello.java*”

emacs

```
/*  My first cup of java  */
class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

⇨ *save*

Practice 3: “Your first cup of Java”

2) Complie source file

```
> javac Hello.java
```

→ Hello.class

3) Run the program

```
> java Hello
```

See What happens?

Study after class

- Unix tutorial for beginners:

  http://www.ee.surrey.ac.uk/Teaching/Unix/

- PC: download: Java™ 2 Platform for PC

  http://java.sun.com/j2se/1.4.2/index...??

Computer Programming

IIE (799-18)

(3) May20, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

Study after class

- PC: download: Java™ 2 Platform for PC(J2SE)

  http://java.sun.com/j2se/index

- Unix tutorial for beginners:

  http://www.ee.surrey.ac.uk/Teaching/Unix/

- cygwin: a Linux-like environment for Windows.

  Information on Installation: Course HP:

  http://gwp01.t.u-tokyo.ac.jp/kouryu/students/classes/ica.htm

Today

Now, Please login to your Unix ...

Java: 1995, Sun Microsystem Inc.

Java:
-Portable
-Object-oriented
-Internet appl.

Content of Java Programming

1. Getting Started
2. Language Basics
3. Control Flow Statement

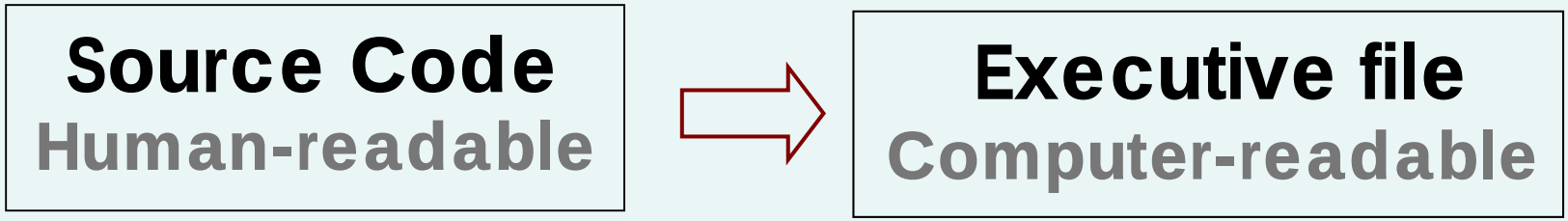
...

A program

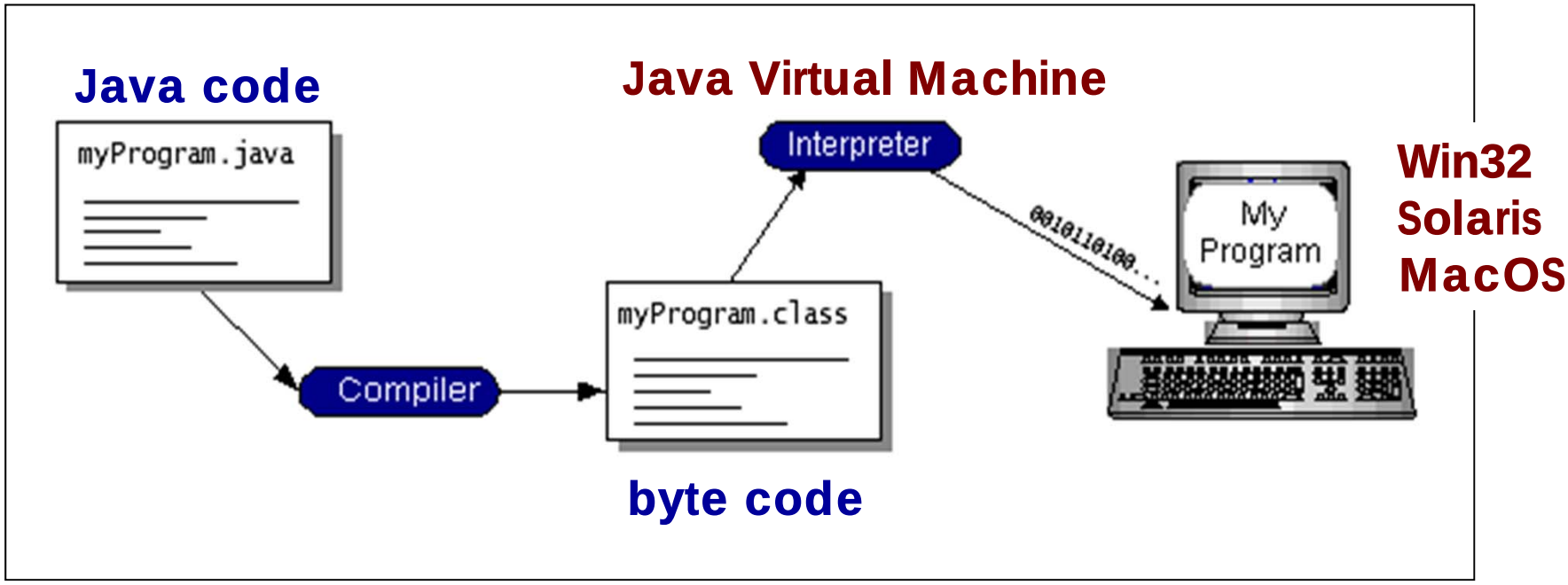
A list of commands stored in a text file

How does a program work?

Compiler/interpreter



1. Getting Started — Java Program



Create a source file

Compile the source file

Run the program (Executive file)

Practice 3: “Your first cup of Java”

1) Create a java source file

Unix

```
> mkdir practice3
> cd practice3
> emacs Hello.java &
```

Case sensitive !

“*Hello.java*”

emacs

```
/*  My first cup of java  */
class Hello {
    public static void main(String[] args) {
        System.out.println(“My first java!");
    }
}
```

Practice 3: “Your first cup of Java”

2) Complie source file

```
> javac Hello.java
```

→ Hello.class

3) Run the program

```
> java Hello
```

See What happens?

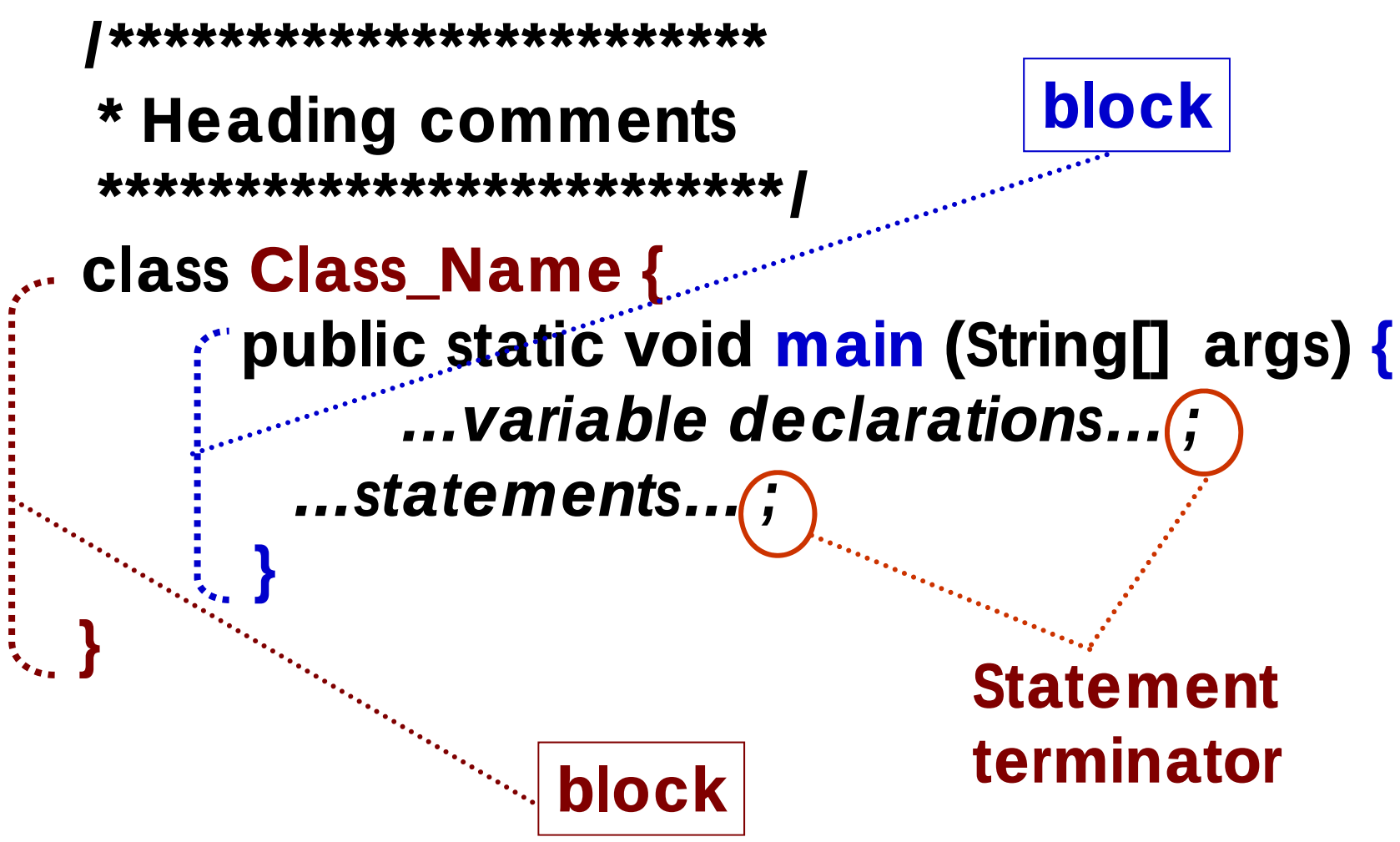
1. Getting Started Running a Java Program

1) Creating Java source code;
   source_name.java

2) Compilering source:
   > javac *source_name.java*
   → *source_name.class*
   bytecode

3) Running:
   > java *bytecode_name* *(without ".class")*

1. Getting Started Basic Program Structure



Practice 3(cont.) Modification

Hello.java

```
/*  My first cup of java
 *                2004.5.20.        */     //comment line
class First {                          // define a class
    public static void main(String[] args)  //main methood
{
          System.out.println("Hello World!");   //print
          int a =18;                            //assign statement
          System.out.println("a="+a) ;
    }
}
```

> javac Hello.java -> to see if you create "Hello.class?

> java Hello

- Keep it simple and stupid (KISS).
- Care about the indentation.
- Always declare what you're going to do by using the comment box or line:

  "/* ..... */" or "//....."

Variables

A *variable* is an item of data named by an identifier.

- Numbers , characters, strings, etc etc
- Begins with alphabets, _ , $
- Avoid using keywords: if, for, …

Variable Declarations

Format: ***Type Name;***

eg. Int sum; float tax; char c;

2. Language Basics — Basic Data Type

Keyword	Description	Size, range
	integers	
byte	Byte-length integer	8-bit, -128~127
short	Short integer	16-bit, -32768~32767
int	Integer	32-bit
long	Long integer	64-bit
	Real numbers	
float	Single-precision floating	32-bit
double	Double-precision floating	64-bit
	other types	
boolean	A boolean value	true or false
char	A single character	16-bit
(string	a set of characters	A Class)

Cast: type conversion, eg. (int)2.73, (double)x

Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder

3%5=3
5%5=0

Relational Operators
To determine the relationship between two values

Operator	Meaning
<=	Less than or equal to
<	Less than
>	Greater than
>=	Greater than or equal to
==	Equal
!=	Not equal

Logical Operators
To form multi-part decisions

	Operator	Use	Meaning
and	&&	op1 && op2	Return true if op1 and op2 are both true, conditionally evaluate op2.
or	||	op1||op2	Return true if either op1 and op2 are true, conditionally evaluate op2.
not	!	!op	Return true if op is false.
and	&	op1 & op2	Return true if op1 and op2 are both true, always evaluate op1 and op2.
or	|	op1||op2	Return true if either op1 and op2 are true, always evaluate op1 and op2.

2. Language Basics Assignment Statement

Format: var = expression;

eg. sum = (12 + 21 - 23)* 41 + 15;

average = sum / 5;

sum = sum + 26.2;

char1 = 'A';

string1 = "The University of Tokyo";

Declaring + Initializing Variables

Separately:

```
int a, b;
a = 7000;
b = 60;
```

Simultaneously:

```
int a = 7000;
int b = 60;

char D= 'm';
String Dept = "Quantum Eng.";
```

Practice 3(cont.): Add calculation

Hello.java

3 integers: a=18, b=4, c

calculate:
c=a%b+(a/b+2)=?

display result like:

My first cup of java:
a= ... b= ...
c= ...

Practice 3(cont.) modification

Hello.java

```
/*  My first cup of java
 *                2004. 5. 20.       */              //comment line
class First  {                                       // define a class
     public static void main(String[] args)  //main methood
      {
            System.out.println("Hello World!");          //print
            int a=18; int b=4;
            int c;
            c=a%b+(a/b+2);

System.out.println("a="+a+"\t"+"b="+b+"\n"+"c="+c) ;
      }
 }
```

System.out.println: printout + change line

```
System.out.println("a="+a+"\t"+"b="+b+"\n"+"c="+c)
```

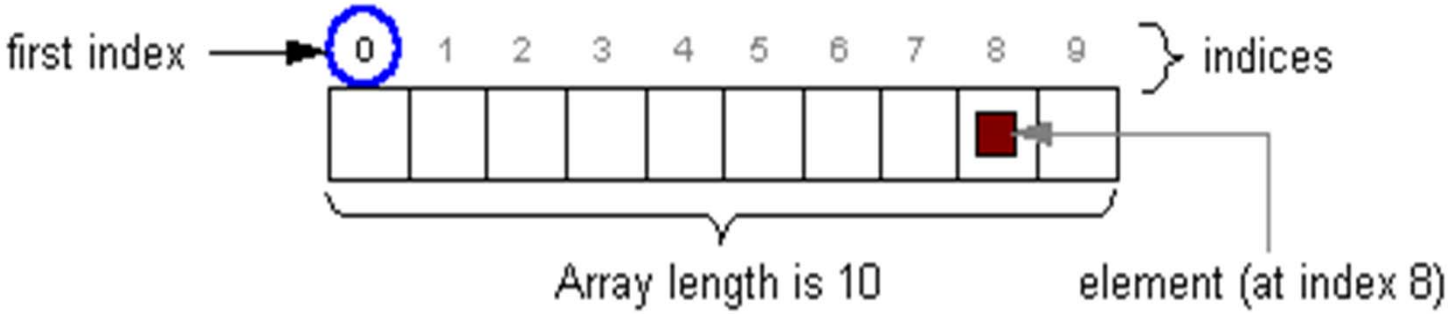
Tab

Changing line

```
a= 18      b= 4

c= ...
```

System.out.print: printout (without change line)

An array: a list of data of the same type.



Declaring: type[ ] *ArrayName* = new type[size]

eg. **An one dimentional array:** **oneArray**
An array element: **oneArray[i]**
length of array: **oneArray.length**

Declaring + Initializing An Array

eg. Separately:

2	3	1	6	5

```
int [ ] oneArray= new int[5];
oneArray[0] = 2;
oneArray[1] = 3;
oneArray[2] = 1;
oneArray[3] = 6;
oneArray[4] = 5;
```

Simultaneously:

```
int [ ] oneArray = {2,3,1,6,5} ;
```

Multidinmentional Arrays

type[ ][ ] *ArrayName* = new type[size1][size2]

eg.

```
int [] [] array = new int[2][2];
array[0][0] = 2;
array[0][1] = 4;
array[1][0] = 1;
array[1][1] = 5;
```

$$\begin{pmatrix} 2 & 4 \\ 1 & 5 \end{pmatrix}$$

or:

```
Int [2] [2] array = { {2,4}, {1,5} } ;
```

3. Control Flow Statement

To specifies the order
in which each statement is executed.

if ... else
while
break
continue
for ...
...

if ... else Statement

```
if (condition) {
    Actions to be performed if the condition is true;
} else {
    Actions to be performed if the condition is false;
}
```

Condition: <exp> conditional op. <exp>

>, <, ==, ...

3. Control Flow Statement if … else

Don’t confuse

a==b with a=b!

eg.

```
if (a==3) {
   a=10;
   System.out.println( “a=“+a) ;
}
else {
  System.out.println (“a=“+a) ;
}
```

“if nest”

```
if (condition1) {
    if (condition2) {
    statement1;
    }
    else {
    statement2;
    }
}
else ...
```

3. Control Flow Statement for

for loop:
Iterate action over a range of values.

```
for ( initial; terminal condition; increment)
{
        code to be iterated;
}
```

Practice 4

To Print out “9x9” Table

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Practice 4

Practice4.java

```
/* Prac4. To print out 9x9 Table.
                                2003.5.20.      */
class Practice4  {
     public static void main(String[] args)  {
          int i=0, j=0;
          int [ ][ ] kuku = new int[9][9];
          for (i=0;i<9;i++)   {
               for (j=0;j<9;j++)   {
                    kuku[i][j]=(i+1)*(j+1);
                    System.out.print(kuku[i][j]+"\t");
               }
          System.out.print("\n");
          }
     }
}
```

i++: i=i+1

Exercise 1

Write a program to give all prime number between 100~200.

- An integer greater than one is called a **prime number** if its only positive divisors (factors) are one and itself. (For example, the prime divisors of 10 are 2 and 5; and the first six primes are 2, 3, 5, 7, 11 and 13.) ;
- m is a prime number if it has no integer quotient when divided by integer between 2~$\sqrt{m}$;
- $\sqrt{m}$ in Java is: Math.sqrt(m), m is double floating.

Request on Exercises

- Subject of the email should include "CP-IIE-exe1".
- Subject or Body should include "Student Number" if you don't use ECC account.
- You should Run the program before sending.

Email to us by June 3.

ying@q.t.u-tokyo.ac.jp

miti@ cvl.iis.u-tokyo.ac.jp

Computer Programming

IIE (799-18)

(4) May 27, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

Content of Java Programming

1. Getting Started (3) 5/20
2. Language Basics
3. Control Flow Statement
4. Basic of Object-Oriented Programming (4) 5/27
...

Now, Please login to your Unix ...

No lesson next week (June 3)

3. Control Flow Statement

To specifies the order
in which each statement is executed.

if ... else
for ... do
while
break
continue
...

3. Control Flow Statement if...else

if ... else Statement

```
if (condition) {
    Actions to be performed if the condition is true;
} else {
    Actions to be performed if the condition is false;
}
```

Condition: <exp> conditional op. <exp>

>, <, ==, ...

3. Control Flow Statement for

for loop:
Iterate action over a range of values.

```
for ( initial; terminal condition; increment)
{
        code to be iterated;
}
```

3. Control Flow Statement while

while statement

```
Initiation
while (termination)  {
        body;
        iteration;  }
```

do-while statement

```
Initiation
do {    body;
        iteration;  }
while (termination);
```

3. Control Flow Statement break

break:

Exit the loop/iteration to run the next statement.

eg.

```
class testbreak {
public static void main(String args[])  {
   int i=0;
   while (i<100) {
      System.out.println("i="+i);
            if (i== 5) break;
      i++ ;
      }
   System.out.println("final i="+i);
   }
}
```

3. Control Flow Statement continue

continue:
Skip next action and go to beginning of loop.

```
eg.
    class testcont {
    public static void main(String args[])  {
    outer: for (int i=1;i<9; i++)  {
            inner:  for (int j=1;j<9; j++)     {
                        System.out.print (j*i+"  ");
                        if (i== j)  { System.out.print ("\n");
                        continue outer;    }
          }
       }
    }
    }
```

Labled

3. Control Flow Statement continue

Result:

