Programming I: Getting started

Programming I: Getting started

Lecture Contents

Some definitions (hardware, software, etc.).

What is programming?

What is Java?

Writing your first program

Processor

- Microprocessor, CPU, chip.
- The computer hardware that executes program instructions (machine code).
- Intel Pentium™, Sparc, Motorola PowerPC
  - Each has its own specific *instruction set*.

Software

- Any computer instructions written to be executed on *hardware*.

- Hardware is touchable, software is untouchable.

- Two main categories of software
  - Systems software (for the computer)
    - operating systems,
    - utility programs,
  - Applications software (for users)

Operating Systems

- The operating system controls the computer.
- e.g. DOS, Unix, Linux, Windows NT, etc.
- Systems Programming = Process of developing *operating system* software.

Applications

- A program that does something useful for the *end user*.
  - Word processor, spreadsheet, Quake IX, etc.

- Application programming is the process of *developing* applications.

(Software) Programmer

- Person who writes programs!

- Responsible for identifying the correct sequence of instructions required and writing them down.

Program

- A sequence of instructions carried out by a computer (or *run*, or *executed*).
- The sequence is typically long and complex.
- Running a program will result in millions or billions of instructions being executed.
- The instruction sequence has to be correct or the program will fail.

A *very* simple program!

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

Programming

- The process of writing programs.
  - Requires 5 important steps:

With a pen and paper!

- Analysis - what is the program meant to do?
- Design - how is the program structured?

On a computer

- Coding - writing the program code.
- Debugging - fixing errors.
- Testing - making sure the program does the right thing.

Programming Languages

- A *textual* language used to write a program.
- *Java* is a programming language.
- *Implementation*: writing a program.

Syntax

- Syntax describes the *grammatical rules* of a language.
  - Valid words.
  - Punctuation.
  - Sentence construction.
  - Rules of use.
- Programs must be syntactically correct.

Semantics

- Semantics give the *meaning* of what you write with a language.
- A program must be semantically correct to do what you expect.
- A programming language must precisely define the meaning of every *statement* that can be written with it.

Syntax v. Semantics

“Catch never ball the my”

Grammatically incorrect.

“This red is an elephant.”

Grammatically correct, but no sensible meaning.

“Seven is the highest number.”

Grammatically correct, sensible meaning, but false.

A Program!

Ordinary text written with an editor. Called *Source Code*.

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

Obeys *syntax* rules & *semantics* of Java.

Compilation

- A computer cannot directly understand or run source code!
- A translation from source code to processor instructions has to be performed.
- This is known as *compilation*.

The Compiler

- Fortunately, a tool called a *Compiler* can translate the text to processor instruction.

- The compiler knows and checks
  - all syntax rules
  - only some of the semantics.

- The compiler is itself a program.

Starting to Program

- We want to get you started as soon as possible!
- But at first you will have to take a lot of things on trust.
- We will return to the details later on.

Writing a Java Program

- Analyse the problem to solve, with paper and pen!
- Sketch an algorithm, with paper and pen!
- Use an editor to type in or edit the program source code
- Save the code to a file
- Compile the file with the Java compiler
- Run the program and see what happens.
- Fix the *bugs*!

Hello

A comment

```
// This is our first program!
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello World") ;
    }
}
```

Our program will be called Hello.

Save in a file called Hello.java

Structure of ‘Hello’

- The whole program is a class definition.
- This <u>can</u> be called anything.
- But it is best if the name is consistent with the effect of the program.

  (“Hello” in this program).

  This is true for all classes, variables and methods.
- A special method called `main()` is the entry point of the program. This is a name that you can’t change. It is executed first and has a fixed signature.

Comments

- Comments are very important
- A source code without comments is unreadable and confusing

Different types of comment

Java documentation

Special comments:

```
/**
        Text
*/
```

- Name of authors, date
- Name and goal of the program
- General definition of a class
- Specification of a method

Usual comments

```
//Text
```

- Explanation of the action of a group of sequences

Example

```
/**
 * <dl>
 * <dt>Purpose: First and very simple program to test java.
 * <dd>
 *
 * <dt>Description:
 * <dd> This program displays hello world on a command window.
 *
 * <dl>
 *
 * @author  Danny Alexander
 * @author  Céline Loscos
 * @version $Date: 2002/09/13$
 *
 */
public class Hello
{
     public static void main(String[] args)
    {
       // Display ‘hello world’ on the screen
       System.out.println("Hello World") ;
    }
}
```

Java Compiler

- The Java compiler is called *jikes* or *javac*
  - To compile use:

```
javac Hello.java
```

Java source code file names must always end with .java

This will create a file called Hello.class

Running a Java Program

- To run a Java program use the command *java*.

```
java Hello
```

Name of the program you want to run.

This is the Java *interpreter* which actually runs the program.

The Java Virtual Machine (JVM)

- Java programs are compiled to *bytecodes* for a *virtual processor*.
- The command java runs the JVM which simulates the virtual processor.
- So your program is run by the JVM which is, in turn, run by the real processor.

Why the complication of a VM ?

- A Java program can run on any real processor that can run the JVM.
- “Compile once, run everywhere”

Origins of Java

- Designed by a group at Sun MicroSystems.
- Originally called Oak - a language for programming consumer devices.
- Renamed to Java and moved to the web.
- Developed into a full scale application programming language.

Summary

- Defined some basic terms.
- Introduced the ideas of a programming language and compilation.
- Seen how to write, compile and run small programs.

Course work notes

- Hand-in deadline: noon, Friday, 10th October.
- Hand-in point: CS reception.
- Only Q1.12 is to be handed-in.
- All Qs to be worked on in lab sessions.

Drawing Shapes

- A number of exercise questions ask you to write programs that draw shapes.

- You are given a complete program which you can copy and then edit.

Program meaning

- At the moment you won’t understand most of the program!!

- Don’t worry.

- Read the comments for guidance.
  (The lines starting with //)

What do you change?

```
// This part of the program does the actual drawing.
public void paint(Graphics g)
{
     // You add/change the statements here to draw
     // the picture you want.
     // For example, draw a diagonal line.
     g.drawLine(0,0,300,300) ;
}
```

Make changes here. See the notes.

Object

- Q: What does `g.drawLine()` mean? What is `g`?
- A: `g` is a *reference* to an *object*.
- The object knows how to draw.
- This *will* become clear later in the course!

Anything else to change?

- Yes, the name of the program.
- See the line that says:

```
class DrawingLine extends Panel
```

Here's the name. Change it!

- Save the program to a *new* .java file using your new name.

Yet more to change?

- Yes.

- Wherever you see the name `DrawingLine` in the program, replace it with the new name.

```
DrawLine drawing=new DrawLine();
```

Here and here

Drawing something different

```
// This part of the program does the actual drawing.
 public void paint(Graphics g)
 {
    // You add/change the statements here to draw
    // the picture you want.
     g.drawRect(150,150,50,50) ;
     g.fillRect(20,20,50,50)  ;
 }
```

New lines of code.

The order of the lines give the sequence by which the picture is drawn.

What else can you draw?

- `g.drawArc(x,y,w,h,startAngle,arcAngle);`
- `g.drawLine(x1,y1,x2,y2);`
- `g.drawOval(x,y,w,height);`
- `g.drawRect(x,y,width,height);`
- `g.drawString(text,x,y);`

A String is a line of text.

Drawing a more complicated picture

- Work out how to draw a complicated picture by drawing it using a series of simpler shapes.

= Problem decomposition.