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
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)


• 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

• 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.ec.c.u-tokyo.ac.jp/cgi-bin passwd.cgi

User’s name
Current password
New password
New password

GO
“Three kinds of world”
iMac Terminal

1. iMac
2. Happy Hacking keyboard
3. Mouse
iMac: Login/Logout

### Login
- **User Name**
- **Password**
- **Return**

### Logout
- **logout+shut down**

### Shutdown
iMac: Applications

- Finder
- Mail
- Safai: Web browser
- X11, Terminal: Unix
- Remote Desktop Connection: Windows environment
- Microsoft: 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
  ...

Open Unix

X11 or

Terminal

Close

exit
Windows Environment

login

Remote Desktop Connection

User Name
Password

logout

Application

- GiveWin2
- Microsoft Office
- Mozilla
- Opera
- SAS (statistics)
- TSP (statistics)
- Adobe Acrobat 6.0 Standard
- Adobe Photoshop Element
- video LAN

...
File Sharing Between Unix and iMac

iMac/Finder

Unix Home Directory → read Unix files

directory: /Desktop

→ read iMac files

Unix/Terminal
Mail Service

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

Mail client

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

Mail (need setting)
Terminal Rooms

**Asano Campus**
- Information Technology Center 1F 5F

**Komaba Campus**
- Information education 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) May 13, 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

Shell

Hardware

Kernel

Programs
Unix: File System, Path

```
/  root  /usr/, /bin/,…

bin  etc  usr  lib  dev

usr2  bin  lib

chen  taka

file1  test  file1

file2
```

/ usr/ usr2/ chen/,…

/ usr/ usr2/ chen/test/, ~/chen/test/
Unix Environment (X11)

Shell for Command:

$ cd test

prompt line command

>, $, #, ...
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   ls
 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

GNUMstep    atokdicts.tar    registry
GNUMstep.sun mytest    test

GNUMstep> ls -F
GNUMstep/ atokdicts.tar registry/ test
GNUMstep.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
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”.
Simple Unix Commands

Getting Help: Online Manual

```bash
> 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
  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

Source Code
Human-readable

→

Executive file
Computer-readable

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 &
```

emacs

```
/* My first cup of java */

class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

“Hello.java”

⇒ save
Practice 3: “Your first cup of Java”

2) Compile 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) May 20, 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 (J 2SE)
  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 Microsystems Inc.

Content of Java Programming

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

Java:
- Portable
- Object-oriented
- Internet appl.
1. Getting Started

Basic Idea

A program
A list of **commands** stored in a text file

How does a program work?

**Compiler/interpreter**

**Source Code**
Human-readable

**Executive file**
Computer-readable
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 &

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

Case sensitive!
Practice 3: “Your first cup of Java”

2) Compile source file

```
> javac Hello.java
```

```
Hello.class
```

3) Run the program

```
> java Hello
```

See What happens?
1) Creating Java source code;
   `source_name.java`

2) Compiling source:
   `javac source_name.java`
   \[\rightarrow source_name.class\]

3) Running:
   `java bytecode_name (without "\class")`
class Class_Name {
    public static void main (String[] args) {
        ...variable declarations...
        ...statements...
    }
}
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 method
    {
        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
1. Getting Started

Style of Coding

• 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 “// ......”
2. Language Basics

Variables

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

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

Variable Declarations

Format: Type Name;

eg. Int sum; float tax; char c;
### 2. Language Basics

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

**Cast**: type conversion, eg. `(int)2.73, (double)x`
## 2. Language Basics

### Operators

#### Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>Remainder</td>
<td></td>
</tr>
</tbody>
</table>

3%5=3
5%5=0
2. Language Basics

Relational Operators

To determine the relationship between two values

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal</td>
</tr>
</tbody>
</table>
### Logical Operators

**To form multi-part decisions**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Use</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td><code>&amp;&amp;</code></td>
<td><code>op1 &amp;&amp; op2</code></td>
</tr>
<tr>
<td>or</td>
<td>`</td>
<td></td>
</tr>
<tr>
<td>not</td>
<td><code>!</code></td>
<td><code>!op</code></td>
</tr>
<tr>
<td>and</td>
<td><code>&amp;</code></td>
<td><code>op1 &amp; op2</code></td>
</tr>
<tr>
<td>or</td>
<td>`</td>
<td>`</td>
</tr>
</tbody>
</table>

Return true if `op1` and `op2` are both true, conditionally evaluate `op2`.

Return true if either `op1` and `op2` are true, conditionally evaluate `op2`.

Return true if `op` is false.

Return true if `op1` and `op2` are both true, always evaluate `op1` and `op2`.

Return true if either `op1` and `op2` are true, always evaluate `op1` and `op2`. 
2. Language Basics    Assignment Statement

Format: \[ \text{var} = \text{expression}; \]

eg. \[ \text{sum} = (12 + 21 - 23) * 41 + 15; \]
    \[ \text{average} = \text{sum} / 5; \]
    \[ \text{sum} = \text{sum} + 26.2; \]
    \[ \text{char1} = 'A'; \]
    \[ \text{string1} = "The University of Tokyo"; \]
2. Language Basics

## Initializing Variables

### Declaring + Initializing Variables

**Separately:**

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

**Simultaneously:**