```
1
2  4
3  6  9
4  8  12  16
5  10  15  20  25
6  12  18  24  30  36
7  14  21  28  35  42  49
8  16  24  32  40  48  56  64
9  18  27  36  45  54  63  72  81
```

4. Basic of Object-Oriented Programming (OOP)

Basic Concepts

Procedure-Oriented programming:

algorithm + data structure → code (functions)

Object-Oriented programming:

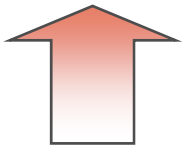
data structure + algorithm → code (methods)

4. Basic of Object-Oriented Programming (OOP)

Basic Concept

Object: basic unit of OOP

In real world
In software



Instance

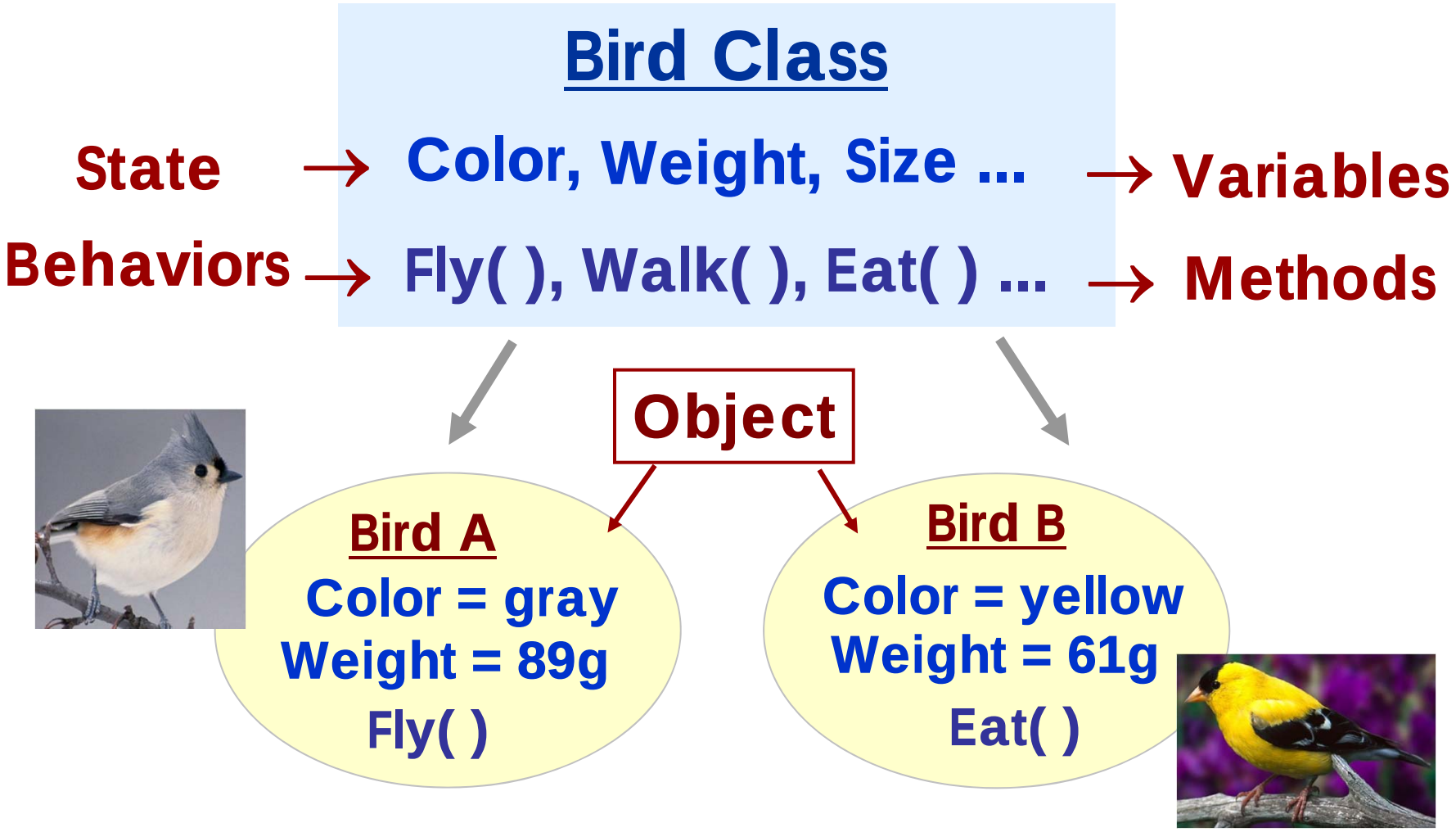
Class

describes the generality of objects

State → **Variables**

Behavior → **Methods**

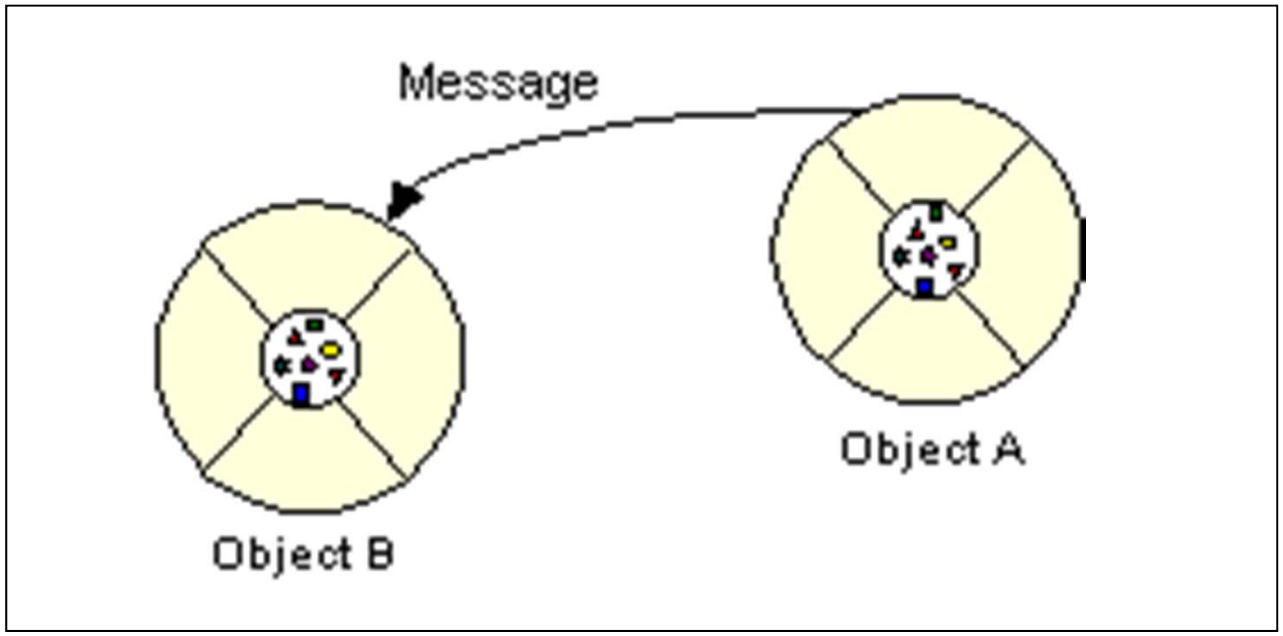
4. Basic of Object-Oriented Programming (OOP)



4. Basic of Object-Oriented Programming (OOP)

Basic Concept

Message



4. Basic of Object-Oriented Programming (OOP)

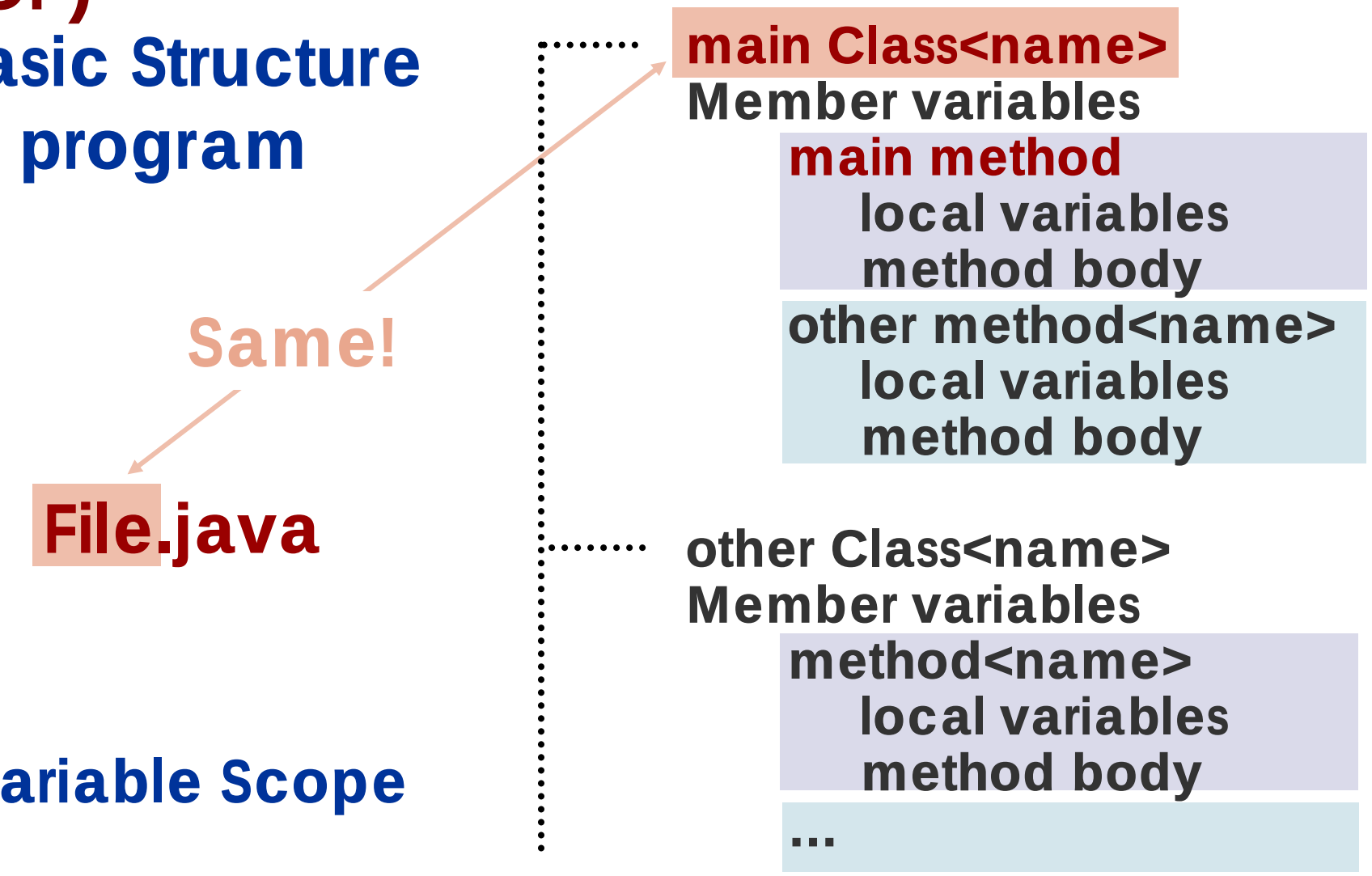
Basic Concept

Characterastics of Object-Oriented programming

- Encapsulation
- Inheritance
- Polymophism

4. Basic of Object-Oriented Programming (OOP)

Basic Structure of program



Variable Scope

Method

- Specify the behavior of an object.
- A unit of codes that do something you need.
- **main** is an special method, and you can also define your own methods.
- Can be called from anywhere (or almost).

Definition of a Method

```
methodDeclaration          {
        local variables
        method body
}
```

public,
private, ...

Declaration of a Method

```
accessLevel modifier returnType methodName
(type name [type name[...] ] )  {
......
}
```

static,
final, ...

double,
int,
void,
...

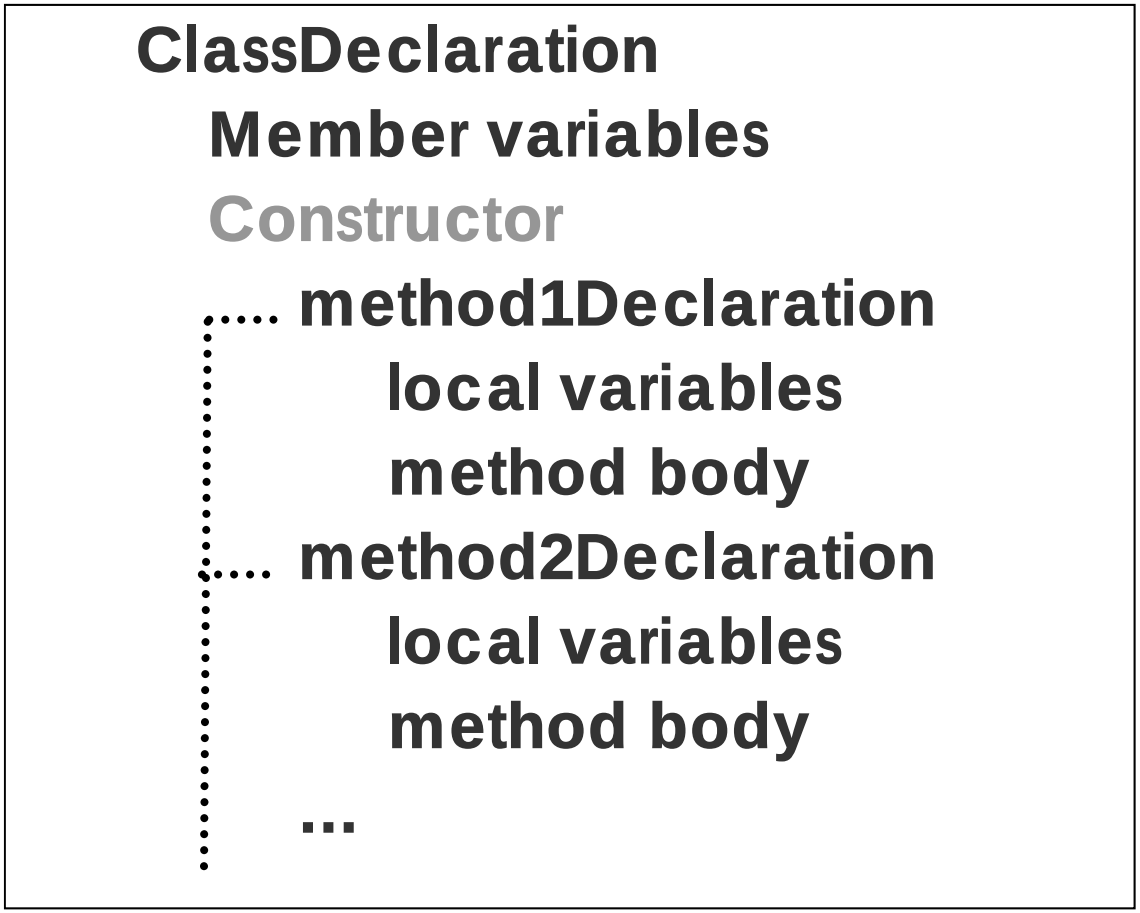
eg. 1

```
int summation(int a, int b ) {
    return(a+b);
}
```

eg. 2

```
public static void main (String arg[ ] ) {
    System.out.println(“My first java!");
}
```

Creating a Class



Declaration of a Class

```
[public] [abstract/final] class ClassName[...] {

.....

}
```

Practice 5-1 Creating a class

Bird.java

```
/* Prac5-1, Bird Class, 2004.5.27.   */
class Bird {
    String color ;
    int weight;
    boolean hungry ;
    public void figure( ) {
       System.out.println("color :"+color);
       System.out.println ("weight :"+weight);
  }
 public void meal ( ) {
    if (hungry) {
       System.out.println("looking for seeds to eat");
    } else {
       System.out.println ("finding toys to play");
      }
 }
}
```

Practice 5-1 Creating a class

To compile the “Bird.java” → “Bird.class”, and run,
→ To see what happens?

```
tying@as301> java Bird

Exception in thread "main"
java.lang.NoSuchMethodError: main
```

How to Create an Object from a Class

Define an object variable (Instance):

className objectName;

Creating a new object (new operator):

objectName = new className ( );

OR

className objectName = new className ( );

eg. Bird BirdA= new Bird;

Using Objects

- Referencing an Object's Variables

  objectName.variableName

- Calling an Object's Method

  objectName.methodName( );

  objectName.methodName(argumentList);

;

eg. Bird BirdA= new Bird;

BirdA.color = "gray";

BirdA.weight = 89;

BirdA.meal(); BirdA.figure();

Practice 5-2 Creating objects

pract52.java

```
/* Prac5-2, object/instance, 2004.5.27.   */
 public class Pract52 {
 public static void main (String [ ] args ) {
      Bird A = new Bird( );
      Bird B = new Bird( );
      A.color="gray" ; A.weight=10; A.hungry=true;
      B.color ="white"; B.weight=24; B.hungry=false;
      System.out.println("This is A bird");
      A.figure( );
      A.meal();
      System.out.println("This is B bird");
      B.figure( );
      B.meal();
   }
 }
```

Practice 5-2 (continue)

- **Keep "Pract52.java" in a new directory:**
  To compile the "pract52.java
  → To see what happens?

```
tying@as301> javac pract52.java
Birdsample.java:3: cannot resolve symbol
symbol  : class Bird
location: class Birdsample
Bird A;  ….
```

- **To confirm "Pract52.java" and Bird.class are in same directory,**
  then run … **→ ?**

Practice 5-3 Combining 2 classes

To write 2 classes into one program
Pract5.java
Compile, ...
Run, ...

```
This is A bird
color :gray
weight :89
looking for seeds to eat
This is B bird
color :yellow
weight :61
finding toys to play
```

Request on Exercises

- Subject of the email should include "CP-IIE-exe1".
- Subject or Body should include "Student Number" if you don't use ECC account.
- You should Run the program before sending.

Email to us by June 3.

ying@q.t.u-tokyo.ac.jp

miti@ cvl.iis.u-tokyo.ac.jp

No lesson next week (June 3), See you on June 10.

Computer Programming

IIE (799-18)

(5) June 6, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

Content of Java Programming

1. Getting Started (3) 5/20
2. Language Basics
3. Control Flow Statement (4) 5/27
4. Basic of Object-Oriented Programming
5. Inheritance and Polymophism (5) 6/6

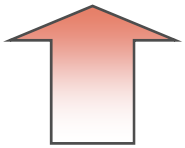
...

Now, Please login to your Unix ...

4. Basic of OOP

Basic concepts

Object: basic unit of OOP



Instance

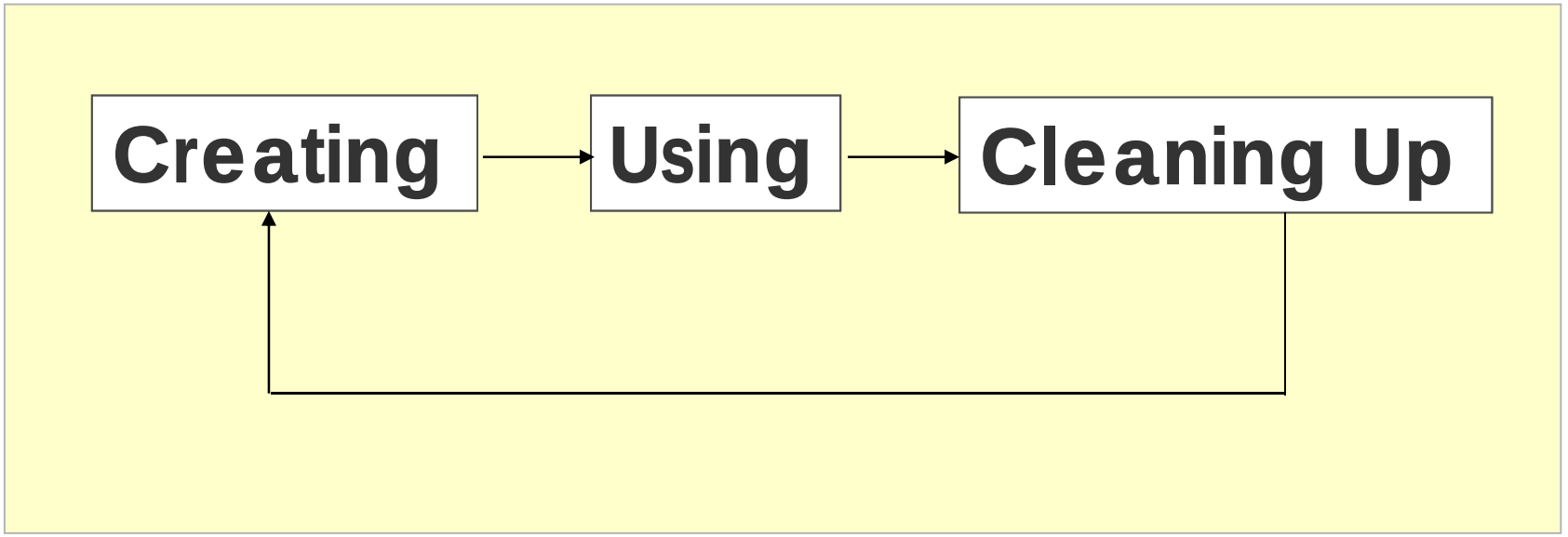
Class

describes the generality of objects

State → Variables

Behavior → Methods

The Life Cycle of an Object



Creating Objects

- Declaration: ClassName objectName;
- Instantiation: new ClassName ( );

  OR

  ClassName objectName = new ClassName ( );
- Initialization:
  - objectName.variableName;
  - objectName.methodName;

    objectName.methodName( );
  - constructor

Using Objects

- Referencing an Object's Variables

  objectName.variableName

- Calling an Object's Method

  objectName.methodName( );

  objectName.methodName(argumentList);

eg. Practice 5-1 creating a class

Bird.java

```
/* Prac5-1, bird class, 2003.11.20.   */
class Bird {
    String color ;
    int weight;
    boolean hungry ;
    public void figure( ) {
      System.out.println("color :"+color);
      System.out.println ("weight :"+weight);
     }
    public void meal ( ) {
      if (hungry) {
        System.out.println("looking for seeds to eat");
         } else {
         System.out.println ("finding toys to play");
       }
    }
}
```

eg. Practice 5-2 creating object

pract52.java

```
/* Prac5-2, object/instance, 2003.11.20. */
 public class Pract52 {
 public static void main (String [ ] args ) {
     Bird A = new Bird( );
     Bird B = new Bird( );
     A.color="gray" ; A.weight=10; A.hungry=true;
     B.color ="white"; B.weight=24; B.hungry=false;
     System.out.println("This is A bird");
     A.figure( );
     A.meal();
     System.out.println("This is B bird");
     B.figure( );
     B.meal();
   }
}
```

Declaration
Instantiation

Referencing
Object's
Variables,
Initializing

Calling
Object's
methods

Cleaning Up Unused Objects

- The garbage Collector
- Finalization

Constructor of Class

- Constructor is an special method to initialize the variables
- Its name must be the same as that of the class

ClassDeclaration

Class variables;

```
ClassName (arg1, arg2 ...) {
        class_var.1= arg1;
        ...
 }
```

Constructo

method1Declaration

local variables;

method body

...

```
eg.
Bird (String x, int y, boolean z ...)
 {
   color = x;
   weight= y;
   hungry= z;
}
```

Creating Object by Constructor

```
ClassName objectName = new ClassName ( );
        objectName.variable = ...
```

```
ClassName objectName = new ClassName (x,y,... );
```

Practice 5-4: Re-write Prac5.java by constructor

In Main method:

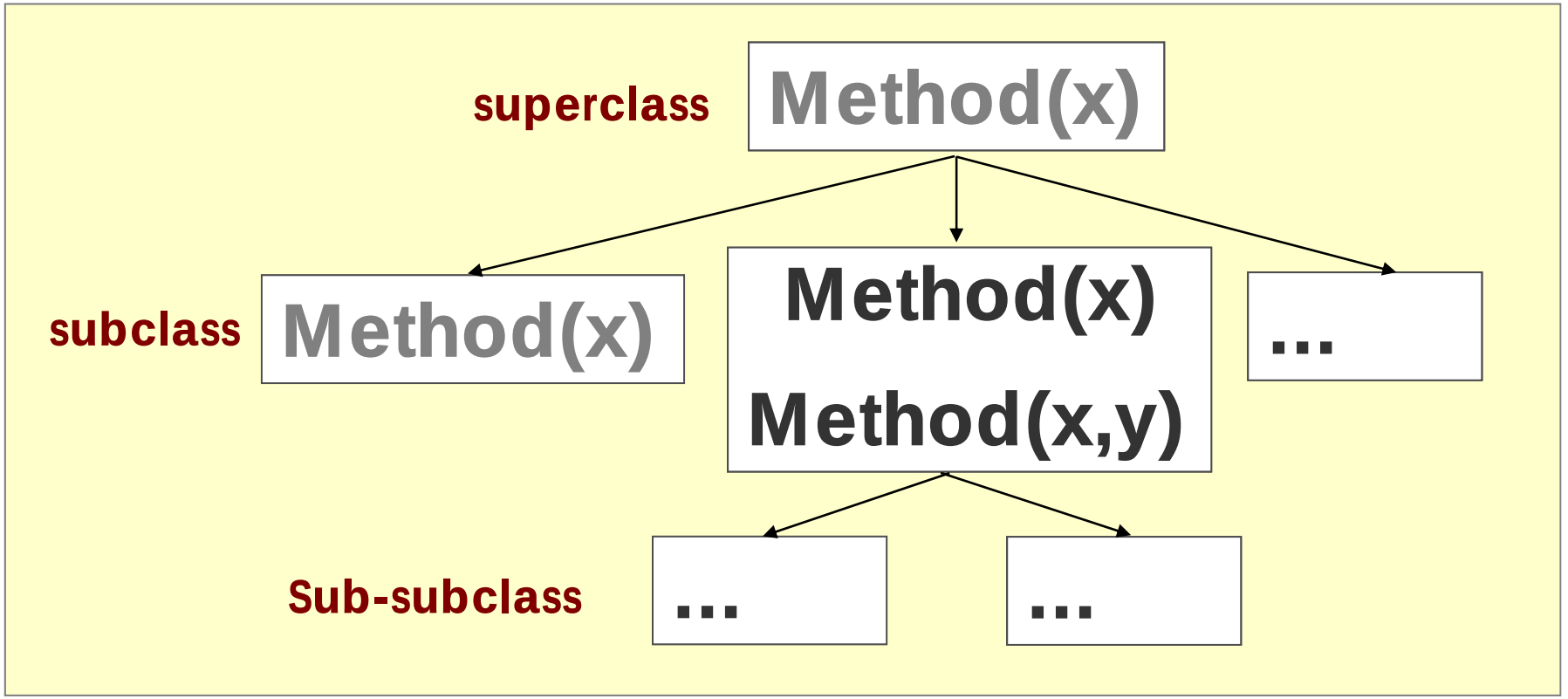
```
Bird A= new Bird("gray",89,true);
Bird B= new Bird("yellow",61,false);
```

In class "Bird":

```
...
String color;
int weight;
boolean hungry;

Bird (String x,int y,boolean z)   {   // constructor
      color=x ;
      weight=y;
      hungry=z ;      }
...
```

5. Inheritance and Polymophism

Inheritance



Polymophism: Override, Overload

5. Inheritance and Polymophism

Inheritance

Allowed to define new (child) object types as extensions of the old one (father).

Feature

- Inherits all variables, methods;
- provide additional functions of its own;
- change the original behavious.

```
class SubclassName extends SuperclassName {
        body of class    }
```

5. Inheritance and Polymophism

keyword “this” and “super”

this: current object	super: super class of current object
this.variable	super.variable
this.Method( )	super.Method( )
this ( )	super ( )

Practice 6 Inheritance

Prac6.java

```
public class Prac6 {
  public static void main (String [ ] args ) {
     Superclass s1 =new Superclass( );
     Subclass s2 =new Subclass( );
     s1.showSuper( );
     s2.showSub( );
     s2.showSuper( );
  }
}
class Superclass {
     String classname = “Superclass Q “;
     public void showSuper( ) {
     System.out.println(“ The method of “ +classname);
  }
}
 class Subclass extends Superclass{
     public void showSub( ) {
     System.out.println(“ The method of Subclass “);
     System.out.println(classname +” is a ? class “);
  }
}
```

Overload

Inside a class, it is possible to define several methods(constructors) in **same name**, but **different content, different arguments**.

Calling overloaded method is according to the **number and type of arguments**.

Overload of Method

Sample1.java

```
class Sample1 {
void printValue(int i) {
System.out.print("i=  "+ i+"\n" );
 }
void printValue(int i,double f) {
System.out.print("i=  "+ i+"\n" );
System.out.print("f=  "+ f+"\n" );
 }
public static void main (String [ ] args ) {
  Sample1 o=new Sample1();
     o.printValue(100);
     o.printValue(10,1.2);
   }
}
```

Overload of Constructor

Sample2.java

```
class Sample2 {
   private int a;
   private int b;
   private double x;
   Sample2(double k ) {
      x=k;
    }
  Sample2(int i, int j ) {
      a=i;
      b=j;
   }
   public static void main (String [ ] args ) {
      Sample2 o1=new Sample2(45.78);
      System.out.println("a="+o1.x);
      Sample2 o2=new Sample2(10,12);
      System.out.println("a="+o2.a+"   b="+o2.b);
   }
}
```

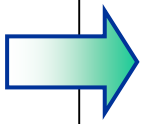
Override

In subclass, it is possible to define the variables/methods in **same name** as ones in superclass (**number and type of arguments are also same**), but different content. Overrided method in subclass can hide some variable and change feature of method in superclass.

5. Inheritance and Polymophism — Override

TestOverride.java

```
public class TestOverride {
    public static void main(String arg[]) {
        B sub0 = new B(6,9);
        sub0.set(7,8);
        int m=sub0.multiply();
       System.out.println("m= "+m);     }
}
class A {
     int i,j;
     void set(int i, int j)  {
         this.i=i;
         this.j=j;    }
     int multiply() {
         return i*j;    }
 }
class B extends A {
      int i,k;
      B(int i, int k) {
            this.i=i;
            this.k=k;    }
      int multiply()  {
           return i*j*k;  }
}
```



Result: m=432

Exercise 2 (submit by Dec. 10)

Solve the quadratic equation: $ax^2+bx+c=0$

$a=0$, not a quadratic equation;
$b^2-4ac=0$, 2 same real roots.
$b^2-4ac>0$, 2 different real roots: $-\frac{b}{2a}\pm\frac{\sqrt{b^2-4ac}}{2a}$;

$b^2-4ac<0$, 2 conjugate complex roots: $-\frac{b}{2a}\pm i\frac{\sqrt{-(b^2-4ac)}}{2a}$.

1) Create a class to hold the variables a,b and c;
2) Extend above class to create a class with the constructor to initialize the variables, and method(s) to calculate the roots of the equation;
3) Execute calculations with certain (a,b,c) for above 4 cease.

Computer Programming

IIE (799-18)

(6) June 17, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

Content of Java Programming

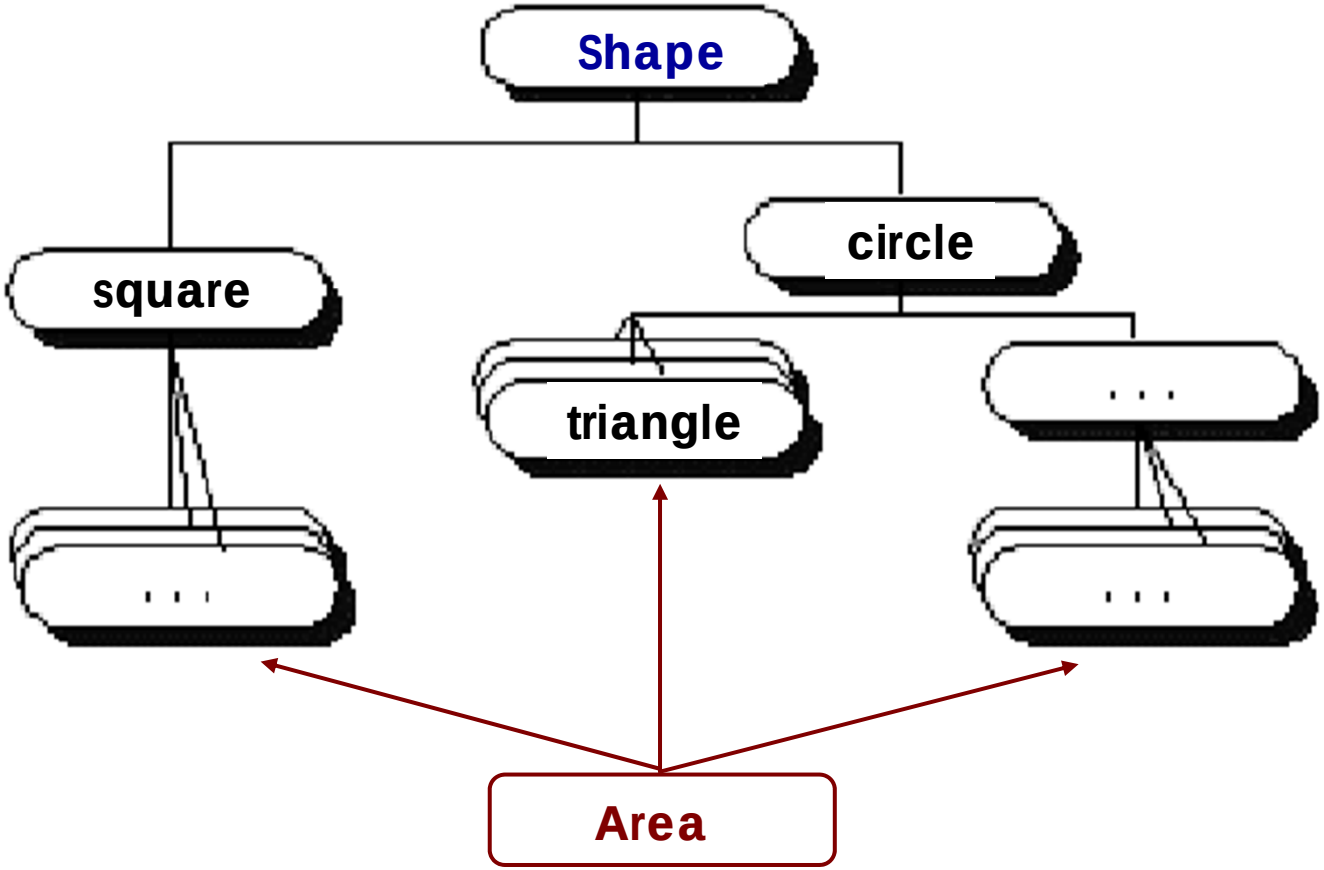
1. Getting Started (3) 5/20
2. Language Basics
3. Control Flow Statement
4. Basic of Object-Oriented Programming (4) 5/27
5. Inheritance and Polymophism (5) 6/10
6. Interface and Package
7. Modifiers
8. Exception
9. Input/Output Stream (6) 6/17

Today

...

Please login to your Unix ...

Single Inheritance and Multiple Inheritance



An **interface** is a named collection of method definitions (without implementations).

Creation

```
[public] interface interfaceName [extends ...] {
        variable (all are constants);
        ...
        method (all are abstract, no body);
        ...
}
```

An interface defines a protocol of behavior.
A class implements an interface means realizing then implementing all methods of the that interface inside the class.
Java supports single inheritance for class, but multiple inheritance for interface.

Implementation

include an implements clause in the class declaration:

```
Class className[extends...] implements interfaceList {
... (implement all abstract methods)...    }
```

eg. "DemoInterface.java" :

```
interface Shape {                          // declare interface
      int x=50, y=50;                      // constant
      void draw( );              }       // declare method
interface Area {                           // declare interface
     double calculate( );     }           // declare method
```

```
class Square implements Shape,Area  {
     private int length;
     void setSize(int l) {   length=l;    }
     public void draw( )  {                  //realize "draw"
            System.out.println ("This is a square");   }
     public double calculate()  {   //realize "calculate"
            return length*length;    }
}
```

```
class Circle implements Shape,Area  { ...}
```

```
class Triangle implements Shape,Area  { ...}
```

...

eg. (continue)

```
public class DemoInterface {                    //main class
      public static void main( String arg{[ ]) {  //main method
      Square sq=new Square();                   //instance "Square"
      ...
      sq.draw();
      sq.setSize(5);
      System.out.println("Area of square is "+sq.calculate());
      ...
      }
}
```

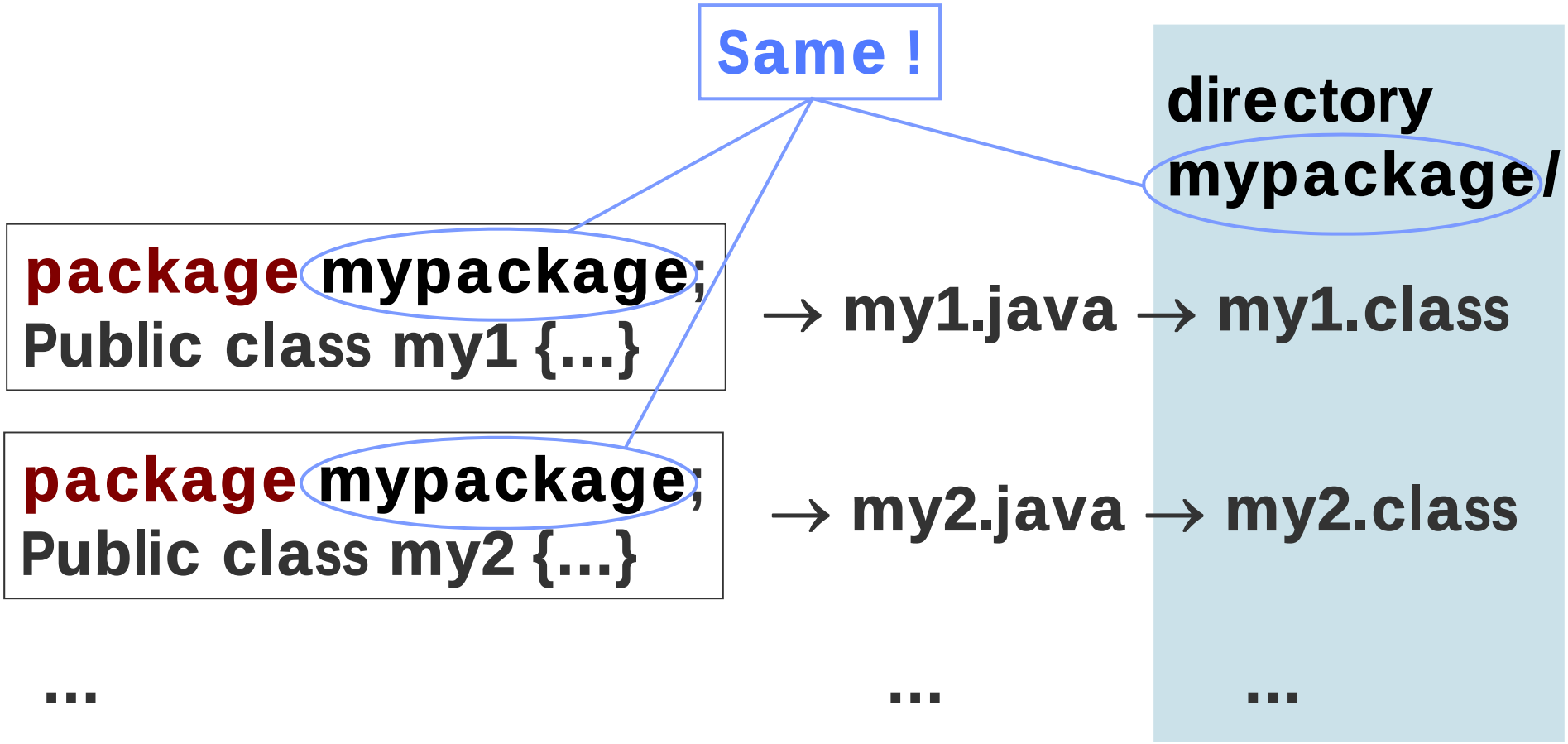
Run: This is a square.
Area of square is 25.0.
...

A **package** is a collection of related classes and interfaces providing access protection and namespace management

Creating a package

Put a **package statement** at the top of the source file in which the class or the interface is defined

Creating a package



Using a package member

- **import** packageName.className ;

  - Import the package member:
    import mypackage.my1;

  - Import the members entire package:
    import mypackage.*

- Refer to the member by its long name
  mypackage.my1
  mypackage.my1 a= new mypackage.my1

Practice 8 Package

Make package:

% mkdir Prac8
% cd Prac8
% mkdir test
% emacs ABC.java → complie → ABC.class

ABC.java

```
// file name:  ABC.java
package test;
public class ABC {
   int x=5;
   public void show()  {
        System.out.println("x= "+x);
   }
}
```

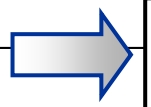
Practice 8 Package

Use package:

% cd Prac8
% emacs Prac8.java → compile → Prac8.class

Prac8.java

```
// file name: Prac8.java    2004. 6 . 17.
import test.*;
class Prac8 {
   public static void main(String args[]) {
      ABC obj=new ABC();
      obj.show();
   }
}
```



Run: x=5

Using existing classes

Many in standard packages:

- **Components, file input/output, mathematical functions, etc. etc.**

- **import the classes defined in those packages (only except java.lang)**

- **Extend Inherit subclass of those existing (superclass) to customise it.**

Java 2 Platform Packages (76)

http://java.sun.com/j2se/1.3/docs/api/overview-summary.html

java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal).
javax.swing	Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

. . . . . .

Classes of java.lang

Math	The class `Math` contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
Number	The abstract class `Number` is the superclass of classes `Byte`, `Double`, `Float`, `Integer`, `Long`, and `Short`.
Object	Class `Object` is the root of the class hierarchy.
Package	`Package` objects contain version information about the implementation and specification of a Java package.
Process	The `Runtime.exec` methods create a native process and return an instance of a subclass of `Process` that can be used to control the process and obtain information about it.
Runtime	Every Java application has a single instance of class `Runtime` that allows the application to interface with the environment in which the application is running.
RuntimePermission	This class is for runtime permissions.
SecurityManager	The security manager is a class that allows applications to implement a security policy.
Short	The Short class is the standard wrapper for short values.
StrictMath	The class `StrictMath` contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
String	The `String` class represents character strings.
StringBuffer	A string buffer implements a mutable sequence of characters.
System	The `System` class contains several useful class fields and methods.
Thread	A *thread* is a thread of execution in a program.

• • • • • •

7. Modifiers

Declaration of Class, Variable, Method

Modi1 Modi2 ... **class** ClassName { ... }
Modi1 Modi2 ... type Variable;
Modi1 Modi2 ... returntype methodName(arg) { ... }

Modifier

Access controlling: public, protected, private, ...
Other modifiers: static, final, abstract, ...

7. Modifiers

Access Control

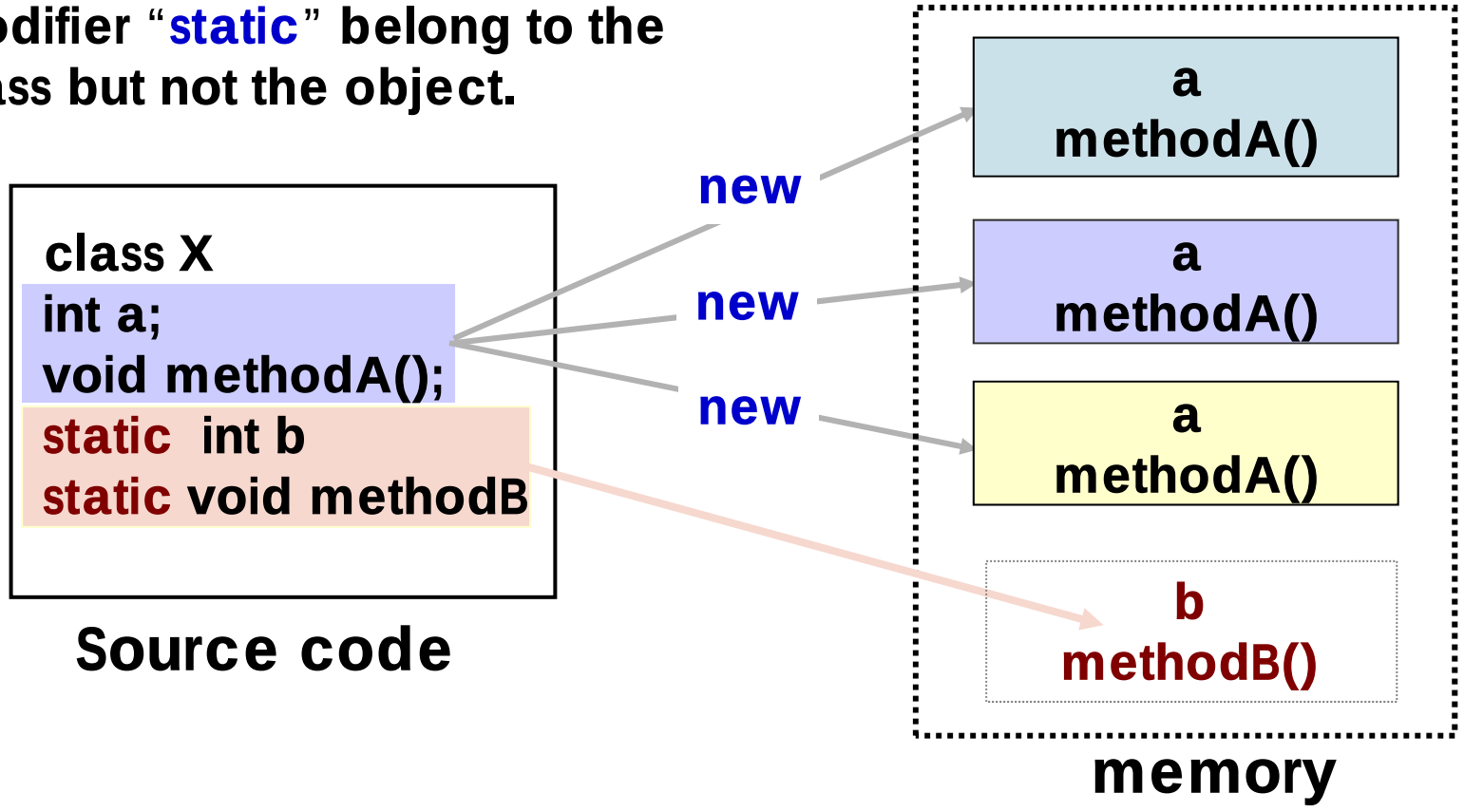
variables, methods, class, package

modifier	Same class	Same package	Subclass	Other classes
public	○	○	○	○
protected	○	○	○	X
default	○	○	X	X
private	○	X	X	X

7. Modifiers

static

Variables or Methods with the modifier "static" belong to the class but not the object.



7. Modifiers

final: final variable: constant
final method: can not be overloaded
can not be overridden
final class: no subclass

abstract: abstract method: no method body
must be overridden
abstract class: can not be instantiated

native: method written in other language,
C, C++, ...

8. Exception

Two type of errors in running

- No way to continue running the program.
- Can be processed in the program.

Exception

An exception is an event that disrupts the normal flow of instruction during the execution of a program.

In Java: Creating an Exception Object and handling it off.

Exception handling

- catch: “try-catch-finally” block
- declaring: [throws]
- defining and throw

Several kinds of Exception

- ArithmeticException
- NullpointerException
- ArrayINdexOutOfBoundsException
- IOException
- ...

Try-catch-finally blocks

```
try { code may have exception
}
catch(ExceptopnType1 ) { action1
}
catch (ExceptopnType1 ) {action2
}
. . . .
finally {final process
}
```

can be abbreviated if unnecessary

Practice 9 Exception

Prac9.java