```java
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 = ...
Hello.java

/* My first cup of java
 * 2004. 5. 20. */ // comment line

class First {
    public static void main(String[] args) // define a class
    {
        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");
    }
}
2. Language Basics

**System.out.println: printout + change line**

```
System.out.println("a=\"a +\" \t\"b=\"b +\n"c=\"c \\
\n18  4
...
```

**System.out.print: printout (without change line)**
2. Language Basics

Array

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

Declaring: \texttt{type[ ] ArrayName = new type[size]} \\

eg. An one dimensional array: \texttt{oneArray} \\
An array element: \texttt{oneArray[i]} \\
length of array: \texttt{oneArray.length}
2. Language Basics

Declaring + Initializing An Array

**Separately:**

\[
\begin{array}{cccccc}
2 & 3 & 1 & 6 & 5 \\
\end{array}
\]

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

**Simultaneously:**

```java
int[] oneArray = {2, 3, 1, 6, 5};
```
2. Language Basics

Array

Multidimensional 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;

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 …

…
3. Control Flow Statement

**if ... else Statement**

```c
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.

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

“if nest”

```java
if (condition1) {
    if (condition2) {
        statement1;
    }
    else {
        statement2;
    }
} else {
    statement3;
}
else ...
```
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**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>48</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td>72</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
</tr>
</tbody>
</table>
/* Practice 4. 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.println(" \ n");
        }
    }
}
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 \( 2 \sim \sqrt{m} \);
- \( \sqrt{m} \) 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
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 Statement

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

Condition:  \(<\text{exp}>\) conditional op. \(<\text{exp}>\)

\(\geq, \leq, \equiv, \ldots\)
for loop:
Iterate action over a range of values.

for ( initial; terminal condition; increment)
{
    code to be iterated;
}
3. Control Flow Statement

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.*

```java
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:
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.println(i*j + " ");
                if (i==j) {
                    System.out.println("\n");
                    continue outer;
                }
            }
        }
    }
}

Labeled
3. Control Flow Statement  

**Result:**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>24</td>
<td>32</td>
<td>40</td>
<td>48</td>
<td>56</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>72</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>
4. Basic of Object-Oriented Programming (OOP)

Basic Concepts

**Procedure-Oriented programming:**
algorithm + data structure $\rightarrow$ code (functions)

**Object-Oriented programming:**
data structure + algorithm $\rightarrow$ code (methods)
4. Basic of Object-Oriented Programming (OOP)

**Basic Concept**

- **Object**: basic unit of OOP
- **Class**: describes the generality of objects
  - **State**: Variables
  - **Behavior**: Methods

**In real world**

**Instance**
4. Basic of Object-Oriented Programming (OOP)

- **Bird Class**
  - **State**: Color, Weight, Size ...
  - **Behaviors**: Fly(), Walk(), Eat() ...
  - **Variables**: Variables
  - **Methods**: Methods

- **Object**
  - **Bird A**
    - Color = gray
    - Weight = 89g
    - Fly()
  - **Bird B**
    - Color = yellow
    - Weight = 61g
    - Eat()
4. Basic of Object-Oriented Programming (OOP)