```
// Practice 9: Exception                    2004. 6. 17.
public class Prac9{
  public static void main (String [ ] args) {
        int a;
 try {
        a=5/0;                    // exception happened!
        System.out.println(a);
    }
    catch(ArithmeticException e) {
    System.out.println("It is a ArithmeticException") ;
    }
  }
}
```

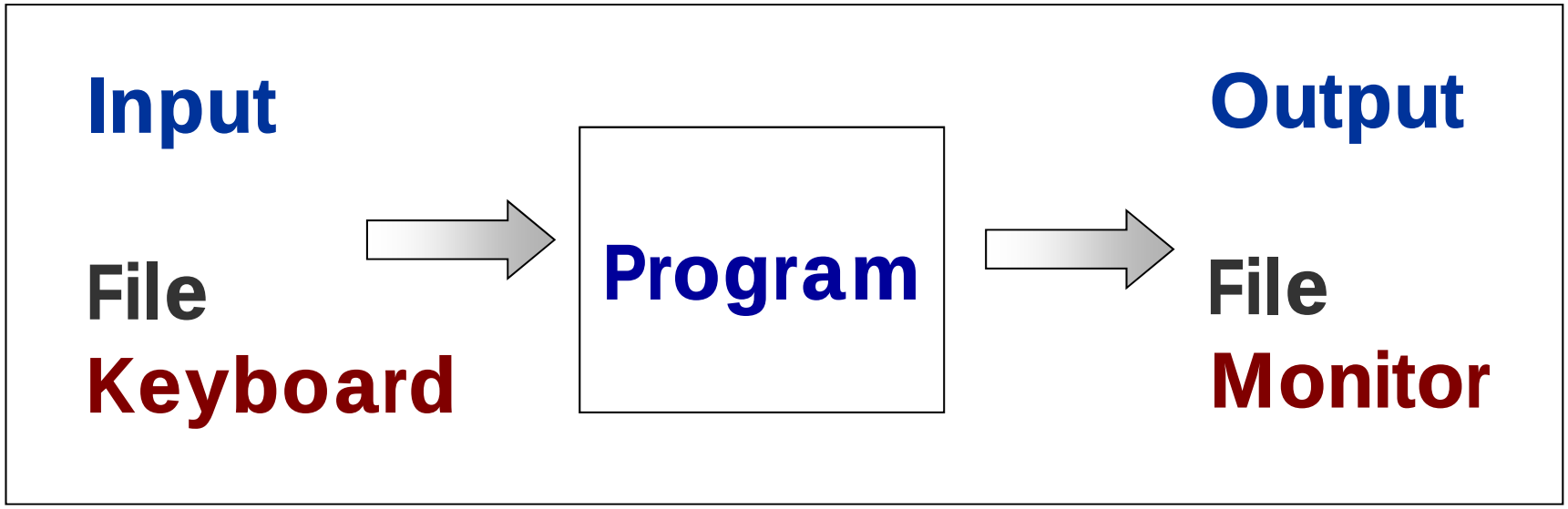
9. Input and Output

System.in.
read()

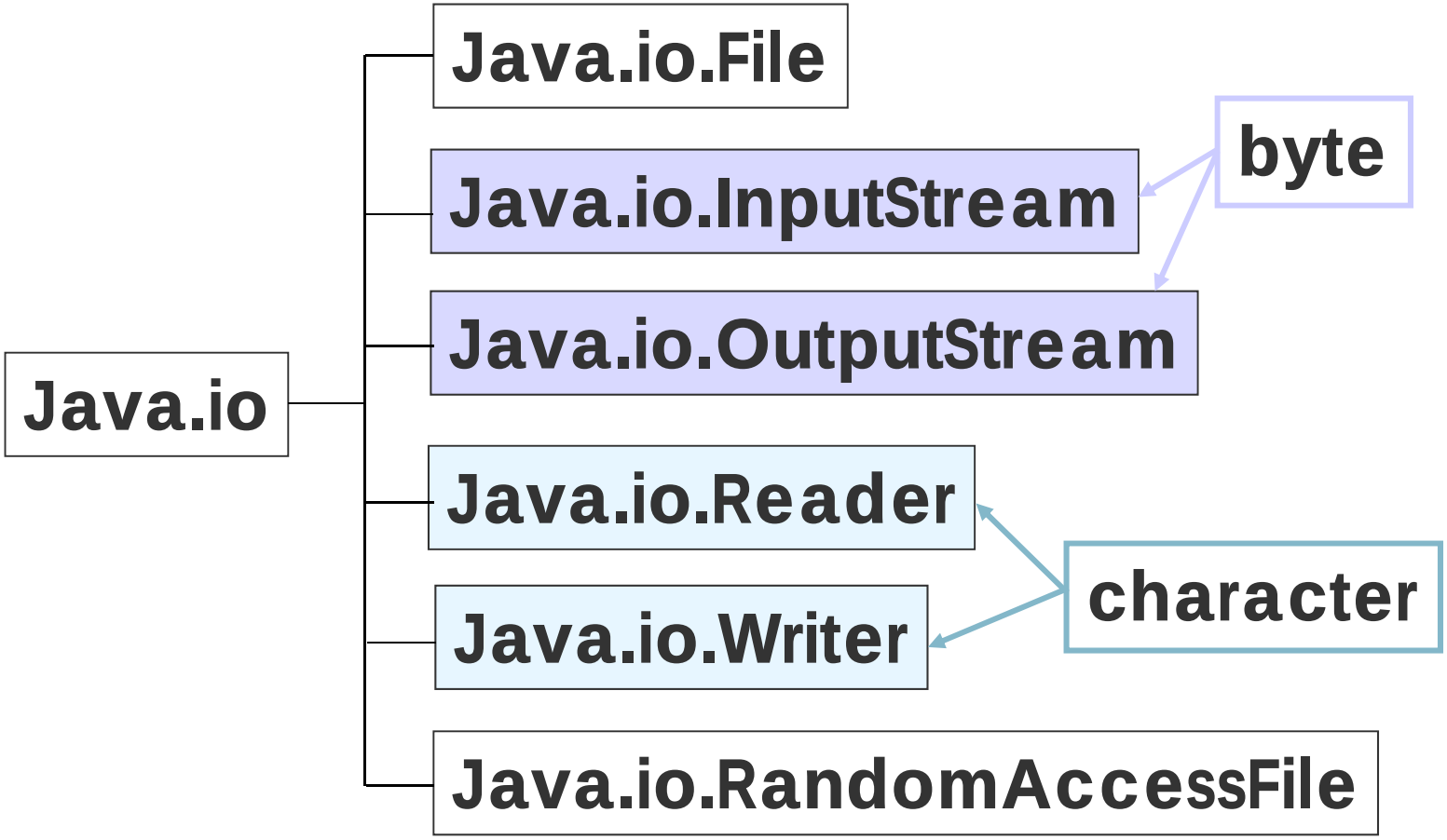
Standard I/O

System.out.
Print()/Prinln()

Stream: flow of data

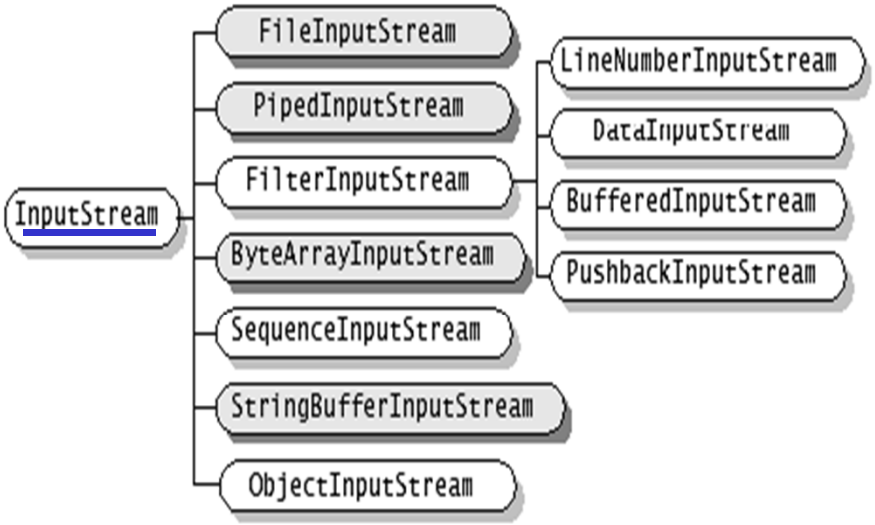


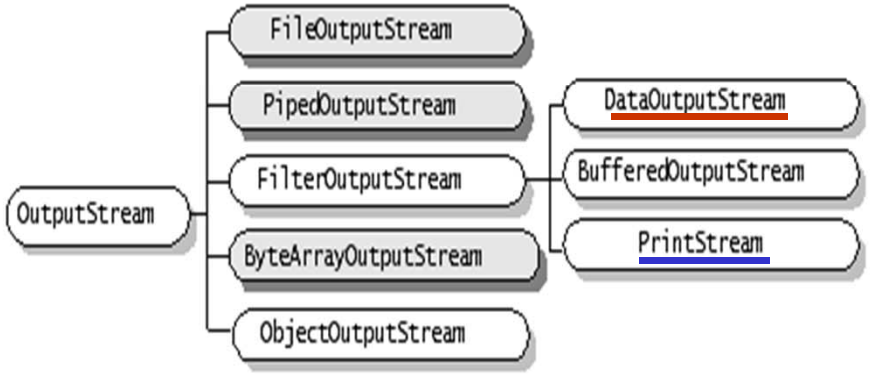
Hierarchy structure of IO classes

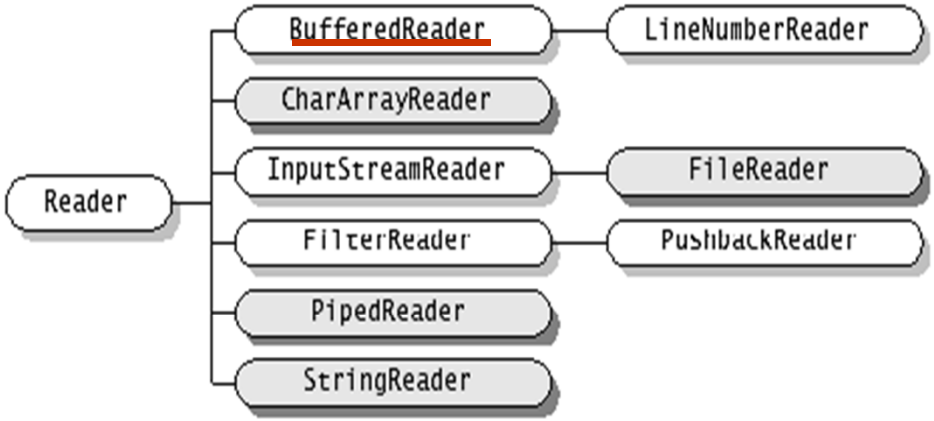


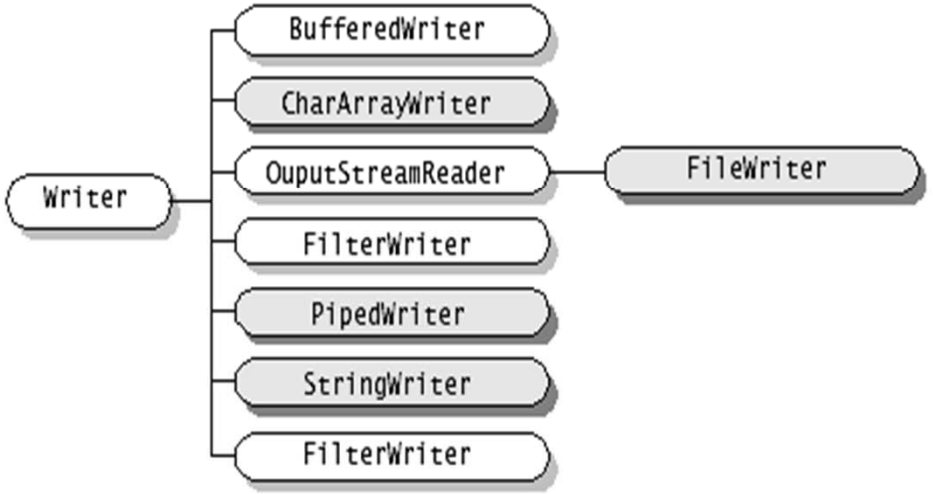
9. Input and Output

Some classes in java.io package





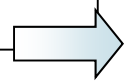




Practice 10 I/O Stream

Prac10.java (1)

```
/*  Practice 10: I/O,                    2004. 6. 17.   */
import java.io.*;
class Prac10{
     public static void main (String [ ] args) {
        System.out.println("Input a floating number: ");
       try {
            BufferedReader in=new BufferedReader
            (new InputStreamReader(System.in));
                String s= in.readLine( );
                double f=Double.valueOf(s).doubleValue();
                double ff=2.0+f;
                System.out.println(" The input string is " +s);
                System.out.println(" The input string+2 is " +s+2.0);
                System.out.println(" The input number is " +ff);
```



Practice 10 I/O Stream

Prac10.java (2)

```
DataOutputStream fo=new DataOutputStream
(new FileOutputStream("data.out"));
fo.writeChar('\n');
fo.writeChars("ff= "+Double.toString(ff));
fo.close();
 } // end try
catch(IOException e) {
    System.out.println (" IOException !" ) ;
 } // end catch
} //
} // e
```

```
tying@as301> java Prac10
Input a floating number:
32.7
The input string is 32.7 (s:string)
The input string+2 is 32.72.0 (s:string)
The input number is 34.7 (f:number→ff: number)
```

Computer Programming

IIE (799-18)

(7) June 24, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

Schedule

- **Former half part**
  - I. Basic of Java programming
- **Later half part**
  - **II. GUI (Graphics and Applets)**
  - **III. Numerical algorithms**

1. Getting Started with Swing (7) 12/18
2. Using Swing components:
   frame, button, label
3. Event handling

Using existing classes

Many in standard packages:

components, file input/output, mathematical functions, etc. etc.

import the classes defined in those packages

Extend Inherit subclass of those existing(superclass) to customise it.

```
class SubclassName extends SuperclassName {...}
```

Graphical User Interface (GUI)

AWT (Abstract Windows Toolkits)

```
import java.awt.*;
```

Swing (Java Foundation Classes Package)

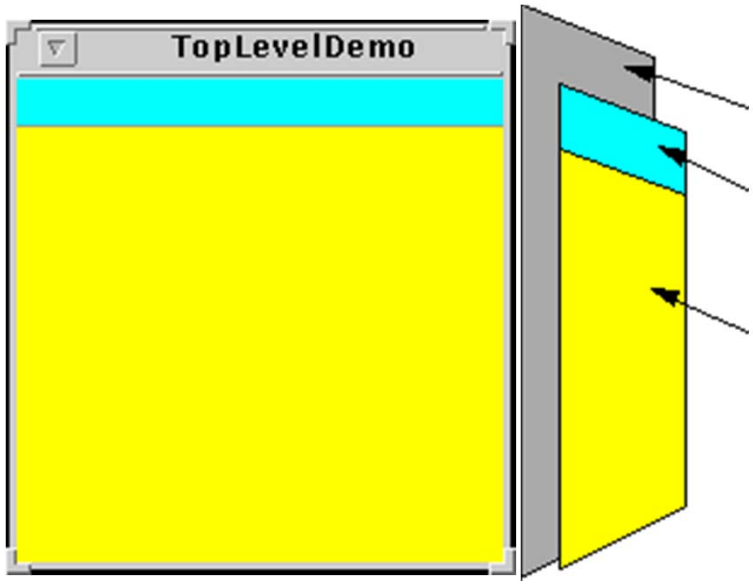
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

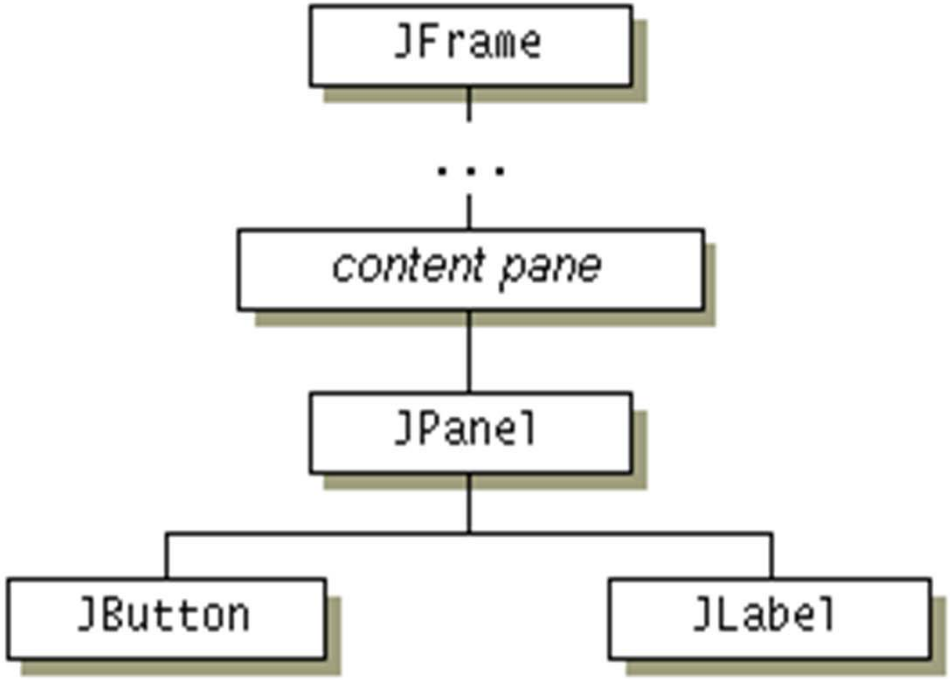
All Swing component: J…

1. Getting Started with Swing

Containment hierarchy of Swing components

Top-level container: JFrame, JDialog, JApplet

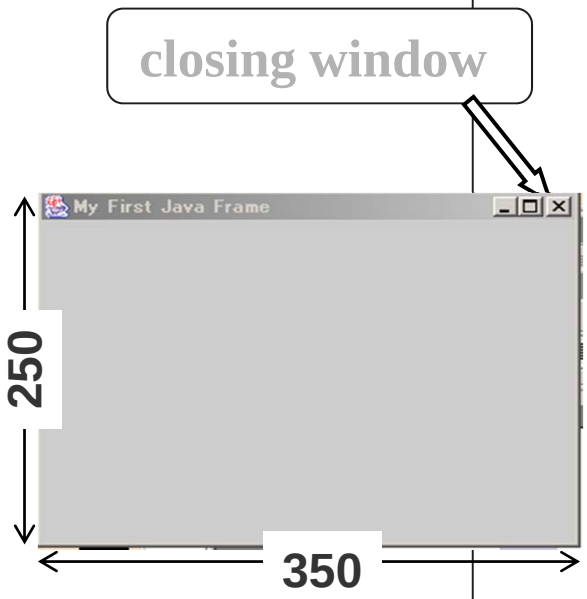




Practice11 Creating a Frame

AppFrame1.java

```
import javax.swing.*;
class Frame1 extends JFrame {
   /* Construction of the frame */
   public Frame1() {
      this.setSize(350, 250);
      this.setTitle("My First Java Frame");
  }
}
public class AppFrame1 {
  /*  Constructor */
    public AppFrame1() {
       Frame1 frame = new Frame1();
       frame.setVisible(true);
    }
  /* Main method */
  public static void main(String[ ] args) {
       new AppFrame1();   // call constructor
  }
}
```



2. Using Swing Components

javax.swing

Class JComponent

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
```

All Implemented Interfaces:

ImageObserver, MenuContainer, Serializable

Direct Known Subclasses:

AbstractButton, BasicInternalFrameTitlePane, Box, Box.Filler, JColorChooser, JComboBox, JFileChooser, JInternalFrame, JInternalFrame.JDesktopIcon, JLabel, JLayeredPane, JList, JMenuBar, JOptionPane, JPanel, JPopupMenu, JProgressBar, JRootPane, JScrollBar, JScrollPane, JSeparator, JSlider, JSpinner, JSplitPane, JTabbedPane, JTable, JTableHeader, JTextComponent, JToolBar, JToolTip, JTree, JViewport

2. Using Swing Components

add, set, get method

- Methods for adding components:

  XXX.add( );

- Methods for getting an object's property:

  objectName.getXXX( );

- Methods for setting an object's property:

  objectName.setXXX( );

2. Using Swing Components

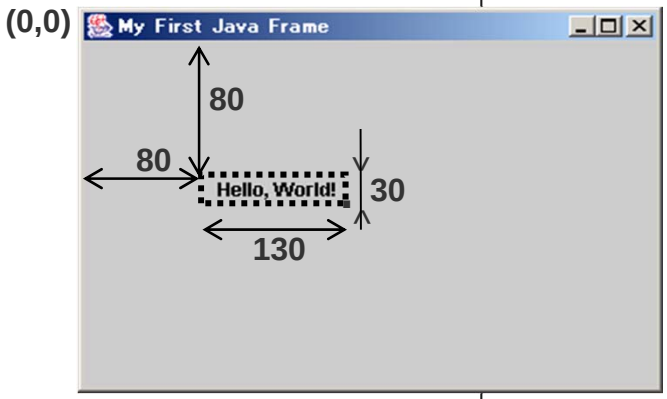
How to add button & label

```
frame = new JFrame(...);
button = new JButton(...);
label = new JLabel(...);
pane = new JPanel();

pane.add(button);
pane.add(label);

frame.getContentPane().add(pane,
BorderLayout.CENTER);
...
```

Practice 12 Frame with adding label

AppFrame2.java (1)

```
import javax.swing.*;
class Frame2 extends JFrame {
  JPanel contentPane;
  JLabel jLabel1 = new JLabel();
   /* Construction of the frame  */
  public Frame2() {
    this.setSize(350, 250);
    this.setTitle("My 2nd Java Frame");
    jLabel1.setText("Hello, World!");
    jLabel1.setBounds(80, 80, 130, 30);
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    contentPane.add(jLabel1, null);
  }
}
```



Practice 12 Frame with adding new part

AppFrame2.java (2)

```
public class AppFrame2 {
/* Constructor */
  public AppFrame2() {
    Frame2 frame = new Frame2();
                    // Quit to shell by clicking "x" button of window
    frame.setDefaultCloseOperation(Frame2.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
/* Main method */
  public static void main(String[ ] args) {
    new AppFrame2();
  }
}
```

... end

3. Events Handling

- **Events:** users' typing a character, pressing a mouse button, move a cursor, etc.
- Any object can be notified of the events.
- You can define what the object should "behave" when notified.

3. Events Handling

Examples of events and events listeners

Act that results in the event	Listener type
User clicks a button, presses Return while typing in a text field, or chooses a menu item	`ActionListener`
User closes a frame (main window)	`WindowListener`
User presses a mouse button while the cursor is over a component	`MouseListener`
User moves the mouse over a component	`MouseMotionListener`
Component becomes visible	`ComponentListener`
Component gets the keyboard focus	`FocusListener`
Table or list selection changes	`ListSelectionListener`

3. Events Handling

Requires three pieces of code

- In the declaration

  public class MyClass implements ActionListener

  (Note: no need if sub-classed from JFrame)

- Registers an instance of the event handler class as a listener

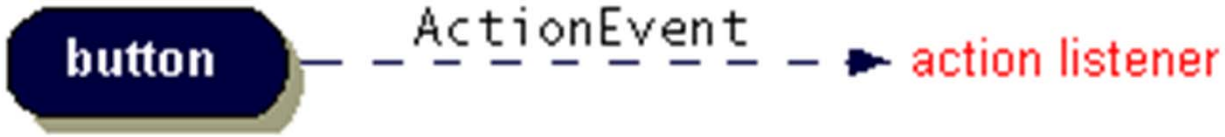
  someComponent.addActionListener(instanceOfMyClass);

- Implements the methods in the listener interface.

  public void actionPerformed(ActionEvent e)
  { ...//code that reacts to the action... }

3. Events Handling

eg. how buttons (JButton) handle mouse clicks

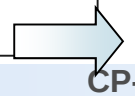


Program: object → register this object as an “action listener”; using addActionListener method;
Event source: clicking a on-screen button;
Button: invocate the action listener's actionPerformed method

Practice 13-1 Event Handling

AppFrame3-1.java (1)

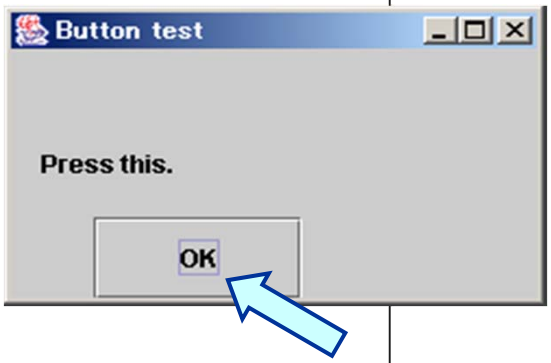
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class Frame3 extends JFrame {
  JPanel contentPane;
  JLabel jLabel1 = new JLabel();
  JButton jButton1 = new JButton();

  /*Construction of a frame*/
  public Frame3() {
    /* Text Label */
    jLabel1.setText("Press this button.");
    jLabel1.setBounds(14, 39, 163, 35);
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.setSize(250,150);
    this.setTitle("Button test");

 /* Setting a button */
    jButton1.setText("OK");
    jButton1.setBounds(38, 83, 96, 41);
    jButton1.addActionListener(new ActionListener() {
```

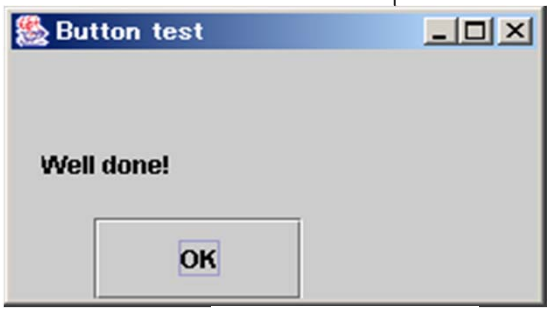


Practice 13-1 Event handling

AppFrame3-1.java (2)

```
  public void actionPerformed(ActionEvent e) {
        jLabel1.setText("Well done!");
      }
    });
    contentPane.add(jButton1, null);
    contentPane.add(jLabel1, null);
  }
}

public class AppFrame3 {

  /*  Constructor */
  public AppFrame3() {
    Frame3 frame = new Frame3();
    frame.setDefaultCloseOperation(Frame3.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }

  /* Main method */
  public static void main(String[ ] args) {
    new AppFrame3();
  }
}
```





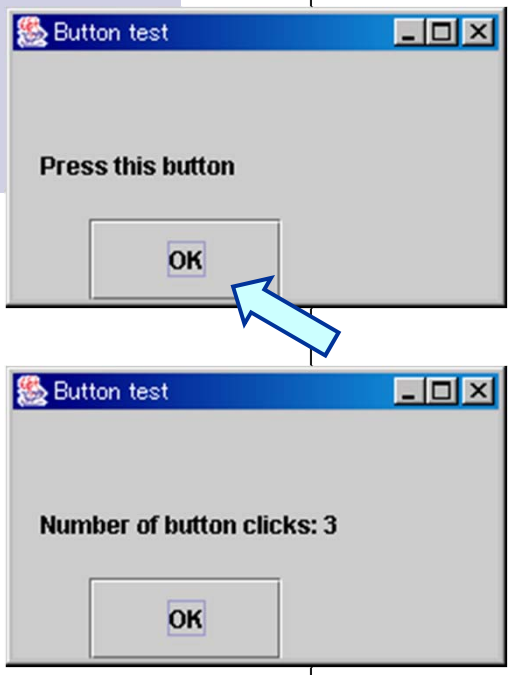
… end

Practice 13-2 Event Handling

Modify AppFrame3-1.java → AppFrame3-2.java

AppFrame3-2.java

```
...
class Frame3 extends JFrame {
   private static String labelPrefix = "Number of button clicks: ";
   private int numClicks = 0;
       JPanel contentPane;
       JLabel jLabel1 = new JLabel(labelPrefix + "0    ");
       JButton jButton1 = new JButton();
...

public void actionPerformed(ActionEvent e) {
    numClicks++;
    jLabel1.setText(labelPrefix + numClicks);
}

...
```



More details on Swing ...

A Visual Index to the Swing Components

http://java.sun.com/docs/books/tutorial/uiswing/components/components.html

The JComponent Class Properties

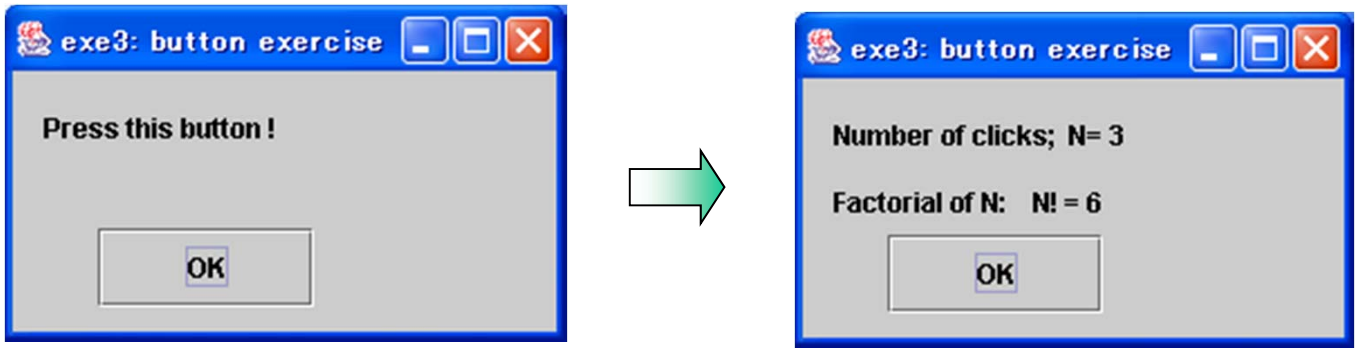
http://java.sun.com/docs/books/tutorial/uiswing/components/jcomponent.html#properties

Exercise 3 (submit by July 7, 2004)

Based on Practice13-2, write a program,

1) To display a frame as show in (a);

2) Clicking the button, to show the number of clicks N and its factorial N! = 1x2 x … xN ,as shown in (b).



Computer Programming

IIE (799-18)

(8) July 1, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

GUI and Applet / Numerical Algorithm

1. Getting Started with swing (7) 6/24
2. Using Swing components:
   frame, button, label
3. Event handling

Mathematical functions

4. More on Swing components:
   input-box, plotting **Today** (8) 7/1

...

Mathematical functions

http://java.sun.com/j2se/1.3/docs/api/java/lang/Math.html

java.lang
Class Math

java.lang.Object
  |
  +--java.lang.Math

public final class Math extends Object

Method Summary	
static double	**abs**(double a) Returns the absolute value of a double value.
static double	**acos**(double a) Returns the arc cosine of an angle, in the range of 0.0 through *pi*.
static double	**asin**(double a) Returns the arc sine of an angle, in the range of −*pi*/2 through *pi*/2.
static double	**atan**(double a) Returns the arc tangent of an angle, in the range of −*pi*/2 through *pi*/2.
static double	**atan2**(double a, double b) Converts rectangular coordinates (b, a) to polar (r, *theta*).
static double	**ceil**(double a) Returns the smallest (closest to negative infinity) double value that is not less than the argument and is equal to a mathematical integer.
static double	**cos**(double a) Returns the trigonometric cosine of an angle.
static double	**exp**(double a) Returns the exponential number *e* (i.e., 2.718...) raised to the power of a double value.
static double	**sin**(double a) Returns the trigonometric sine of an angle.
static double	**sqrt**(double a) Returns the correctly rounded positive square root of a double value.
static double	**tan**(double a) Returns the trigonometric tangent of an angle.

...

Mathematical functions

double E=2.71…,
double PI= 3.14159…

int abs(int a), long abs(long a)
double sin(double a), double cos(double a)
double exp(double a), double log(double a),
…

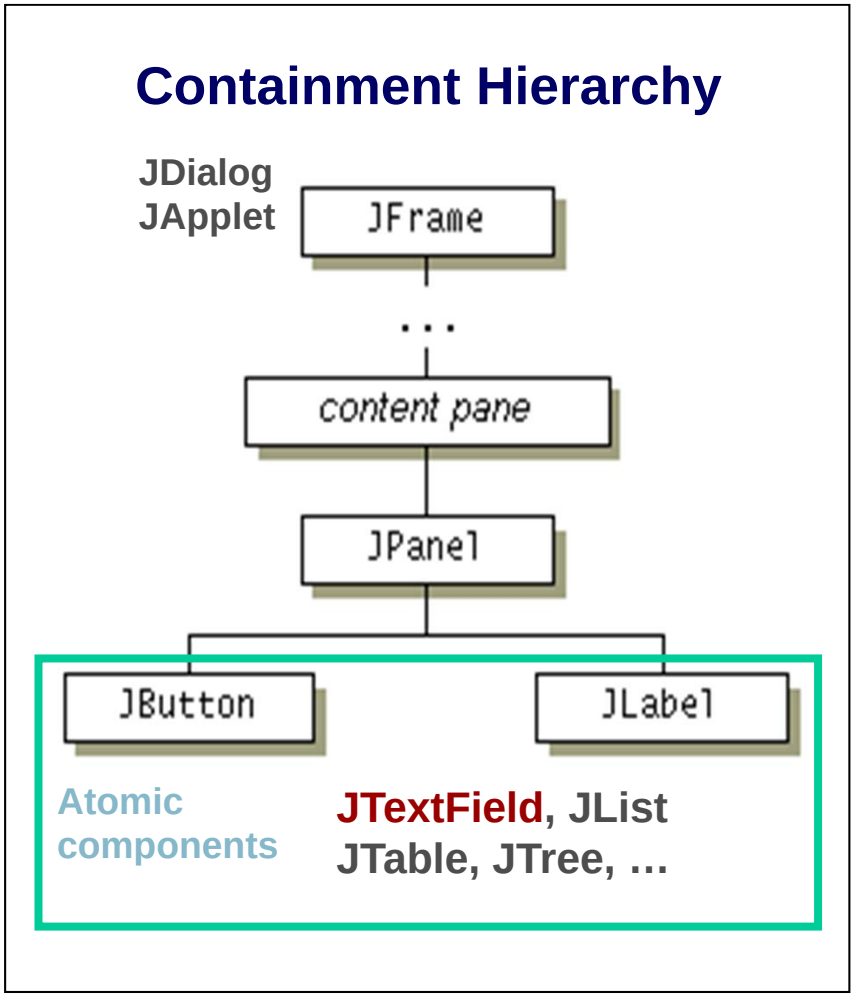
Usage

Math.sin(x), Math.cos(x), …

GUI: Adding components

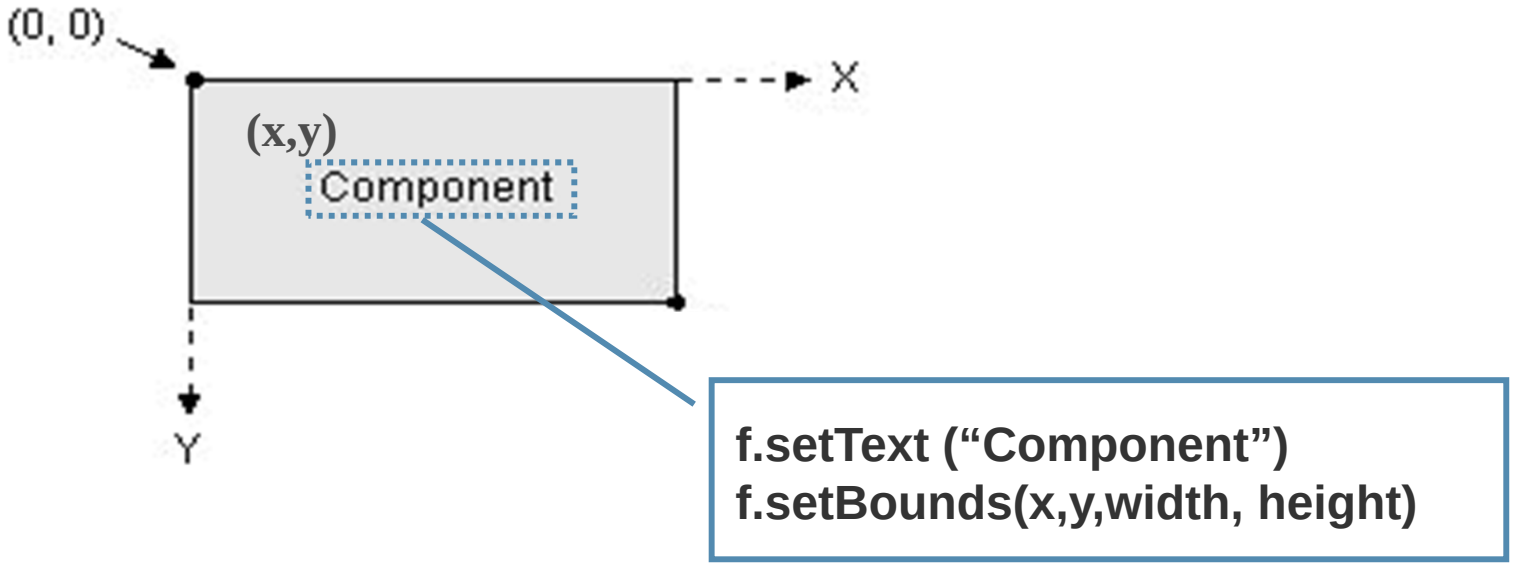
```
frame = new JFrame(…);
button = new JButton(...);
label = new JLabel(...);
textfield = new JTextField(...);
pane = new JPanel();
pane.add(button);
pane.add(label);
frame.getContentPane().add(
pane,BorderLayout.CENTER)
;
```

http://java.sun.com/docs/books/tutorial/uiswing/components/components.html



The coordinate system

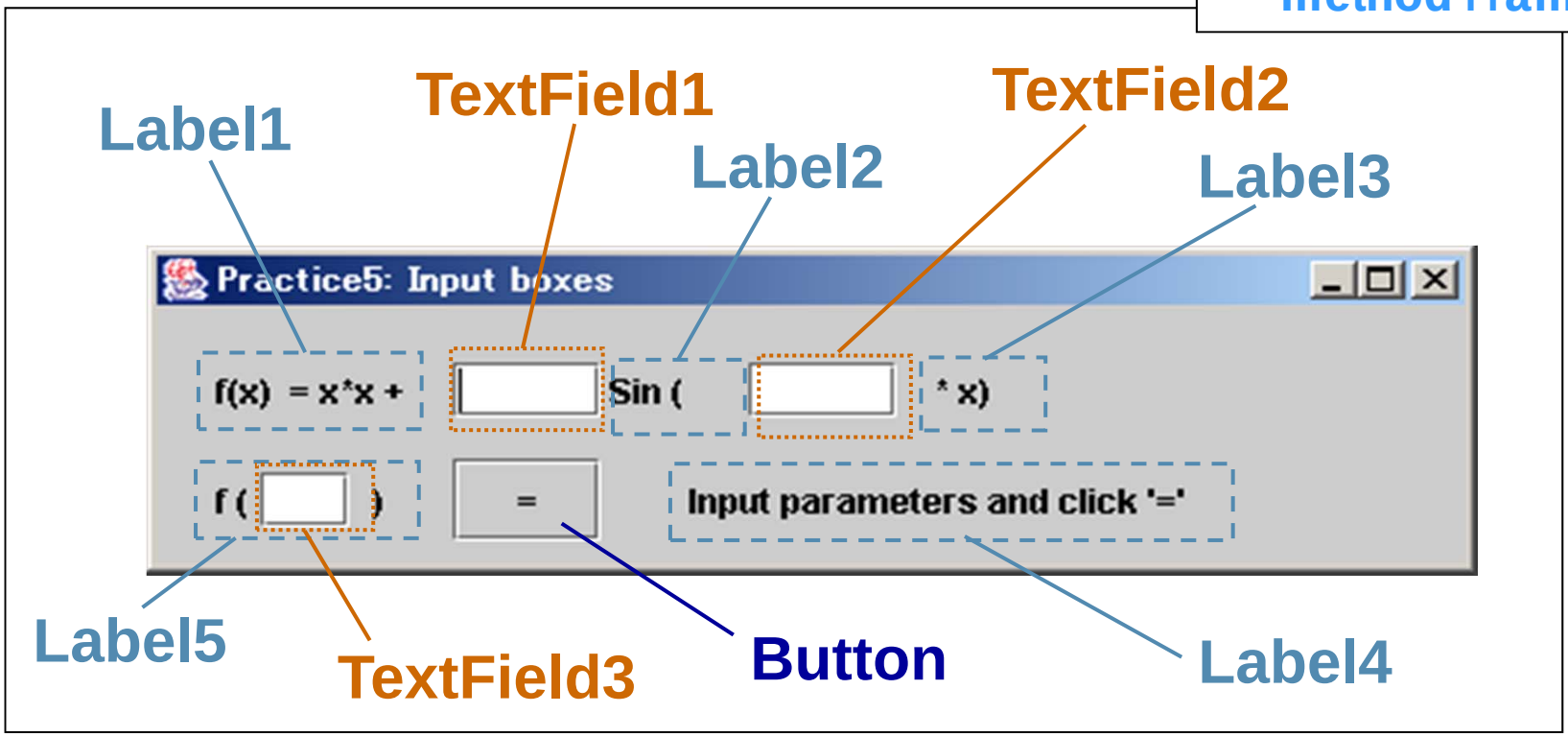
Each component has its own integer coordinate system.
(0,0): upper left corner of a component's painting area.
X: increases to the right Y: increases downward



Practice 14: "Function Calculator"

Designing a GUI to calculate value of a function with keyboard input parameters: $f(x) = x^2 + aSin(bx)$

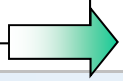
Program:
- **Main class: AppFrame4**
  main method
- **extend JFrame -> Frame4:**
  method Frame4



Practice 14: “Function Calculator”

AppFrame4.java (1)

```
// Pracetice 14: input box       July. 1, 2004
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AppFrame4{
    public AppFrame4(){
        Frame4 frame = new Frame4();
       // Quit to command line when clicking “x” button of window
        frame.setDefaultCloseOperation(Frame4.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    public static void main (String[]args){
        new AppFrame4();
    }
}
```



Practice 14: “Function Calculator”

AppFrame4.java (2)

```
class Frame4 extends JFrame {
   /* Declare components in Panel */
   JPanel contentPane;
   JLabel jLabel1 = new JLabel();
   JLabel jLabel2 = new JLabel();
   JLabel jLabel3 = new JLabel();
   JLabel jLabel4 = new JLabel();
   JLabel jLabel5 = new JLabel();
   JTextField jTextField1 = new JTextField();
   JTextField jTextField2 = new JTextField();
   JTextField jTextField3 = new JTextField();
   JButton jButton1 = new JButton();
```

Practice 14: “Function Calculator”

AppFrame4.java (3)

```
/* Construction of the frame */
    public Frame4 (){
        jLabel1.setText("f(x)  = x*x + ");
        jLabel1.setBounds(20,20,100,20);
        jLabel2.setText(" Sin ( ");
        jLabel2.setBounds(150,20,50,20);
        jLabel3.setText(" * x)");
        jLabel3.setBounds(260,20,50,20);
        jLabel4.setText("Input parameters and click '='");
        jLabel4.setBounds(180,60,220,20);
        jLabel5.setText("f (              )");
        jLabel5.setBounds(20,60,130,20);
        jTextField1.setBounds(100,20,50,20);
        jTextField2.setBounds(200,20,50,20);
        jTextField3.setBounds(35,60,30,20);
        contentPane = (JPanel)this.getContentPane();
        contentPane.setLayout(null);
        this.setSize(450,120);  //size of frame
        this.setTitle("Practice14: Input boxes");
```

Practice 14: “Function Calculator”

AppFrame4.java (4)

```
/* Setting a button */
        jButton1.setText("=");
        jButton1.setBounds(100,55,50,30);
        jButton1.addActionListener(new ActionListener (){
            public void actionPerformed(ActionEvent e){
                double a = Double.parseDouble(jTextField1.getText());
                double b = Double.parseDouble(jTextField2.getText());
                double c = Double.parseDouble(jTextField3.getText());
                double fx = c*c+a*Math.sin(b*c) ;
                String longResult = new Double(fx).toString();
                String finalResult = longResult.substring(0,6);
                jLabel4.setText(finalResult);
            }
        });
```

Practice 14: “Function Calculator”

AppFrame4.java (5)

```
/* Adding components to pane */
        contentPane.add(jLabel1,null);
        contentPane.add(jLabel2,null);
        contentPane.add(jLabel3,null);
        contentPane.add(jLabel4,null);
        contentPane.add(jLabel5,null);
        contentPane.add(jTextField1,null);
        contentPane.add(jTextField2,null);
        contentPane.add(jTextField3,null);
        contentPane.add(jButton1,null);
    }         // end of method Frame4
}         // end of class Frame4
```

… end

Painting

- Create a panel to put an object in:
  private class MyPanel extends JPanel {….

- Use swing painting method: paintComponent
  e.g. public void paintComponent (Graphics g) {
  super.paintComponent (g)
  g.setColor(new Color(255, 0, 128));
  g.fillOval(0, 0, 10, 20);
  .. ..
  }
- Add MyPanel to the content pane of the main frame.

Graphic Methods

Methods of Graphic class for painting

Lines: **drawLine**(*int x1, int y1, int x2, int y2*)

Rectangles: **drawRect** (*int x, int y, int w, int h*)

fillRect (*int x, int y, int w, int h*)

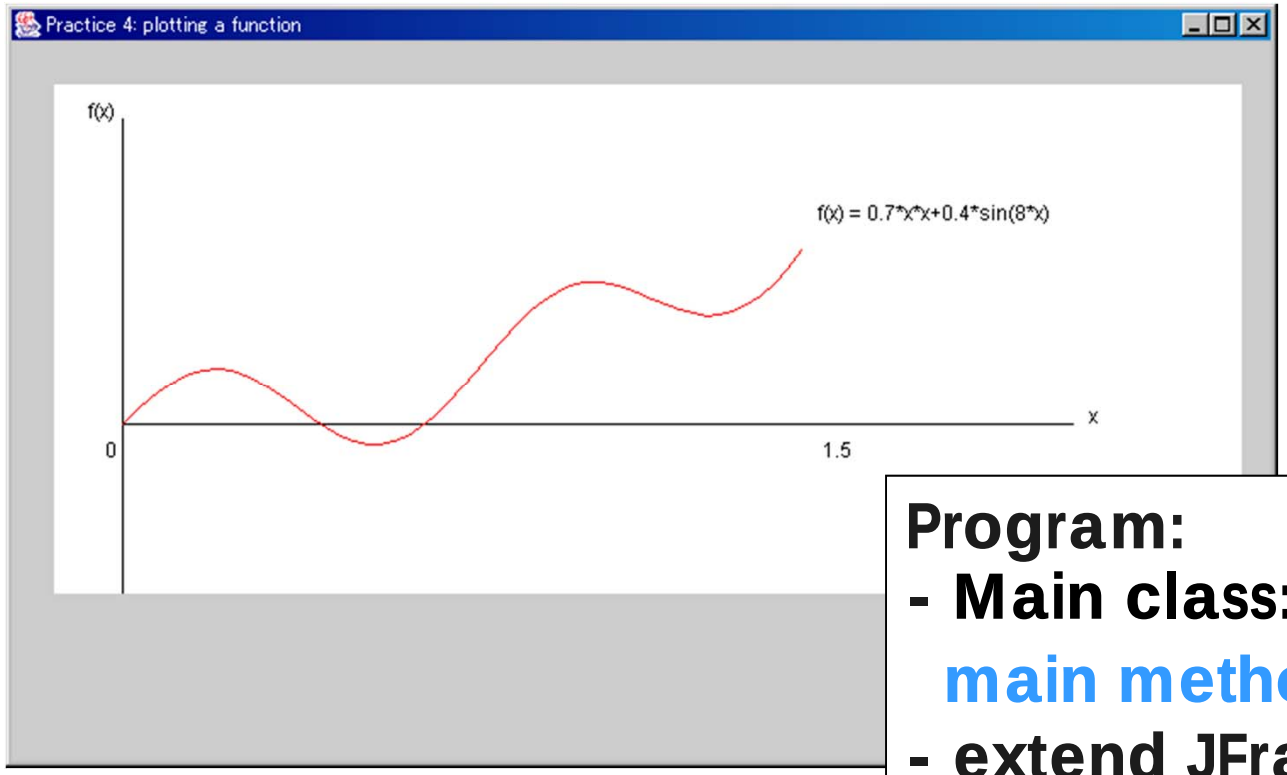
Ovals: **drawOval**, **fillOval**

Arcs: **drawArc**, **fillArc**

Polygons: **drawPolygon**, **drawPolyline**, **fillPolygon**

...

Practice 15: Plotting a function



$$f(x) = cx^2 + aSin(bx)$$

$$a=0.4,\ b=8.0,\ c=0.7$$

Program:

- **Main class: AppFrame5**
  - **main method**
- **extend JFrame -> MyFrame:**
  - **method MyFrame**
  - **JPanel -> MyPanel:**
    - **override paintComponent**
- **FuncVal class: calculate**

Practice 15: Plotting a function

AppFrame5.java (1)

```
// Practice 15: plotting function     July 1, 2004
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AppFrame5{
    public AppFrame5(){
        MyFrame frame = new MyFrame();
        // Quit to command line when clicking “x” button of window
        frame.setDefaultCloseOperation(MyFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    public static void main (String[]args){
        new AppFrame5();
    }
}
```

Practice 15: Plotting a function

AppFrame5.java (2)

```
class MyFrame extends JFrame {
    /* Declare components in Frame */
    JPanel contentPane;
    public MyPanel myPanel1 = new MyPanel ();
    public MyFrame() {
        contentPane = (JPanel)this.getContentPane();
        contentPane.setLayout(null);
        this.setSize(750,450);    //size of frame
        this.setTitle("Practice 15: plotting a function");
        contentPane.add(myPanel1,null);
            double upper=1.5;    // start to calculate points
            double lower=0;
            int numinterval=50;
```

Practice 15: Plotting a function

AppFrame5.java (3)

```
FuncVal funcval1 = new FuncVal ();
    double resultFunc [ ] = funcval1.calculate(upper,lower,numinterval);
    double h = (upper-lower)/numinterval;
          myPanel1.x1 = lower;
          myPanel1.x2 = upper;
          myPanel1.n = numinterval;
          myPanel1.h = h;
          myPanel1.resultFunc = resultFunc;
          myPanel1.setBackground(Color.yellow);
          myPanel1.setBounds(new Rectangle (25,25,700,300));
                  //size of panel (canvas)
  }
```

Practice 15: Plotting a function

AppFrame5.java (4)

```
private class MyPanel extends JPanel {
        public double x1;
        public double x2;
        public double h;
        public int n;
        public double [ ] resultFunc;
        public int [ ] xPoints = new int [2];
        public int [ ] yPoints = new int [2];
        public void paintComponent(Graphics g) {
           super.paintComponent (g);  //painting background
           g.setColor(Color.black);
           g.drawLine(40,20,40,280);      // Draw Y axis
           g.drawLine(40,200,600,200);  // X axis
           g.drawString("f(x)",20,20);     // lable of “X”
           g.drawString("x",610,200);     // lable of “Y”
           g.drawString("0",30,220);       // origin
           g.drawString(" "+x2,450,220);   // mark “x2”
           g.drawString("f(x) = 0.7*x*x+0.4*sin(8*x)",450,80);
```

Practice 15: Plotting a function

AppFrame5.java (5)

```
for (int i=0;i<n;i++) {
                // Notice: convert of plot positions and function values
            xPoints [0] = (int)(40+(x1+(i*h*400/(x2-x1))));
            xPoints [1] = (int)(40+(x1+(i+1)*h*400/(x2-x1)));
            yPoints [1] = (int)(200-resultFunc[i+1]*150/2);
            yPoints [0] = (int)(200-resultFunc[i]*150/2);
            g.setColor(Color.red);
            g.drawLine(xPoints [0],yPoints [0],xPoints [1],yPoints [1]);
          }
       }   // paint method ends here
   }   // MyPanel calss ends here
} // MyFrame class ends here
```

Practice 15: Plotting a function

AppFrame5.java (6)