**Basic Concept**

**Message**
4. Basic of Object-Oriented Programming (OOP)

Basic Concept

Characteristics of Object-Oriented programming

- Encapsulation
- Inheritance
- Polymorphism
4. Basic of Object-Oriented Programming (OOP)

Basic Structure of program

- main Class<name>
  - Member variables
  - main method
    - local variables
    - method body
  - other method<name>
    - local variables
    - method body

- other Class<name>
  - Member variables
  - method<name>
    - local variables
    - method body

File.java

Variable Scope

Same!
4. Basic of OOP

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).
4. Basic of OOP

**Definition of a Method**

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

**Declaration of a Method**

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

- `public`, `private`, ...
- `static`, `final`, ...
- `double`, `int`, `void`, ...
- `static`, `final`, ...

- `void`
eg. 1

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

eg. 2

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

Class

Creating a Class

ClassDeclaration
  Member variables
  Constructor
    method1Declaration
      local variables
      method body
    method2Declaration
      local variables
      method body
    ...
4. Basic of OOP  

Declaration of a Class

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

......

}
```
/** Prac 5-1, 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?

tyng@as301> java Bird

Exception in thread "main"
java.lang.NoSuchMethodError: main
4. Basic of OOP

Object

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;
4. Basic of OOP

Object

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();`
/** 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
2. Language Basics
3. Control Flow Statement
4. Basic of Object-Oriented Programming
5. Inheritance and Polymorphism

... 

Now, Please login to your Unix ...
4. Basic of OOP

Basic concepts

Object: basic unit of OOP

Class describes the generality of objects

State Variables

Behavior Methods
4. Basic of OOP

Object

The Life Cycle of an Object

Creating → Using → Cleaning Up
4. Basic of OOP

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**
  \[ \text{objectName}.variableName \]

- **Calling an Object’s Method**
  \[ \text{objectName}.methodName( ); } \]
  \[ \text{objectName}.methodName(\text{argumentList} ); } \]
eg. Practice 5-1 creating a class

```java
/* Prac 5-1, 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");
        }
    }
}
```
/* 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();
    }
}
4. Basic of OOP

Object

Cleaning Up Unused Objects

• The garbage Collector

• Finalization
Constructor of Class

- Constructor is a special method to initialize the variables.
- Its name must be the same as that of the class.
4. Basic of OOP

ClassDeclaration

Class variables;

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

method1Declaration

local variables;

method body

...

Constructor

eg.

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

Constructor
4. Basic of OOP

Creating Object by Constructor

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

```
ClassName objectName = new ClassName (x,y,... );
```
Practice 5-4: Re-write Prac 5.java by constructor

In Main method:

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

In class "Bird":

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

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

Inheritance

Polymorphism: Override, Overload
5. Inheritance and Polymorphism

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 behaviour.

```
class SubclassName extends SuperclassName {
    body of class
}
```
### 5. Inheritance and Polymorphism

#### Keyword “this” and “super”

<table>
<thead>
<tr>
<th>this:</th>
<th>super:</th>
</tr>
</thead>
<tbody>
<tr>
<td>current object</td>
<td>super class of current object</td>
</tr>
<tr>
<td>this.variable</td>
<td>super.variable</td>
</tr>
<tr>
<td>this.Method( )</td>
<td>super.Method( )</td>
</tr>
<tr>
<td>this ( )</td>
<td>super ( )</td>
</tr>
</tbody>
</table>
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 ");
    }
}
5. Inheritance and Polymorphism

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.
5. Inheritance and Polymorphism

Overload of Method

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);
    }
}
5. Inheritance and Polymorphism

Overload of Constructor

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 Polymorphism

Override