```
class FuncVal {
    /* list of variables
    x2 : upper limit
    x1 : lower limit
    n  : number of interval
    h  : width of interval (x2-x1)/n */
    double h;
    public double [ ] calculate (double x2, double x1, int n) {
        double result [ ] = new double [n+1];
        h = (x2-x1)/n;
        for (int i=0; i<n+1; i++) {
          result [i] = 0.7*(x1+(i*h))*(x1+(i*h))+0.4*Math.sin(8.0*(x1+(i*h)));
        }
        return result;
    }
}
```

… end

Exercise 4 (submit by July. 14, 2004)

Based on Practice 14 and Practice 15,

1) Create a GUI to calculate the value of

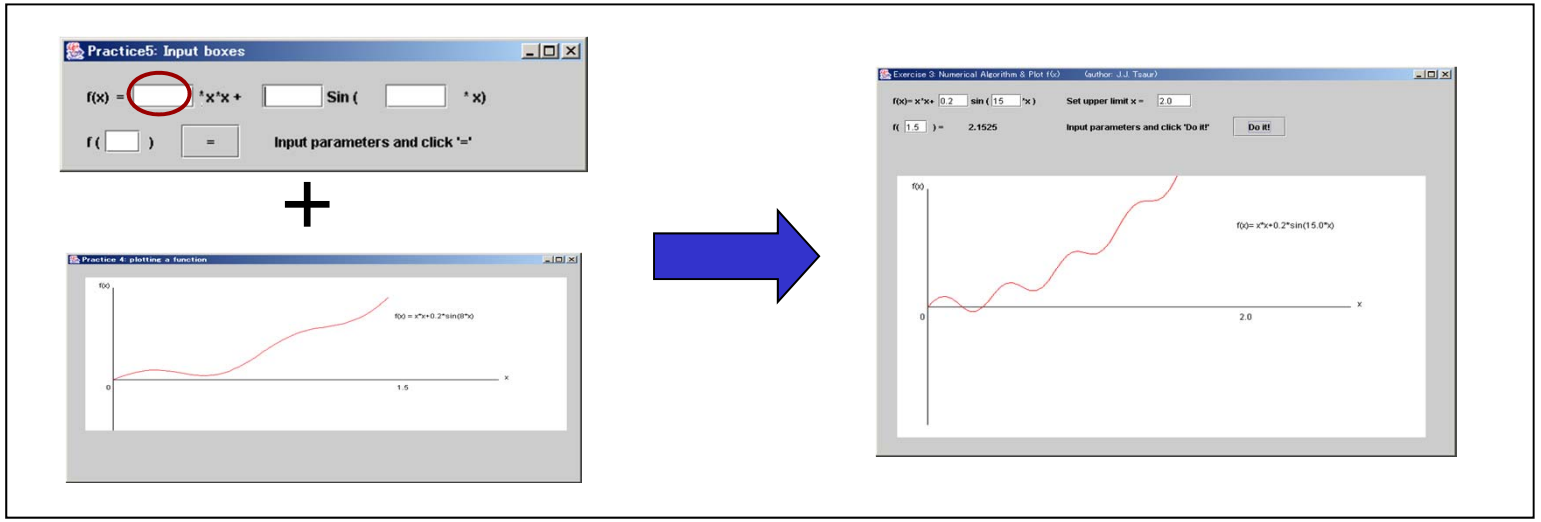
$$f(x) = cx^2 + aSin(bx)$$

by keyboarding *a*, *b*, *c*, and variable *x*, show result;

2) Display the curve of function in same interface.

3) Re-calculate *f(x)* & Redraw the curve of *f(x)*

by inputting new parameters (Use "myobject.updateUI();")



Computer Programming

IIE (799-18)

(9) July 8, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

GUI and Applet / Numerical Algorithm

1. Getting Started with swing (7) 6/24
2. Using Swing components:
   frame, button, label
3. Event handling

Mathematical functions (8) 7/1

4. More on Swing components:
   input-box, plotting
5. More on Swing components: table

Today

Basic Idea, Error

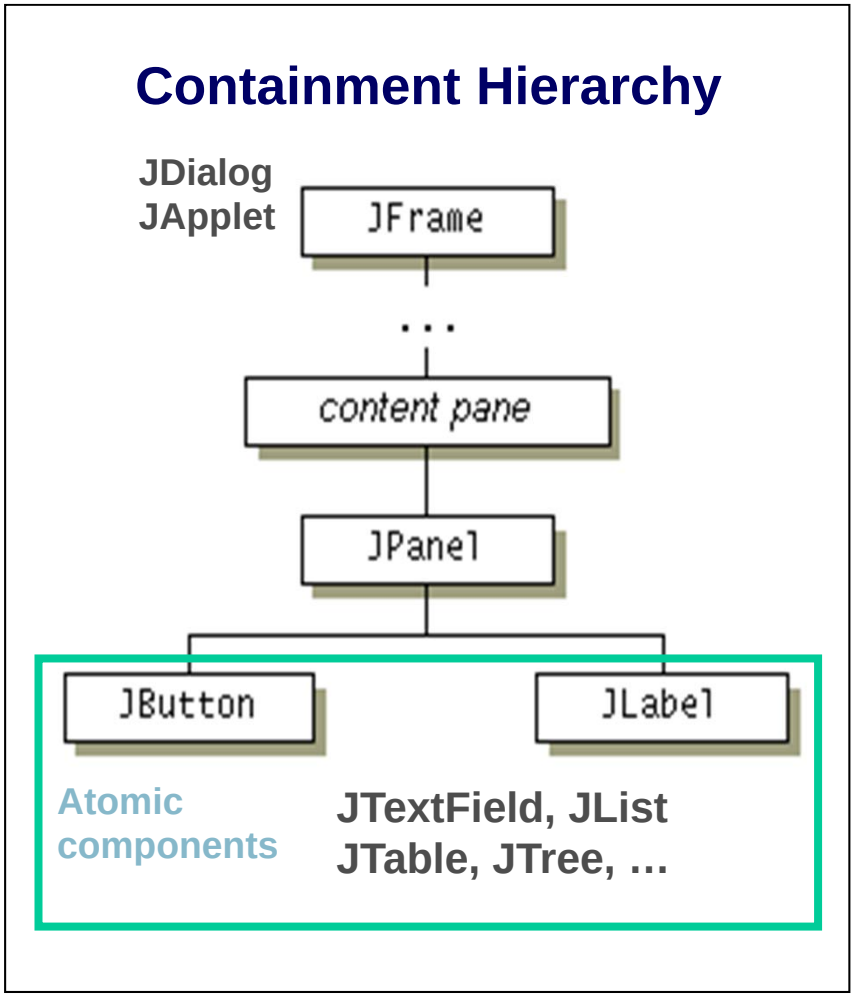
Root finding, Integration

Final Assignment 1 / Questionnaire (9) 7/8

...

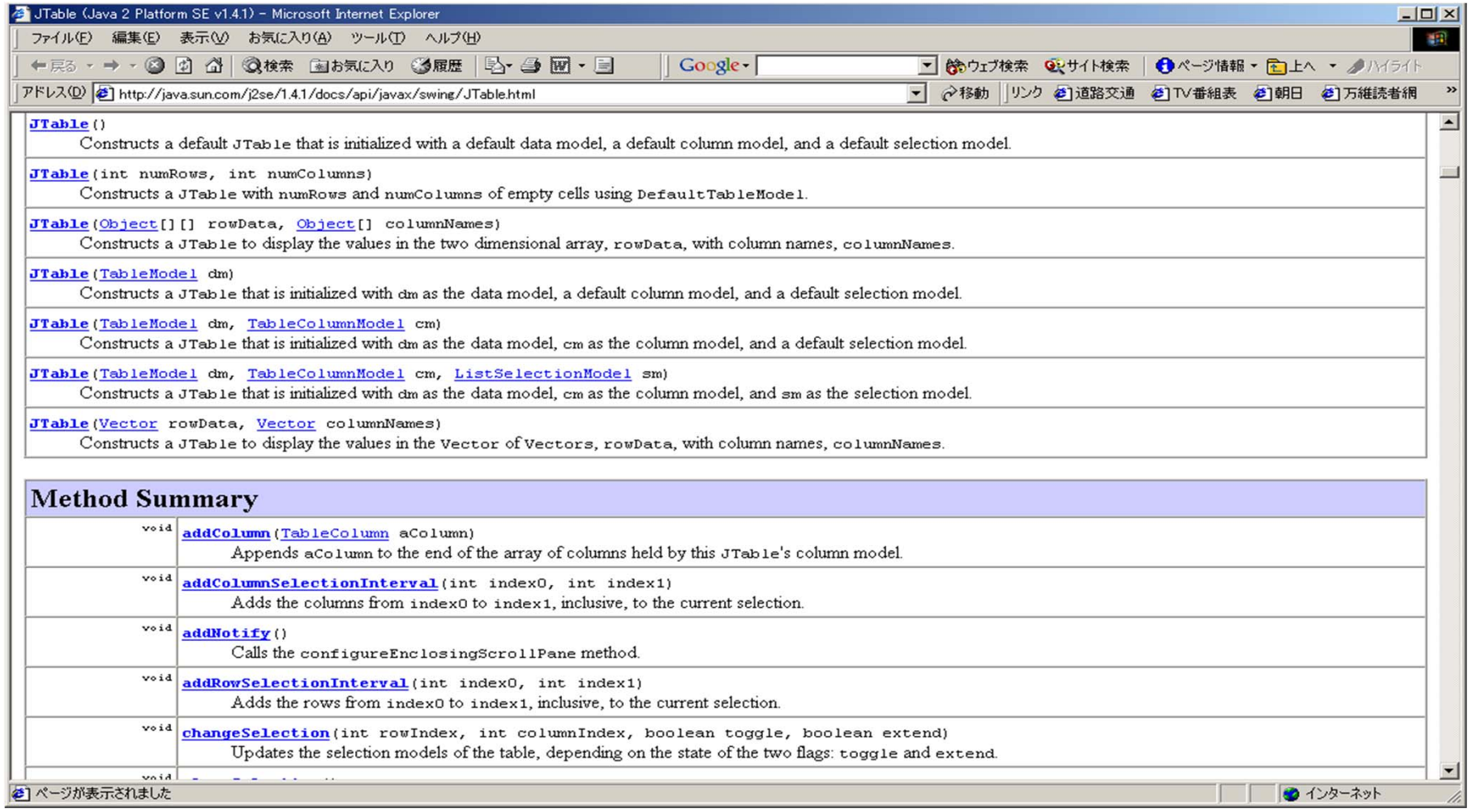
JTable: to display / edit 2D-tables of cells

```
frame = new JFrame(...);

table = new Jtable(m,n);

contentPane.add(jTable1,
null)

table.TableModel model
= table.getModel();

getContentPane().add(sc
rollPane,BorderLayout.CE
NTER);

...
```



javax.swing.JTable classes

http://java.sun.com/j2se/1.4.1/docs/api/javax/swing/JTable.html



See an example ...

Some Properties

cellSelectionEnabled

rowSeletionAllowed

columnSelectionAllowed

autoResizeMode

Manipulating table data (all values are assumed to be of String)

getValueAt
(rowIndex, columnIndex)

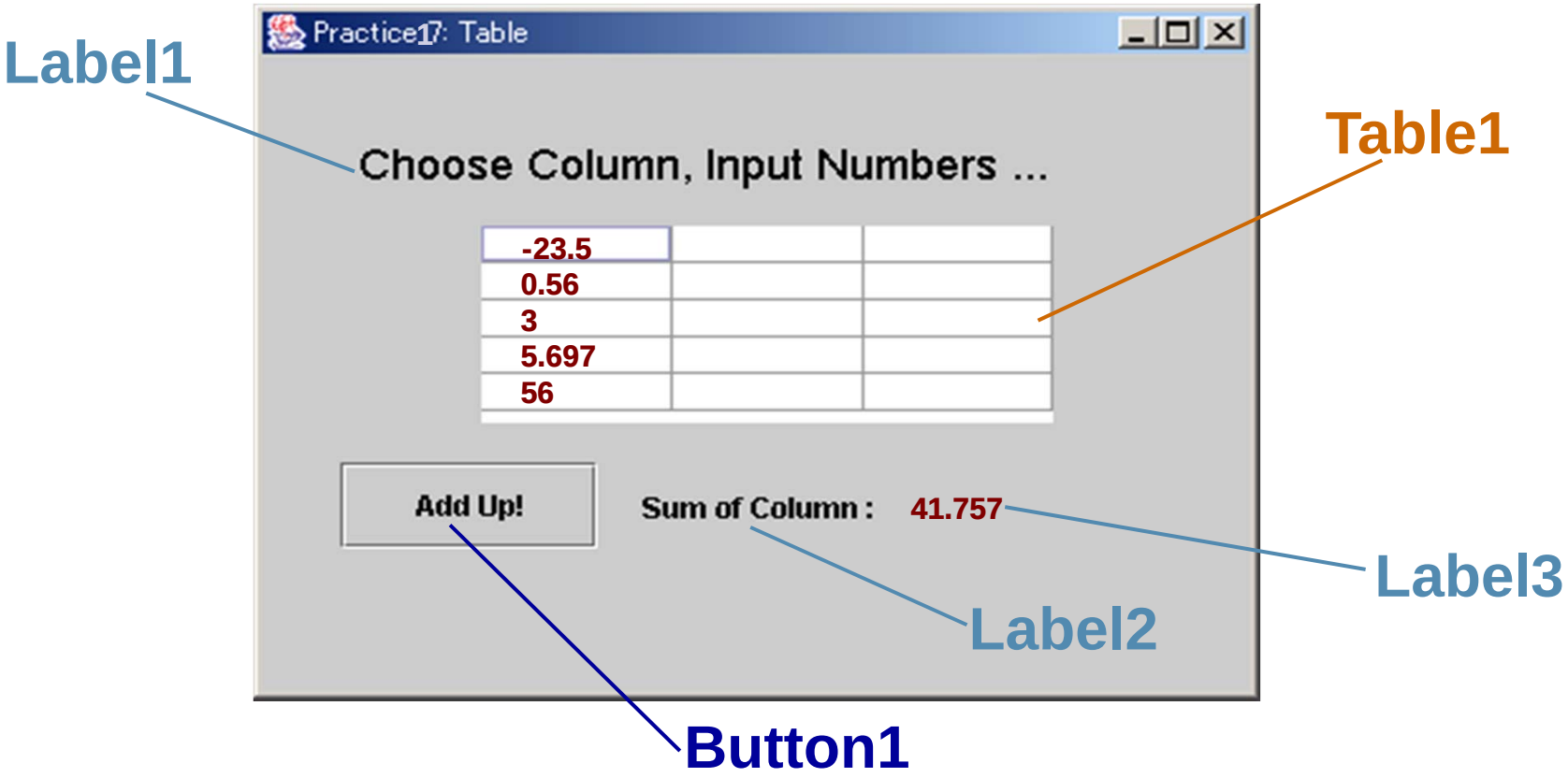
setValueAt
(rowIndex, columnIndex)

getSelectedRows()

getRowCount()

Practice 16: A simple table

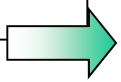
A (5x3) table: Calculate the sum of a column



Practice 16: A simple table

Prac16TableApp.java (1)

```
// Practice 16, Table,              July  8, 2004
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Prac16TableApp {
// Constructor
  public Prac16TableApp() {
    TableFrame frame = new TableFrame();
    frame.setDefaultCloseOperation(TableFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
// Main method
  public static void main(String[] args) {
    new Prac16TableApp();
  }
}
```



Practice 16: A simple table

Prac16TableApp.java (2)

```
class TableFrame extends JFrame {
  JPanel contentPane;
  JTable jTable1 = new JTable(5,3);
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JButton jButton1 = new JButton();
  JLabel jLabel3 = new JLabel();
//  Construciton of the frame
   public TableFrame() {
    try {
      jbInit();          // components initialization
    }
    catch(Exception e) {     //catch error in initialization
      e.printStackTrace();    //print out error message
    }
  }                  // end of method of TableFrame
```

Practice 16: A simple table

Prac16TableApp.java (3)

```
//  Initialisation of the component
  private void jbInit() throws Exception  {   //override jbInit
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(null);
    this.setSize(400, 300);
    this.setTitle("Practice 16: Table");
    jTable1.setColumnSelectionAllowed(true);
    jTable1.setRowSelectionAllowed(false);
    jTable1.setBounds(87, 73, 227, 85);
    jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
    jLabel1.setText("Choose Column, Input Numbers ...");
    jLabel1.setBounds(25, 25, 300, 43);
    jLabel2.setText("Sum of Column : ");
    jLabel2.setBounds(161, 181, 109, 26);
```

Practice 16: A simple table

Prac16TableApp.java (4)

```
jButton1.setText("Add Up!");
    jButton1.setBounds(31, 175, 102, 37);
    jButton1.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
    });
    jLabel3.setBounds(278, 181, 76, 30);
    contentPane.add(jLabel1, null);
    contentPane.add(jTable1, null);
    contentPane.add(jLabel2, null);
    contentPane.add(jLabel3, null);
    contentPane.add(jButton1, null);
  }  // end of jbInit
```

Practice 16: A simple table

Prac16TableApp.java (5)

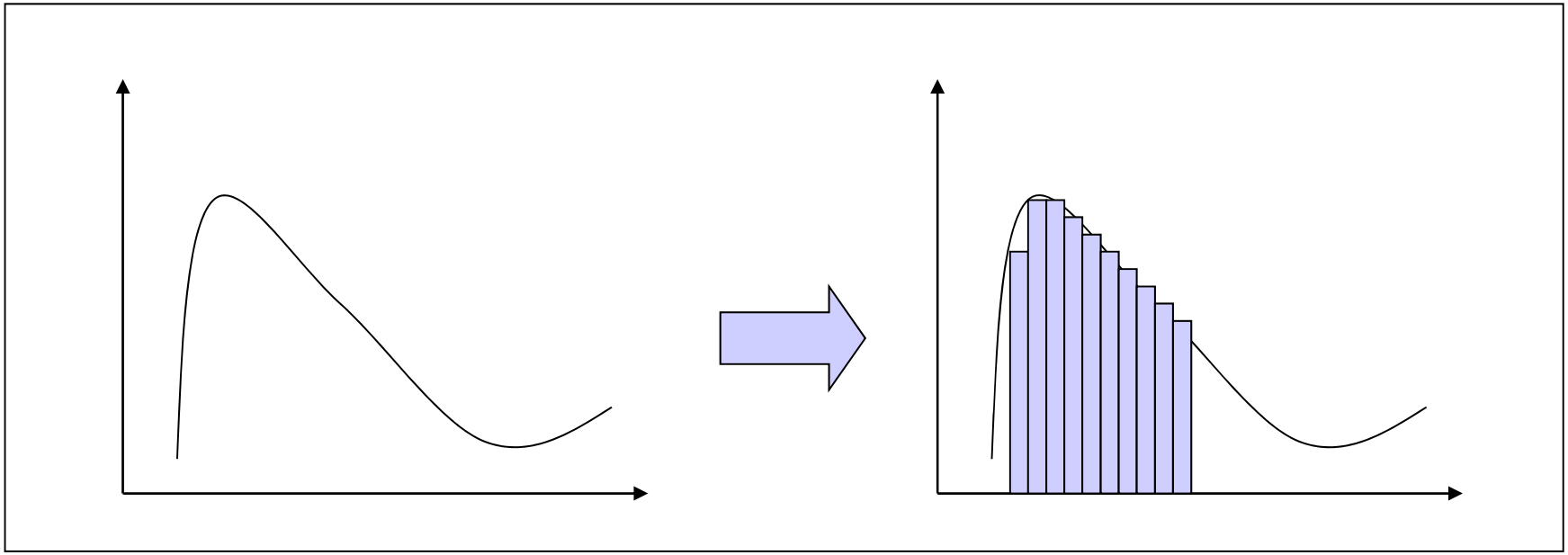
```
// Accpeting data and sum up
   void jButton1_actionPerformed(ActionEvent e) {
   int columnIdx = jTable1.getSelectedColumn();
   double columnSum = 0;
   double tmp = 0;
   for (int i =0; i < jTable1.getRowCount(); i++) {
     String tmpString = (String) jTable1.getValueAt(i, columnIdx);
     if (tmpString != null) {
       tmp = Double.parseDouble(tmpString);
     }
     else tmp = 0;
     columnSum += tmp;
   }
   String txt1 = Double.toString(columnSum);
   jLabel3.setText(txt1);
  }              // end of method of jButton1_action…
}                //end of class TableFrame
```

… End

Numerical algorithm

Basic Idea

Solve a mathematical problem only by repeating a set of simple operations.

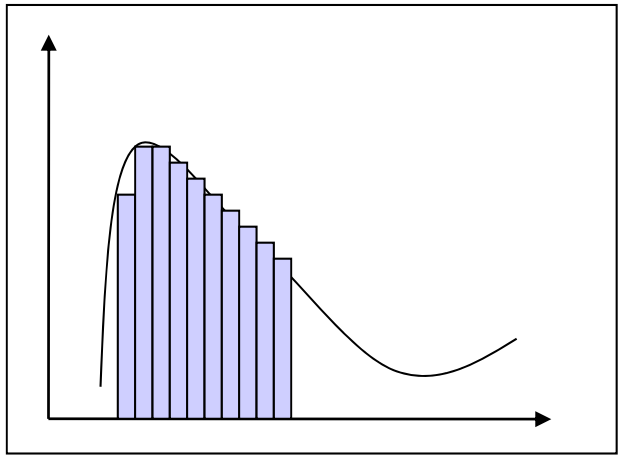


Errors in numerical calculation

Roundoff error:

1/3 = 0.33333333333333333333……

Truncation error:



Consider using long and double instead of int and float, respectively.

Basic Numerical Algorithm (1)

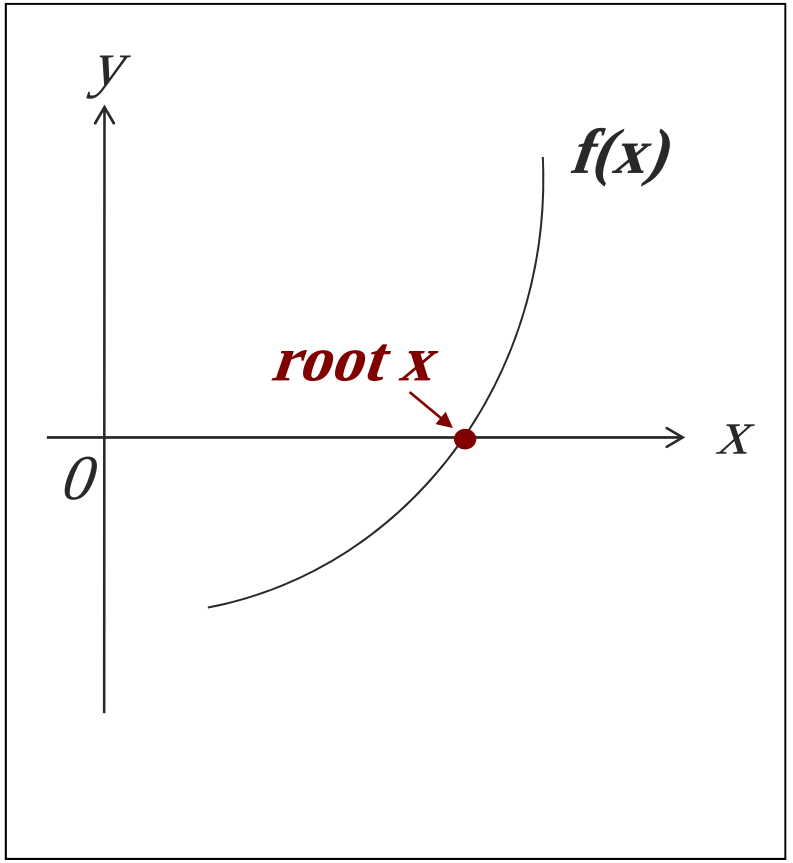
Roots Finding

Root(s) of *f(x)*:

Solution(s) of 1D-equation

$f(x) = 0$

Graphically, *x*-intercepts of the curve of *f(x)*



Bisection Method

Based on

Intermediate Value Theorem,

assuming 2 values: *x1* and *x2*

$f(x1) < 0$ and $f(x2) > 0$,

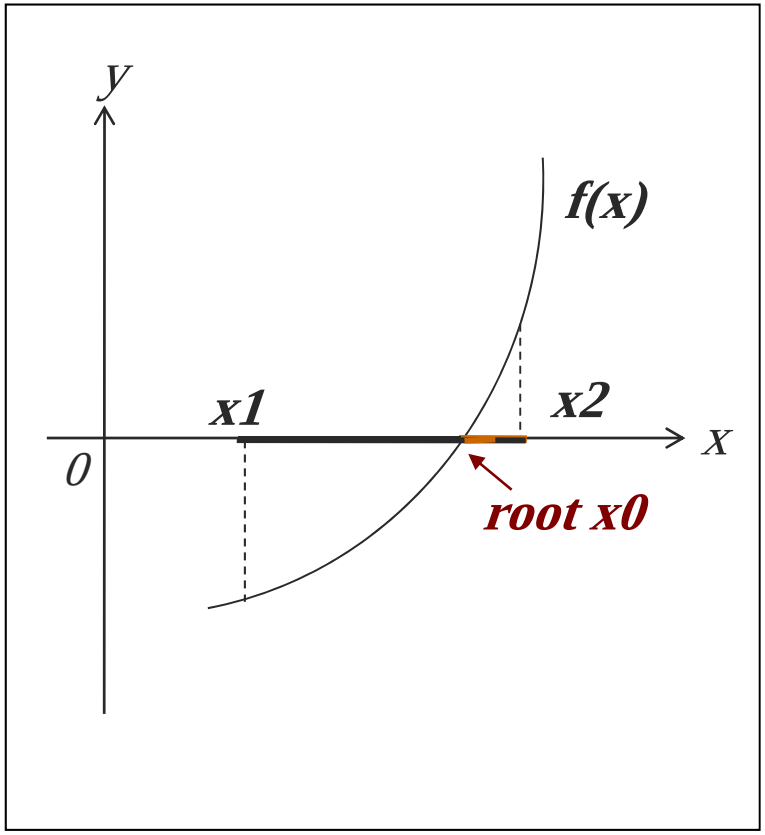
If this is the case (and the function f is continuous),

there must be at least one

value *x0* that falls between

x1 and *x2* such that

$f(x0) = 0$.



Practice 16: Root finding (Bisection)

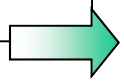
Finding root by bisection:

$$2 + x^2 - (x-1)\,x^2 = 0$$

Practice 17: Root finding (Bisection)

Prac17Bisect.java (1)