```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

"i" is hidden
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

<table>
<thead>
<tr>
<th>1. Getting Started</th>
<th>(3) 5/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Language Basics</td>
<td></td>
</tr>
<tr>
<td>3. Control Flow Statement</td>
<td></td>
</tr>
<tr>
<td>4. Basic of Object-Oriented Programming</td>
<td>(4) 5/27</td>
</tr>
<tr>
<td>5. Inheritance and Polymorphism</td>
<td>(5) 6/10</td>
</tr>
<tr>
<td>6. Interface and Package</td>
<td></td>
</tr>
<tr>
<td>7. Modifiers</td>
<td>Today</td>
</tr>
<tr>
<td>8. Exception</td>
<td></td>
</tr>
<tr>
<td>9. Input/Output Stream</td>
<td>(6) 6/17</td>
</tr>
</tbody>
</table>

... Please login to your Unix ...
Single Inheritance and Multiple Inheritance

Diagram:
- **Shape**
  - **square**
  - **circle**
    - **triangle**
    - ...
An interface is a named collection of method definitions (without implementations).

**Creation**

```java
[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":

```java
interface Shape {
    int x=50, y=50;
    void draw();
} // declare interface

interface Area {
    double calculate();
} // declare interface

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

class Circle implements Shape, Area {
    ...}

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

    public static void main(String[] args) {
        Square sq = new 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.
6. Interface and Package

Creating a package

```
package mypackage;
public class my1 {...}
```

```
package mypackage;
public class my2 {...}
```

```
my1.java
my1.class
```

```
my2.java
my2.class
```

…

directory

mypackage/

Same!
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 a = new mypackage.my1;`
Make package:

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

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

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

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
# 6. Interface and Package

## Java 2 Platform Packages (76)

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.applet</td>
<td>Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.</td>
</tr>
<tr>
<td>java.awt</td>
<td>Contains all of the classes for creating user interfaces and for painting graphics and images.</td>
</tr>
<tr>
<td>java.awt.color</td>
<td>Provides classes for color spaces.</td>
</tr>
<tr>
<td>java.awt.datatransfer</td>
<td>Provides interfaces and classes for transferring data between and within applications.</td>
</tr>
<tr>
<td>java.awt.dnd</td>
<td>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.</td>
</tr>
<tr>
<td>java.awt.event</td>
<td>Provides interfaces and classes for dealing with different types of events fired by AWT components.</td>
</tr>
<tr>
<td>java.io</td>
<td>Provides for system input and output through data streams, serialization and the file system.</td>
</tr>
<tr>
<td>java.lang</td>
<td>Provides classes that are fundamental to the design of the Java programming language.</td>
</tr>
<tr>
<td>java.lang.ref</td>
<td>Provides reference-object classes, which support a limited degree of interaction with the garbage collector.</td>
</tr>
<tr>
<td>java.lang.reflect</td>
<td>Provides classes and interfaces for obtaining reflective information about classes and objects.</td>
</tr>
<tr>
<td>java.math</td>
<td>Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal).</td>
</tr>
<tr>
<td>javax.swing</td>
<td>Provides a set of &quot;lightweight&quot; (all-Java language) components that, to the maximum degree possible, work the same on all platforms.</td>
</tr>
</tbody>
</table>
# 6. Interface and Package

## Classes of java.lang

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

<table>
<thead>
<tr>
<th>modifier</th>
<th>Same class</th>
<th>Same package</th>
<th>Subclass</th>
<th>Other classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>protected</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>default</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>private</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Variables or Methods with the modifier "static" belong to the class but not the object.

```java
class X {
    int a;
    void methodA();
    static int b
    static void methodB()
}
```

Source code

Memory
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.
8. Exception

Exception handling

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

Several kinds of Exception

- Arithmetic Exception
- NullpointerException
- ArrayINdexOutOfRange Exception
- IOException
- ...
8. Exception

**Try-catch-finally blocks**

```java
try { code may have exception }
    catch(ExceptionType1) { action1 }
    catch(ExceptionType1) { action2 }
    ....
finally { final process }
```

*can be abbreviated if unnecessary*
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 Arithmetic Exception");
        }
    }
}
9. Input and Output

Stream: flow of data

Input
- File
- Keyboard

Program

Output
- File
- Monitor

Standard I/O

System.in.read()

System.out.Print()/Println()
9. Input and Output

Hierarchy structure of IO classes

- **Java.io**
  - **Java.io.File**
  - **Java.io.InputStream**
  - **Java.io.OutputStream**
  - **Java.io.Reader**
  - **Java.io.Writer**
  - **Java.io.RandomAccessFile**

Types:
- byte
- character
9. Input and Output

Some classes in java.io package
/* 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);
        }
    }
}
DataOutputStream fo = new DataOutputStream (new FileOutputStream("data.out"));
    fo.writeChar('
');
    fo.writeChars("ff= "+Double.toString(ff));
    fo.close();
} // end try
    catch (IOException e) {
        System.out.println(" IOException ! ");
    } // end catch
} // end main method
} // end main class

```java
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)

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**
  1. Basic of Java programming

- **Later half part**
  1. GUI (Graphics and Applets)
  2. Numerical algorithms
     1. Getting Started with Swing
     2. Using Swing components: frame, button, label
     3. Event handling

(7) 12/18
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.

```java
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
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

Class JComponent

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

All Implemented Interfaces:
   ImageObserver, MenuContainer, Serializable

Direct Known Subclasses:
   AbstractButton, BasicInternalFrameTitlePane, Box, BoxFiller, JColorChooser, JComboBox, JFileChooser, JInternalFrame, JInternalFrameDesktopIcon, 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:
  \[ \text{XXX.add( )}; \]

- Methods for getting an object’s property:
  \[ \text{objectName.getXXX( )}; \]

- Methods for setting an object’s property:
  \[ \text{objectName.setXXX( )}; \]
2. Using Swing Components

How to add button & label

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

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

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

    public static void main(String[] args) {
        Frame2 f2 = new Frame2();
    }
}
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();
    }
}
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

<table>
<thead>
<tr>
<th>Act that results in the event</th>
<th>Listener type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User clicks a button, presses Return while typing in a text field, or chooses a menu item</td>
<td><strong>ActionListener</strong></td>
</tr>
<tr>
<td>User closes a frame (main window)</td>
<td>WindowListener</td>
</tr>
<tr>
<td>User presses a mouse button while the cursor is over a component</td>
<td>MouseListener</td>
</tr>
<tr>
<td>User moves the mouse over a component</td>
<td>MouseMotionListener</td>
</tr>
<tr>
<td>Component becomes visible</td>
<td>ComponentListener</td>
</tr>
<tr>
<td>Component gets the keyboard focus</td>
<td>FocusListener</td>
</tr>
<tr>
<td>Table or list selection changes</td>
<td>ListSelectionListener</td>
</tr>
</tbody>
</table>
3. Events Handling

Requires three pieces of code

- In the declaration
  ```java
  public class MyClass implements ActionListener
  (Note: no need if sub-classed from JFrame)
  ```
- Registers an instance of the event handler class as a listener
  ```java
  someComponent.addActionListener(
    instanceOfMyClass);
  ```
- Implements the methods in the listener interface.
  ```java
  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: invoke the action listener's actionPerformed method
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() {
        public void actionPerformed(ActionEvent e) {
            // Add your action listener code here
        }
    });
}
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();
}

public void actionPerformed(ActionEvent e) {
    jLabel1.setText("Well done!");
}
}

... end
Practice 13-2  Event Handling

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

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

    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).
GUI and Applet / Numerical Algorithm

1. Getting Started with swing
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

```
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 $E = 2.71 \ldots$

double $\pi = 3.14159 \ldots$

int $\text{abs}(\text{int} \ a)$, long $\text{abs}(\text{long} \ a)$

double $\text{sin}(\text{double} \ a)$, double $\text{cos}(\text{double} \ a)$

double $\text{exp}(\text{double} \ a)$, double $\text{log}(\text{double} \ a)$,

...  