```
public class Prac17Bisect {              // Root finding, July  8, 2004
  // The function that we are going to find a root for.
  public static double f(double x) {
       return 2 + x*x - (x-1)*x*x;
    }
  // The main method
  public static void main(String[ ] argv) {
     double left, right, middle, middley, accuracy;
     int i=0;
     left = 0;
     right = 4.0;
     accuracy=0.0001;
     System.out.println ("Find root by bisection"+"\n");
     System.out.println (" x-left of interval: "+left+"\n"+" x-right of
       interval "+right);
     System.out.println (" Accurary : "+accuracy+"\n");
```



Practice 17: Root finding (Bisection)

Prac17Bisect.java (2)

```
if (f(left) * f(right) > 0)
    System.out.println("Function does not change sign");
else  {
    // Keep halving the size of the interval
    // until it is as small as our required accuracy.
    while (right - left > accuracy)     {
      // Compute the value of the function
      // At the middle of the interval
    middle = (left + right) / 2;
    middley = f(middle);
```

Practice 17: Root finding (Bisection)

Prac17Bisect.java (3)

```
// Switch to using whichever half the sign now changes in.
      if (middley * f(right) <= 0)   {
        left = middle;
        }
      else   {
        right = middle;
        }
      System.out.println((i+1)+"   "+left+"   "+right);
      i++;
    }
      // Output our result.
     System.out.println("\nRoot is at");
     System.out.println("x = "+left);
  }
 }
}
```

... End

Practice 17: Root finding (Bisection)

```
Find root by bisection
 x-left of interval: 0.0
 x-right of interval: 4.0
 Accurary : 1.0E-4

1   2.0   4.0
2   2.0   3.0
3   2.0   2.5
4   2.25   2.5
5   2.25   2.375
6   2.3125   2.375
7   2.34375   2.375
8   2.34375   2.359375
9   2.3515625   2.359375
10   2.35546875   2.359375
11   2.357421875   2.359375
12   2.3583984375   2.359375
13   2.35888671875   2.359375
14   2.359130859375   2.359375
15   2.3592529296875   2.359375
16   2.3592529296875   2.35931396484375

Root is at
x = 2.3592529296875
```

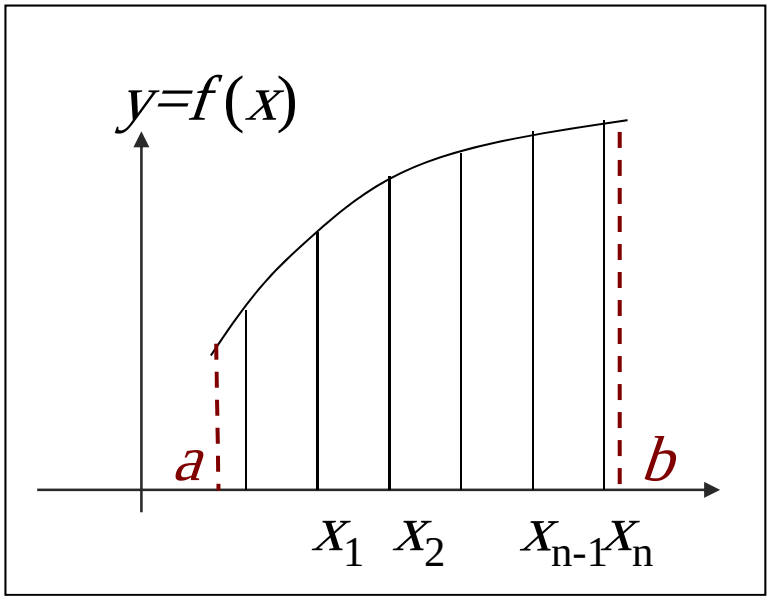
change to higher accuracy, to see how many iterations need?

Stability & Convergency

Basic Numerical Algorithm (2)

Integration of function

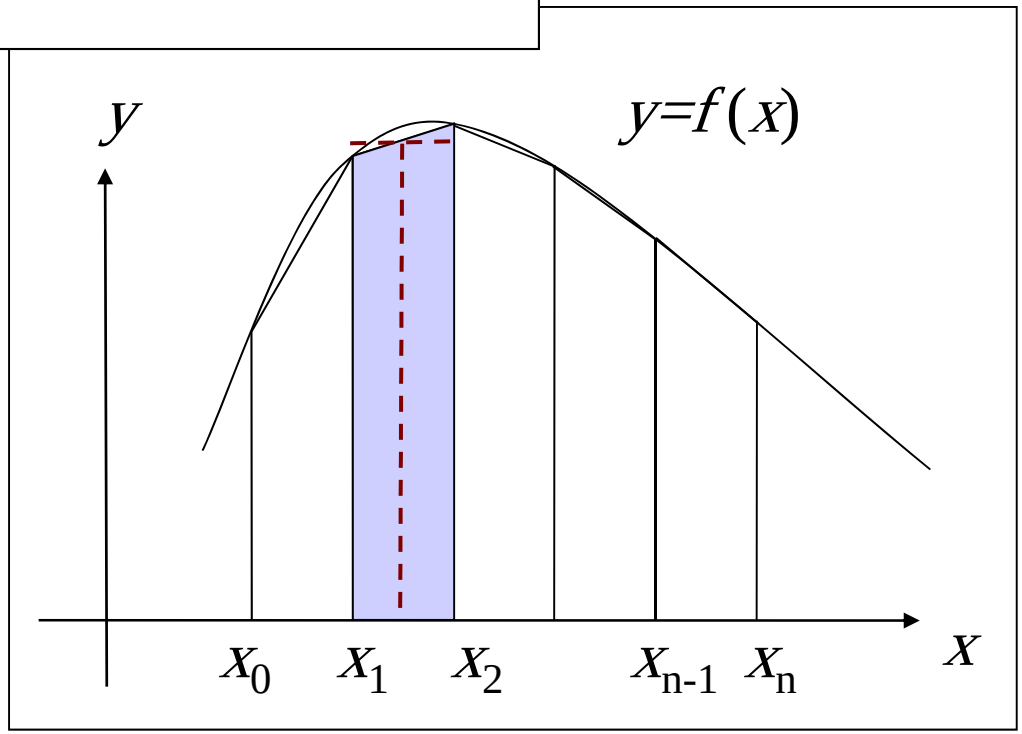
- Perhaps with the **longest history** in numerical algorithms.
- Useful as integrals of elementary functions **cannot** always be computed **analytically**.
- Surprisingly **simple** when written as a computer program.



$$I = \int_a^b f(x)dx \qquad h = \frac{b-a}{n}$$

Trapezoidal Rule

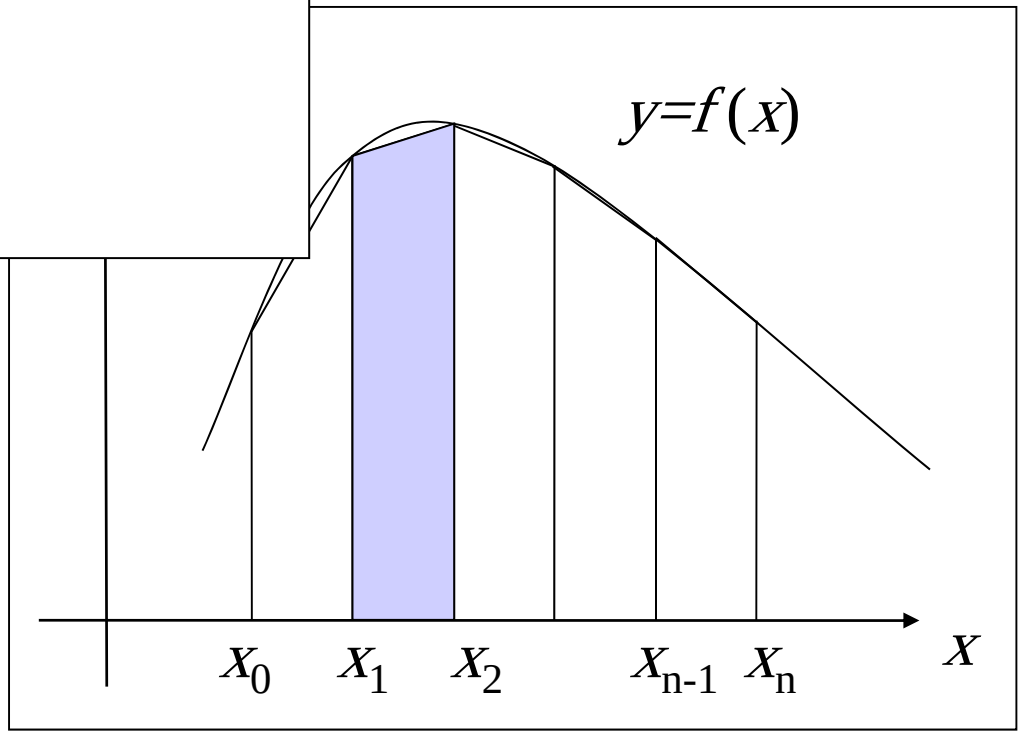
$$\int_a^b f(x)dx = \frac{h}{2}\sum_{i=1}^{n}\{f(x_i) + f(x_{i-1})\}$$



Error greater where $f''(x)$ is large.

Simpson's rule

$$\int_a^b f(x)dx = \frac{h}{3}\{f(a) + 4\sum_{i=1,3,5...}^{n} f(a+ih) + 2\sum_{i=2,4,6...}^{n} f(a+ih) + f(b)\}$$



Practice 18: Integral (trapezoidal)

Prac18.java

```
// Practice18: Integral          July 8, 2004
public class Prac18 {
  public static void main(String[ ] argv) {
    double total, x1, y1,x2,y2;
    total = 0;
    double a=1.0;
    double b=2.0;
    int n=100;
    double h=(b-a)/n;
    for (int i=0; i<=99; i++)   {
        x1 = 1 + i/100.0;
        y1 = 1 / (x1 + 1);
        x2= 1 + (i+1)/100.0;
        y2=1 / (x2 + 1);
        total += (y1+y2)*h/2.0;
   }
    System.out.println(total);
  }
}
```

$$\int_{1.0}^{2.0} \frac{dx}{1+x} = ?$$

n=100,
n=1000,
n=10000
...

Exercise 5 (Submit by July 21, 2004)

Modify the code of Practice 17:

1) Make it accept initial data
   (left, right, accuracy) by keyboard.
2) Make it output the result into a file
   called “result.dat”.

On Final Assignment

Two questions (50% of final score)

Set out: One today, Another next week

Requests:

1) Run your program and get normal result

2) Send a **java code file** and a **MS Word file**
as project report: algorithm, GUI design, test results and analysis.

3) All by email to: ying@q.t.u-tokyo.ac.jp

miti@cvl.iis.u-tokyo.ac.jp

4) **Deadline: July 31, 2004 (also 5 exercises)**

Notice

You will be given no mark if:

- **Your program does not work;**
- **You missed the deadline;**
- **Your codes are unreasonably similar to others'.**

Final Assignment 1

Write a program to calculate the integral of a function like:

$$f(x) = k_1 \sin(ax) + k_2 \cos(bx) \qquad [x_a, x_b]$$

1) Constants such as k_1, k_2, a, b, x_a and x_b can be specified by a user (not by yourself!);
2) Create a GUI to show input panel, plot of $f(x)$ and fill the area for integration;
3) Discuss the error generated from this algorithm by comparing the answer with analytical result.

Computer Programming

IIE (799-18)

(10) July 15, 2004

Lecturer: Ying CHEN

ying@q.t.u-tokyo.ac.jp

TA: Miti Ruchanurucks

miti@cvl.iis.u-tokyo.ac.jp

GUI and Applet / Numerical Algorithm

1. Getting Started with swing (7) 6/24
2. Using Swing components: frame, button, label
3. Event handling

Mathematical functions (8) 7/1

4. More on Swing components: input-box, plotting
5. More on Swing components: table (9) 7/8

Basic Idea, Error
Root finding, Integration

Final Assignment 1 / Questionnaire

6. Java Applet

Monte Carlo Method,Random Number

Final Assignment 2

Closing

Today (10) 7/15

Java Applet

Applet

A Java program to be included in HTML pages and executed in Java-enabled browsers.

AWT ~ Java Applet
Swing ~ Java JApplet

Java Applet and Java Application

Java Application: ordinary Java program
Java Applet: Java program for web page

Main difference

Applets do **not** need to implement a main method **main ( )**;
Applets are executed by HTML.

How a Java Applet works

- **Importing Classes and Packages:**

```
import java.awt.*;
import javax.swing.*;
```

- **Defining an Applet Subclass:**

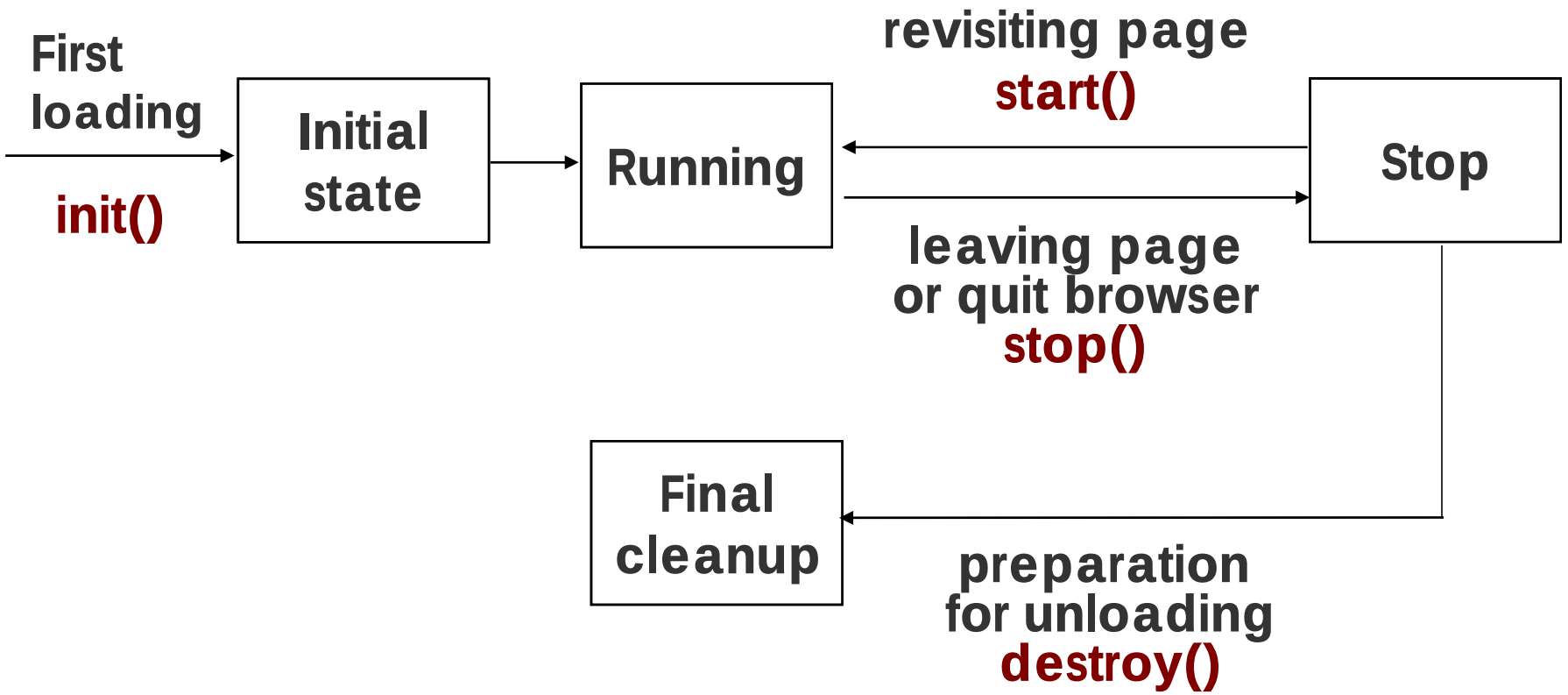
```
public class xx extends JApplet {   }
```

- **Implementing Applet Methods:**

init(), start(), paint(), stop(), destroy()

- **Running/Viewing an Applet:** → HTML
  - through a web browser
  - Unix: "appletviewer xx.html"

Life Circle of Applet



- Security Manager: “sandbox”

Including an Applet into HTML

HelloApplet.java

```
import java.awt.*;
import javax.swing.*;

public class HelloApplet
extends JApplet {

...

}
```

compile
“.class”

Hello.html

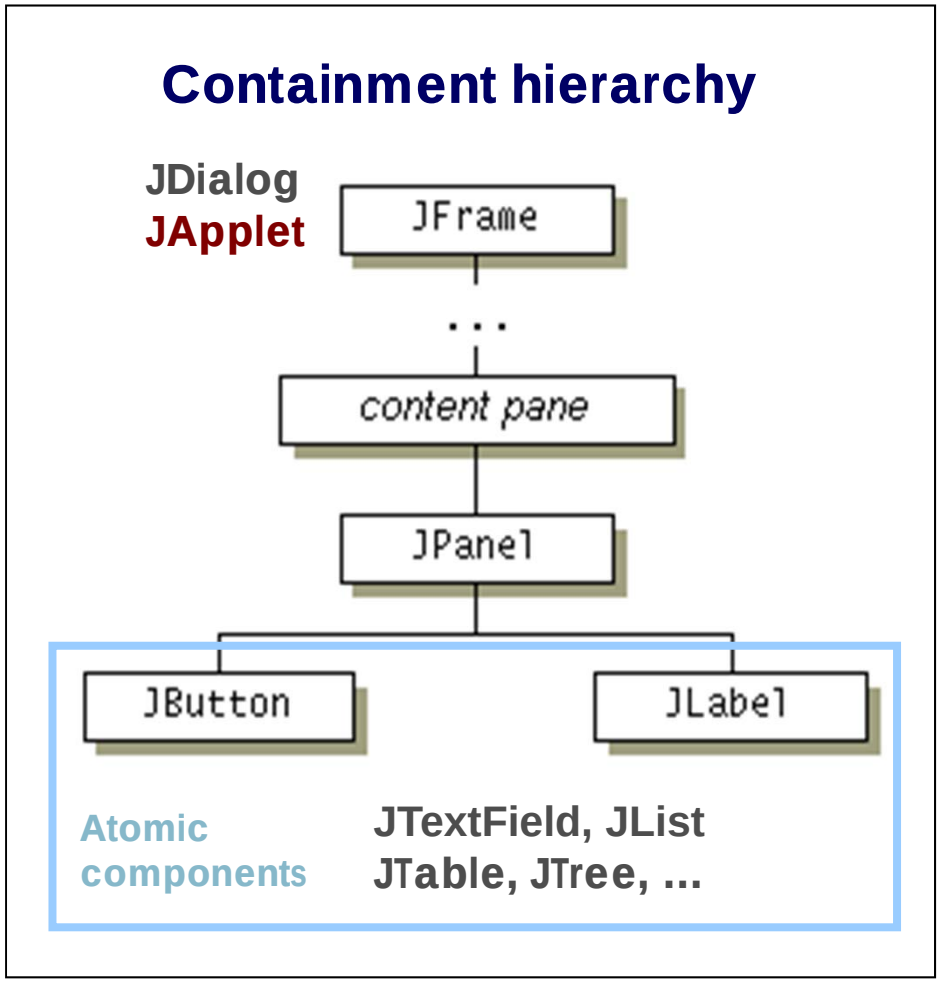
```
<html>

...

<applet>
code="HelloApplet.class"
width=200 height=150>

</applet>

...

</html>
```

<Applet> tag

How to write a JApplet

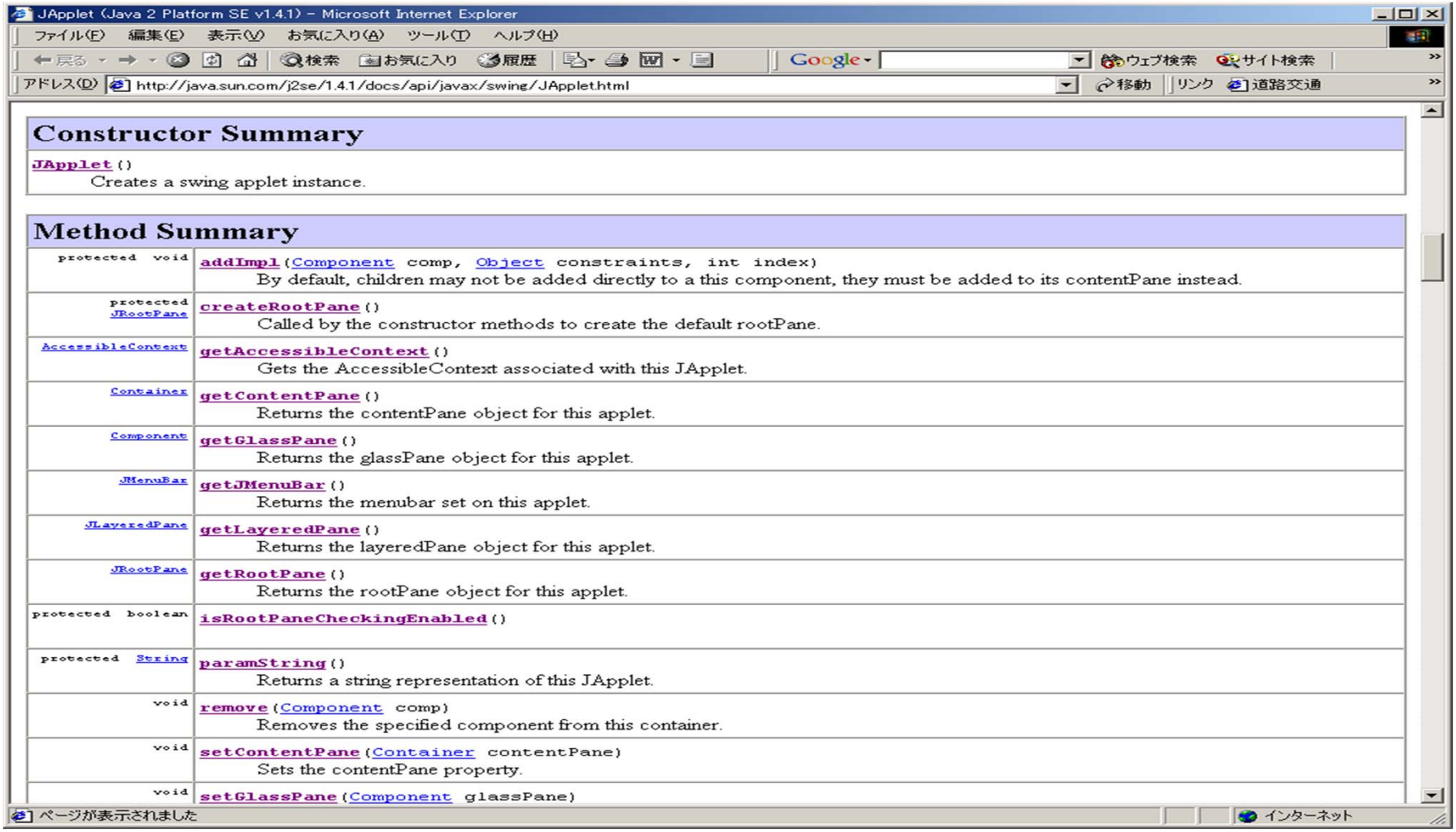
- public class HelloApplet
  extends JApplet { ... }
- public void init() {
  MyPanel contentPane
  = new MyPanel();
  setContentPane();
  ... }
- class MyPanel extends
  JPanel { ... }

Every Applet needs to override at least one of 3 methods: init(), start(), paint()



javax.swing.JApplet class

http://java.sun.com/j2se/1.4.1/docs/api/javax/swing/JApplet.html



Practice 19: My first Java Applet

HelloApplet.java