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

```java
f.setText ("Component")
f.setBounds(x,y,width, height)
```
Practice 14: “Function Calculator”

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

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

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();
}
/* Construction of the frame */
public Frame4 (){
    jLabel1.setText("f(x) = x\times + ");
    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);
    jTextField1.setBounds(100,20,50,20);
    jTextField2.setBounds(150,20,50,20);
    jTextField3.setBounds(260,20,50,20);
    jTextField1.setText(" ");
    jTextField2.setText(" ");
    jTextField3.setText(" ");
    contentPane = (JPanel)this.getContentPane();
    contentPane.setLayout(null);
    contentPane.setContentPane(mainPanel);
    contentPane.setBounds(20,60,130,20);
    contentPane.setBorder(null);
    contentPane.setTitle("Practice14: Input boxes");
    this.setSize(450,120);  //size of frame
    this.setTitle("Practice14: Input boxes");
}
/* 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”
Practice 14: “Function Calculator”

```java
/* 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

Program:
- Main class: AppFrame5
- extend JFrame -> MyFrame:
- JPanel -> MyPanel:
  - FuncVal class: calculate

\[ f(x) = cx^2 + a\sin(bx) \]

\( a = 0.4, \ b = 8.0, \ c = 0.7 \)
// 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

```java
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

```java
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)
    }
```
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("  "+x2,450,220);   // mark “x2”
        g.drawString(" f(x) = 0.7*x*x+0.4*sin(8*x)",450,80);
}
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 class ends here
} // MyFrame class ends here
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;
    }
}
Exercise 4  (submit by July. 14, 2004)

Based on Practice 14 and Practice 15,
1) Create a GUI to calculate the value of
   \[ f(x) = cx^2 + a\sin(bx) \]
   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();”)

![GUI example](image-url)
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Getting Started with swing</strong></td>
<td>(7) 6/24</td>
</tr>
<tr>
<td><strong>2. Using Swing components:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>frame, button, label</td>
</tr>
<tr>
<td><strong>3. Event handling</strong></td>
<td></td>
</tr>
<tr>
<td><strong>4. More on Swing components:</strong></td>
<td>(8) 7/1</td>
</tr>
<tr>
<td></td>
<td>input-box, plotting</td>
</tr>
<tr>
<td><strong>5. More on Swing components:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>table</td>
</tr>
<tr>
<td></td>
<td>Basic Idea, Error</td>
</tr>
<tr>
<td></td>
<td>Root finding, Integration</td>
</tr>
<tr>
<td><strong>Final Assignment 1 / Questionnaire</strong></td>
<td>(9) 7/8</td>
</tr>
</tbody>
</table>

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

table = new JTable(m,n);

cContentPane.add(jTable1, null)

tableTableModel model = table.getModel();

gContentPane().add(scrollPane,BorderLayout.CENTER);

...
javax.swing.JTable classes

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

```java
JTable()
    Constructs a default JTable that is initialized with a default data model, a default column model, and a default selection model.

JTable(int numRows, int numColumns)
    Constructs a JTable with numRows and numColumns of empty cells using DefaultTableModel.

JTable(Object[] rowData, Object[] columnNames)
    Constructs a JTable to display the values in the two dimensional array, rowData, with column names, columnNames.

JTable(TableModel dm)
    Constructs a JTable that is initialized with dm as the data model, a default column model, and a default selection model.

JTable(TableModel dm, TableColumnModel cm)
    Constructs a JTable that is initialized with dm as the data model, cm as the column model, and a default selection model.

JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm)
    Constructs a JTable that is initialized with dm as the data model, cm as the column model, and sm as the selection model.

JTable(Vector rowData, Vector columnNames)
    Constructs a JTable to display the values in the Vector of Vectors, rowData, with column names, columnNames.
```

Method Summary

- ```addColumn(TableColumn aColumn)```  
  Appends aColumn to the end of the array of columns held by this JTable's column model.

- ```addColumnSelectionInterval(int index0, int index1)```  
  Adds the columns from index0 to index1, inclusive, to the current selection.

- ```addNotify()```  
  Calls the configureEnclosingScrollPane method.

- ```addRowSelectionInterval(int index0, int index1)```  
  Adds the rows from index0 to index1, inclusive, to the current selection.

- ```changeSelection(int rowIndex, int columnIndex, boolean toggle, boolean extend)```  
  Updates the selection models of the table, depending on the state of the two flags: toggle and extend.
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

The table contains the following values:

<table>
<thead>
<tr>
<th>Label1</th>
<th>Label2</th>
<th>Label3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button1</td>
<td>Table1</td>
<td></td>
</tr>
</tbody>
</table>

Choose Column, Input Numbers ...

| -23.5 |          |        |
| 0.56  |          |        |
| 3     |          |        |
| 5.697 |          |        |
| 56    |          |        |

Add Up!  Sum of Column: 41.757
Practice 16: A simple table

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

public class Prac16TableApp {
    public Prac16TableApp() {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        new Prac16TableApp();
    }
}
```
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();

    // Constructor 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

// 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);
}
Practise 16: A simple table

```java
jButton1.setText("Add Up!");
jButton1.setBounds(31, 175, 102, 37);
jButton1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
    }
});
label3.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

// Accepting 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
Numerical algorithm

Basic Idea
Solve a mathematical problem only by repeating a set of simple operations.
Errors in numerical calculation

**Roundoff error:**

\[
\frac{1}{3} = 0.33333333333333333333333333333333\ldots
\]

**Truncation error:**

Consider using long and double instead of int and float, respectively.
Root(s) of $f(x)$: 

Solution(s) of 1D-equation

$f(x) = 0$

Graphically, $x$-intercepts of the curve of $f(x)$
Based on

**Intermediate Value Theorem**, assuming 2 values: \( x_1 \) and \( x_2 \)

\( f(x_1) < 0 \) and \( f(x_2) > 0 \),

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

there must be at least one value \( x_0 \) that falls between \( x_1 \) and \( x_2 \) such that

\( f(x_0) = 0 \).
Finding root by bisection:

\[ 2 + x^2 - (x-1)x^2 = 0 \]
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(" Accuracy : "+accuracy+"\n");
if \((f(left) \times f(right)) > 0\)

```java
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.

```java
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)

// 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);
}
Practice 17: Root finding (Bisection)

Find root by bisection
x-left of interval: 0.0
x-right of interval: 4.0
Accuracy: 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

Stability & Convergency

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

\[
I = \int_{a}^{b} f(x)dx \quad h = \frac{b-a}{n}
\]
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

\[ \int_{a}^{b} f(x) \, dx = \frac{h}{3} \{ f(a) + 4 \sum_{i=1,3,5\ldots}^{n} f(a + ih) + 2 \sum_{i=2,4,6\ldots}^{n} f(a + ih) + f(b) \} \]
Practice 18: Integral (trapezoidal)

```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  
...
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”.

Exercise 5  (Submit by July 21, 2004)
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)
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:

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

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

<table>
<thead>
<tr>
<th>Topic</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Getting Started with swing</td>
<td>(7) 6/24</td>
</tr>
<tr>
<td>2. Using Swing components: frame, button, label</td>
<td></td>
</tr>
<tr>
<td>3. Event handling</td>
<td></td>
</tr>
<tr>
<td>4. More on Swing components: input-box, plotting</td>
<td>(8) 7/1</td>
</tr>
<tr>
<td>5. More on Swing components: table</td>
<td>(9) 7/8</td>
</tr>
<tr>
<td>6. Java Applet</td>
<td></td>
</tr>
<tr>
<td><strong>Final Assignment 1</strong> / <strong>Questionnaire</strong></td>
<td></td>
</tr>
<tr>
<td>7. Mathematical functions</td>
<td></td>
</tr>
<tr>
<td>8. Root finding, Integration</td>
<td></td>
</tr>
<tr>
<td>9. Basic Idea, Error</td>
<td></td>
</tr>
<tr>
<td>10. Monte Carlo Method, Random Number</td>
<td>(10) 7/15</td>
</tr>
<tr>
<td><strong>Final Assignment 2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Closing</strong></td>
<td></td>
</tr>
</tbody>
</table>

*Today* (10) 7/15

---

*CP-II (10), July 15, 2004*
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:**
  
  ```java
  import java.awt.*;
  import javax.swing.*;
  ```

- **Defining an Applet Subclass:**
  ```java
  public class xx extends JApplet {
  }
  ```

- **Implementing Applet Methods:**
  ```java
  init(), start(), paint(), stop(), destroy()
  ```

- **Running/Viewing an Applet:**
  - HTML
    - through a web browser
    - Unix: “appletviewer xx.html”
Life Circle of Applet

First loading
init()