```
import java.awt.*;
import javax.swing.*;

public class HelloApplet extends JApplet {
    public void init() {
        MyPanel contentPane= new MyPanel();
        setContentPane(contentPane);
        this.setSize(400,300);
    }
}

class MyPanel extends JPanel {
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawString("Hello World !", 80,30);
    g.drawString("- My first java applet", 80,60);
    g.drawOval(80,90,50,30);
    g.fillRect(150,90, 50,30);
    }
}
```

Practice 18: My first Java Applet

Hello.html

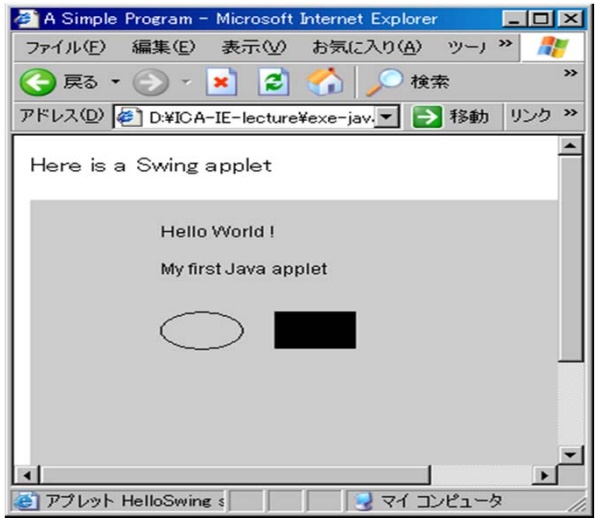
```
<HTML>
<HEAD>
<TITLE>A Simple Program</TITLE>
</HEAD>
<BODY>
Here is a swing applet
<p>
<APPLET CODE="HelloApplet.class" WIDTH=400 HEIGHT=300>
</APPLET>
</BODY>
</HTML>
```

Practice 19: My first Java Applet

2 ways of running

By JDK command:
>appletviewer Hello.html

In a web browser:
open local file:



Basic numerical algorithm (3)

Monte Carlo Method

- **A statistical methods by using sequences of random numbers.**

  Also called Random sampling, Statistic simulation, Statistic experiment.

- **Apply to numerical calculation:**

  Multiply Integral, Equations' solution, …

- **Apply to simulation of Probabilistic processes.**

Monte Carlo algorithm

- **Probability distribution function (pdf)**– the physical /mathematical system described by a set of pdf's.
- **Random number** generator – a source of random numbers with **uniform distribution** on the unit interval.
- **Sampling rule** – a prescription for sampling from the specified pdf's.
- **Scoring** - outcomes must accumulate into overall scores for quantities of interest.
- **Error** estimation – measure of the statistical error (variance) as a function of the number of trials.

Random numbers generator

Linear congruential generator - Uniform Deviates:

$$I_{j+1} = (aI_j + c) \bmod m$$

where,

mod: modulus

$I_0, I_1, I_2, \ldots$, each between 0 and $m-1$; I_0 : *seed.*

m is a very large number.

eg. $2^{15}=32768$, $2^{31}=2147483648$

Function of random numbers

System-supplied random number

Math.random()

returns a double [0, 1] with uniform distribution

Monte Carlo Integration

To integrate a function over a complicated domain

D: complicated domain.

D': Simple domain, superset of D.

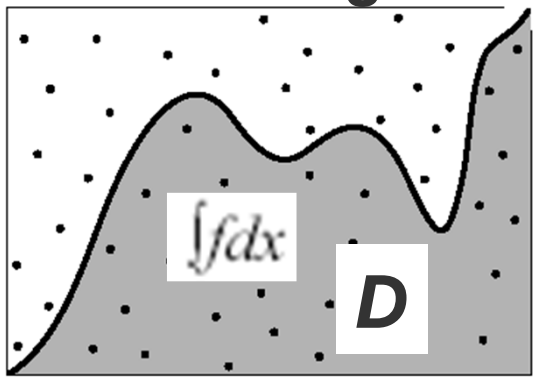
Picking random points over D':

Counting: N: points over D

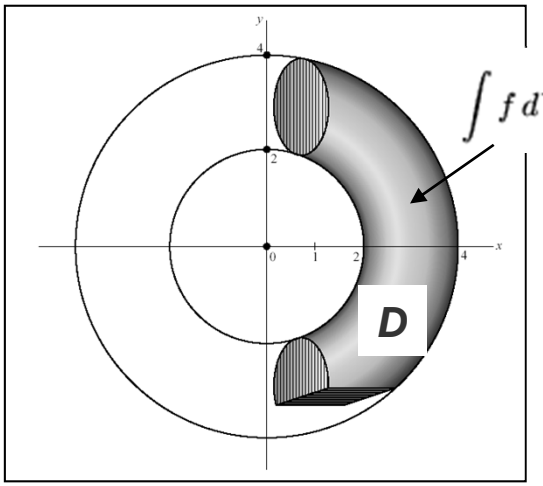
N': points over D'

$$\frac{V(\text{or } A)_D}{V(\text{or } A)_{D'}} = \frac{N}{N'}$$

D': rectangular



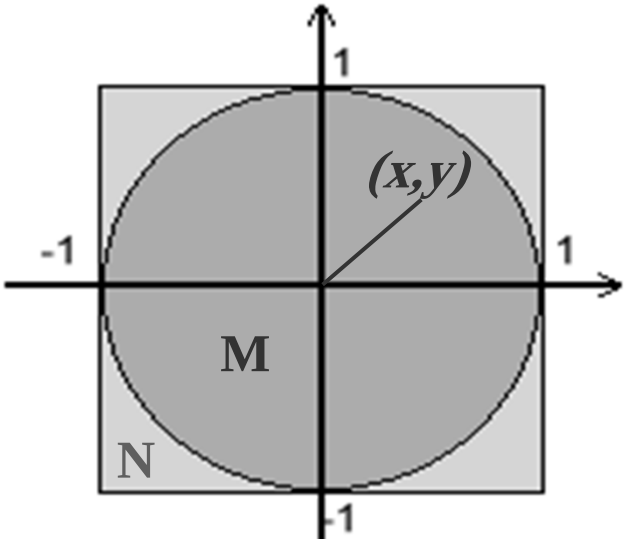
D': circle



Practice 20-1: Estimating π by Monte Carlo Method

The probability of a random point lying inside the unit circle:

$$\mathbf{P}\left(x^2+y^2<1\right)=\frac{A_{circle}}{A_{square}}=\frac{\pi}{4}$$



If pick a random point *N* times and *M* of those times the point lies inside the unit circle:

$$\mathbf{P}^{\circ}\left(x^2+y^2<1\right)=\frac{M}{N}$$

if *N* becomes very large, $\mathbf{P}=\mathbf{P}^{0}$ $\rightarrow$ $\pi=\frac{4\cdot M}{N}$

Practice 20-1: Estimating π by Monte Carlo Method

Create a program to calculate π now !
(testing various n: large number)

Result:

Calculate Pi by Monte Carlo Method

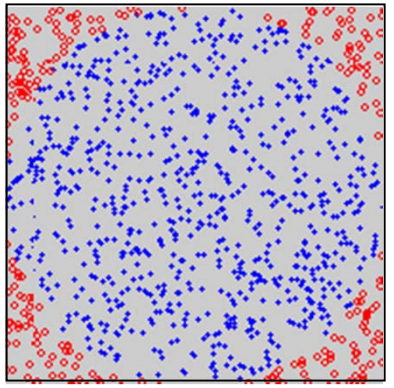
n= 10000 Pi= 3.1388

n= 100000 Pi= 3.1452

n= 1000000 Pi= 3.14164

n= 10000000 Pi= 3.1422784

…



Practice 20-1: Estimating π by Monte Carlo Method

RandomPi.java

```
public class RandomPi {
   public static void main(String[ ] arg)  {
   System.out.println ("Calculate Pi by Monte Carlo Method"+"\n");
      double x, y, pi;
      int n[ ]=new int [4];
      n[0]=10000;  n[1]=100000;
      n[2]=1000000;  n[3]=10000000;
      for (int k=0; k<n.length;k++) {
           int s=0;
           for(int i=1;i<n[k];i++) {
               x=Math.random();
               y=Math.random();
              if (x*x+y*y<=1.0) s++;
           }
     pi=4*(double)s/(double)n[k];
     System.out.println ("n= "+n[k]+"\t"+"Pi= "+pi+"\n");
     }
  }
}
```

Practice 20-2: Estimating π: calculation + plot

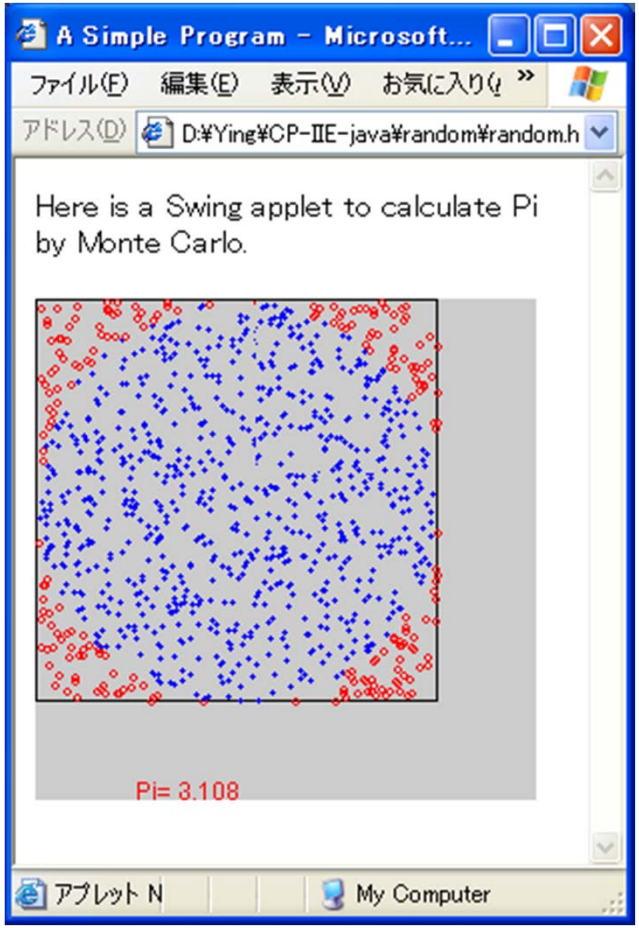
PlotRandomPi.java(1
)

```
import java.awt.*;
import javax.swing.*;

public class NrandomSwing extends JApplet {
   public void init() {
      MyPanel contentPane= new MyPanel();
      setContentPane(contentPane);
       this.setSize(300,300);
   }
 }

 class MyPanel extends JPanel {
       int i, x, y ,n=1000, width=200;
       double x1,y1, r, pi;
       int s=0;
           g.drawRect(0,0,width,width);
           r=((double)width)/2.0;
```

Practice 20-2: Estimating π: calculation + plot

PlotRandomPi.java(2)

```
for(i=0;i<n;i++)  {
     x1=width*Math.random();
     y1=width*Math.random();
     x=(int)(x1);
     y=(int)(y1);
         if(((x-r)*(x-r)+(y-r)*(y-r))<=r*r) {
            g.setColor(Color.blue);
            g.fillOval(x,y,3,3);
            s++;
          }
         else  {
            g.setColor(Color.red);
            g.drawOval(x,y,3,3);
          }
   }  //end for
      pi=4.0*(double)s/(double)n;
      String txt=Double.toString(pi);
      g.drawString("Pi= "+txt, 50,250);
  }  // end paint method
}   //end of MyPanel
```



On Final Assignment

Two questions (50% of final score)

Set out: One today, Another next week

Requests:

1) Run your program and get normal result

2) Send a java code file and a MS Word file
as project report: algorithm, GUI design, test results and analysis.

3) All by email to: ying@q.t.u-tokyo.ac.jp

miti@cvl.iis.u-tokyo.ac.jp

4) Deadline: July 31 (also 5 home exercises)

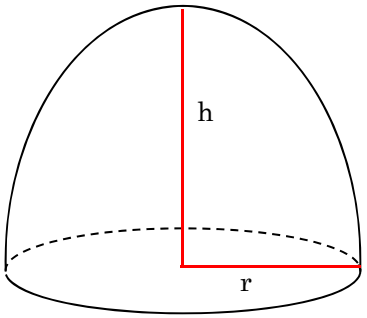
Notice

You will be given no mark if:

- **Your program does not work;**
- **You missed the deadline;**
- **Your codes are unreasonably similar to others'.**

Final Assignment 2

- Write a program to calculate the volume of an paraboloid with radius r, height h.
- To design a GUI to deal with input and output (see a sample below).
- Compare the results by analysis and by Monte Carlo methods.



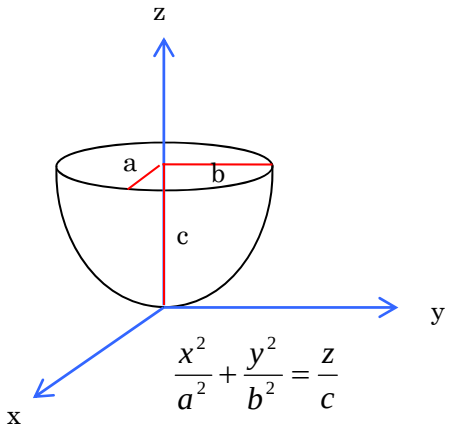
Radius [ ] Height [ ] Total points [ ]

Volume(analysis):

Volume(Monte Carlo):

Error (%):

Calculate



Closing the lecture

Summary

- Basic Java language
- Concept of object-oriented programming
- Introduction to GUI
- Brief introduction to numerical analysis

Thanks Mr. Miti for nice TA !

Thank you very much !

Welcome to join “Fundamental Numerical Analysis”

(2004 Winter Semester: Oct. 2004)

Exercise 1

Write a program to give all prime number between 100~200.

- An integer greater than one is called a **prime number** if its only positive divisors (factors) are one and itself. (For example, the prime divisors of 10 are 2 and 5; and the first six primes are 2, 3, 5, 7, 11 and 13.) ;
- m is a prime number if it has no integer quotient when divided by integer between 2~$\sqrt{m}$;
- $\sqrt{m}$ in Java is: Math.sqrt(m), m is double floating.

Request on Exercises

- Subject of the email should include
  "CP-IIE-exe1".
- Subject or Body should include
  "Student Number" if you don't use ECC account.
- You should **Run** the program before sending.

Email to us by June 3.

ying@q.t.u-tokyo.ac.jp

miti@ cvl.iis.u-tokyo.ac.jp

Exercise 2 (submit by June 23)

Solve the quadratic equation: $ax^2+bx+c=0$

$a=0$, not a quadratic equation;
$b^2-4ac=0$, 2 same real roots.
$b^2-4ac>0$, 2 different real roots: $-\frac{b}{2a}\pm\frac{\sqrt{b^2-4ac}}{2a}$;

$b^2-4ac<0$, 2 conjugate complex roots: $-\frac{b}{2a}\pm i\frac{\sqrt{-(b^2-4ac)}}{2a}$.

1) Create a class to hold the variables a,b and c;
2) Extend above class to create a class with the constructor to initialize the variables, and method(s) to calculate the roots of the equation;
3) Execute calculations with certain (a,b,c) for above 4 cease.

Request on Exercises

- Subject of the email should include
  "CP-IIE-exe2".
- Subject or Body should include
  "Student Number" if you don't use ECC account.
- You should Run the program before sending.

Email to us by June 23.

ying@q.t.u-tokyo.ac.jp

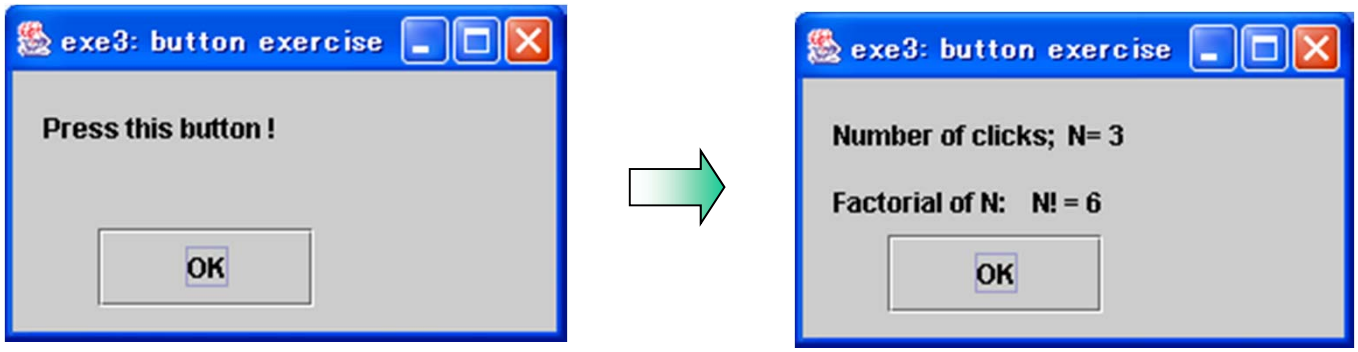
miti@ cvl.iis.u-tokyo.ac.jp

Exercise 3 (submit by July 7, 2004)

Based on Practice13-2, write a program,

1) To display a frame as show in (a);

2) Clicking the button, to show the number of clicks N and its factorial N! = 1x2 x … xN ,as shown in (b).



Request on Exercises

- Subject of the email should include
  "CP-IIE-exe3".
- Subject or Body should include
  "Student Number" if you don't use ECC account.
- You should **Run** the program before sending.

Email to us by July 7.

ying@q.t.u-tokyo.ac.jp

miti@ cvl.iis.u-tokyo.ac.jp

Exercise 4 (submit by July. 14, 2004)

Based on Practice 14 and Practice 15,

1) Create a GUI to calculate the value of

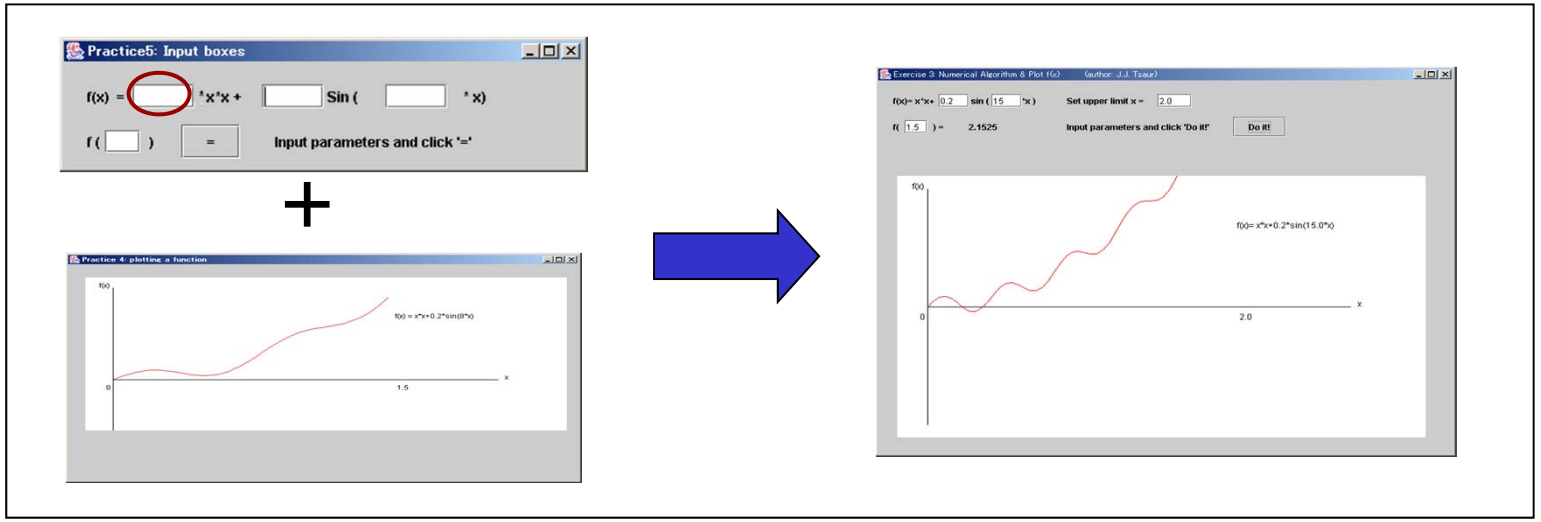
$$f(x) = cx^2 + aSin(bx)$$

by keyboarding *a*, *b*, *c*, and variable *x*, show result;

2) Display the curve of function in same interface.

3) Re-calculate *f(x)* & Redraw the curve of *f(x)*

by inputting new parameters (Use "myobject.updateUI();")



Request on Exercises

- Subject of the email should include
  "CP-IIE-exe4".
- Subject or Body should include
  "Student Number" if you don't use ECC account.
- You should **Run** the program before sending.

Email to us by July 14.

ying@q.t.u-tokyo.ac.jp

miti@ cvl.iis.u-tokyo.ac.jp

Exercise 5 (Submit by July 21, 2004)

Modify the code of Practice 17:

1) Make it accept initial data (left, right, accuracy) by keyboard.

2) Make it output the result into a file called “result.dat”.

Request on Exercises

- Subject of the email should include "CP-IIE-exe5".
- Subject or Body should include "Student Number" if you don't use ECC account.
- You should Run the program before sending.

Email to us by July 21.

ying@q.t.u-tokyo.ac.jp

miti@ cvl.iis.u-tokyo.ac.jp

Final Assignment 1

Write a program to calculate the integral of a function like:

$$f(x) = k_1 \sin(ax) + k_2 \cos(bx) \qquad [x_a, x_b]$$

1) Constants such as k_1, k_2, a, b, x_a and x_b can be specified by a user (not by yourself!);
2) Create a GUI to show input panel, plot of $f(x)$ and fill the area for integration;
3) Discuss the error generated from this algorithm by comparing the answer with analytical result.

On Final Assignment

Two questions (50% of final score)

Set out: One today, Another next week

Requests:

1) Run your program and get normal result

2) Send a **java code file** and a **MS Word file**
as project report: algorithm, GUI design, test results and analysis.

3) All by email to: ying@q.t.u-tokyo.ac.jp

miti@cvl.iis.u-tokyo.ac.jp

4) **Deadline: July 31, 2004 (also 5 exercises)**

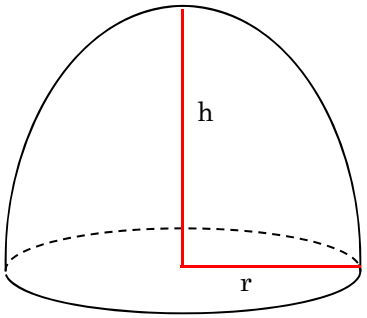
Notice

You will be given no mark if:

- **Your program does not work;**
- **You missed the deadline;**
- **Your codes are unreasonably similar to others'.**

Final Assignment 2

- Write a program to calculate the volume of an paraboloid with radius r, height h.
- To design a GUI to deal with input and output (see a sample below).
- Compare the results by analysis and by Monte Carlo methods.



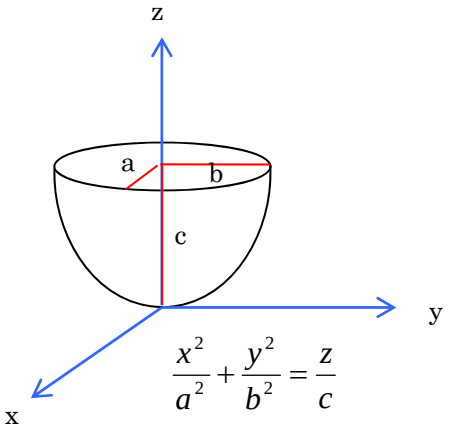
Radius [ ] Height [ ] Total points [ ]

Volume(analysis):

Volume(Monte Carlo):

Error (%):

Calculate



On Final Assignment

Two questions (50% of final score)

Set out: One today, Another next week

Requests:

1) Run your program and get normal result

2) Send a **java code file** and a **MS Word file**
as project report: algorithm, GUI design, test results and analysis.

3) All by email to: ying@q.t.u-tokyo.ac.jp

miti@cvl.iis.u-tokyo.ac.jp

4) **Deadline: July 31 (also 5 home exercises)**

Notice

You will be given no mark if:

- **Your program does not work;**
- **You missed the deadline;**
- **Your codes are unreasonably similar to others'.**