Initial state → Running

revisiting page
start()

revisiting page
start()

leaving page
or quit browser
stop()

leaving page
or quit browser
stop()

Final cleanup

preparation for unloading
destroy()

Final cleanup

- Security Manager: “sandbox”
Including an Applet into HTML

HelloApplet.java

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

public class HelloApplet extends JApplet {
  ...
}
```

compiled "class"

Hello.html

```html
<html>
  ...
  <applet
code="HelloApplet.class"
width=200 height=150>
  ...
  </applet>
  ...
</html>
```
How to write a JApplet

• public class HelloApplet extends JApplet {
  ...
}
• public void init() {
  MyPanel contentPane = new MyPanel();
  getContentPane();
  ... }
• 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

**Constructor Summary**

**JApplet()**

Creates a swing applet instance.

**Method Summary**

```java
protected void addImpl(Component comp, Object constraints, int index)
    By default, children may not be added directly to a this component, they must be added to its contentPane instead.

protected JRootPane()
    Called by the constructor methods to create the default rootPane.

AccessibleContext getAccessibleContext()
    Gets the AccessibleContext associated with this JApplet.

Container getContentPane()
    Returns the contentPane object for this applet.

Component getGlassPane()
    Returns the glassPane object for this applet.

JMenu getJMenuBar()
    Returns the menuBar set on this applet.

JLayeredPane getLayeredPane()
    Returns the layeredPane object for this applet.

JRootPane getRootPane()
    Returns the rootPane object for this applet.

protected boolean isRootPaneCheckingEnabled()

protected String paramString()
    Returns a string representation of this JApplet.

void remove(Component comp)
    Removes the specified component from this container.

void setContentPane(Container contentPane)
    Sets the contentPane property.

void setGlassPane(Component glassPane)
```
import java.awt.*; import javax.swing.*;

public class HelloApplet extends JApplet {
    public void init() {
        MyPanel contentPane = new MyPanel();
        getContentPane().add(contentPane);
        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);
    }
}
<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:
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:

\[ I_{j+1} = (aI_j + c) \mod m \]

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’

$$\frac{V(\text{or } A)_D}{V(\text{or } A)_{D'}} = \frac{N}{N'}$$
Practice 20-1: Estimating $\pi$ by Monte Carlo Method

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

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

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

$$P^0 \left( x^2 + y^2 < 1 \right) = \frac{M}{N}$$

if $N$ becomes very large, $P = P^0$ \implies \pi = \frac{4 \cdot M}{N}$
Create a program to calculate $\pi$ now!
(testing various $n$: large number)

Result:

Calculate $\pi$ by Monte Carlo Method

- $n=10000$, $\pi=3.1388$
- $n=100000$, $\pi=3.1452$
- $n=1000000$, $\pi=3.14164$
- $n=10000000$, $\pi=3.1422784$
- ...
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;
        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");
        }
    }
}
import java.awt.*;
import javax.swing.*;

public class NrandomSwing extends JApplet {
    public void init() {
        MyPanel contentPane = new MyPanel();
        getContentPane().add(contentPane);
        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 $\pi$: calculation + plot
Practice 20-2: Estimating $\pi$ : calculation + plot

```java
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)
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.

\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z}{c}
\]

<table>
<thead>
<tr>
<th>Radius</th>
<th>Height</th>
<th>Total points</th>
</tr>
</thead>
</table>

Volume (analysis):
Volume (Monte Carlo):
Error (%):
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”
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 Practice 13-2, write a program,
1) To display a frame as shown in (a);
2) Clicking the button, to show the number of clicks $N$ and its factorial $N! = 1 \times 2 \times \ldots \times N$, 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
   \( f(x) = cx^2 + a\sin(bx) \)
   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();”)

---

![GUI to calculate and display function](image)
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
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”.

Exercise 5  (Submit by July 21, 2004)
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@c.nl.iis.u-tokyo.ac.jp
Final Assignment 1

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

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

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 an 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.

![Diagram of a paraboloid with radius $r$ and height $h$.]

Radius [ ] Height [ ] Total points [ ]

Volume (analysis): [ ]
Volume (Monte Carlo): [ ]
Error (%): [ ]

$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \frac{z}{c}$
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'.