UCL

COMP1007
Principles of Programming

UCL

Agenda

- Definitions.
- What is programming?
- What is Java?
- Writing your first program.
- Classes and Objects.

UCL

Reading

- You should be reading chapters 1 & 2 of the text book.

UCL

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.

UCL

Programmer

- Person who writes programs!
- Responsible for identifying the correct sequence of instructions required and writing them down.
- Who you want to be :-)

UCL

Application Program

- A program that does something useful for the end user.
  - Word processor, spreadsheet, web browser, etc.
- A tool to perform tasks to achieve a goal.

Systems Programming

- An operating system controls and manages a computer.
  - DOS, Unix, GNU/Linux, Mac OS X, Windows XP, etc.
- Systems programming is the process of developing operating system software.
  - And supporting tools.

Software Engineering

- The process of developing programs.
- Involves:
  - Requirements – what is the program meant to do?
  - Analysis – how should the program behave?
  - Design – how is the program structured?
  - Coding – writing the program code.
  - Debugging – fixing errors.
  - Testing – making sure the program works.
  - Deployment – putting the program into use.
  - Maintenance – keeping the program working.

Processor

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

Programming Language

- A textual language used to write a program.
  - Read/written by programmer.
  - English-like appearance (sort of).
- The meaning of what you write has to be completely and precisely defined.
- Java is a programming language.

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

"This sentence is an elephant."

- Grammatically correct but no sensible meaning.

"This sentence is false."

- Grammatically correct but logically inconsistent.

Nonsense programs...

- A program can be syntactically correct but not do anything useful or sensible.
- However, what a program does can always be exactly determined.

Is that actually true?

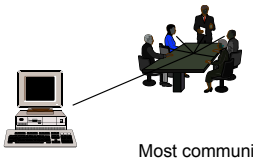
Determinism v. Non-Determinism

- What a deterministic program does can always be known.
- The behaviour of non-deterministic programs can't (easily) be predicted reliably.
- Your programs should be deterministic!

But non-deterministic programs can be written with Java.

Communication

- Programming involves a lot of communication.



Most communication occurs between people (or with yourself!)

Communication skills

- Speech, writing, drawing, diagrams, etc.
- To talk about the design of a program with other people is hard and you need to be quite precise.
- To describe a program to a computer you have to be absolutely precise.

Readability

- Programming languages are for people to describe programs.
- The text of a program should be written for other people to read.
  - Think about what makes a book, newspaper or website easy to read and understand.

Questions?

Writing a 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.

A Program!

Ordinary text written with an editor. Called Source Code.

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

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 do the text to processor instruction translation.
- The compiler knows and checks all syntax rules but only some of the semantics.
- The compiler is itself a program.

Writing a Java Program

- 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.
- Remove the bugs!

Fix syntax errors

Fix semantic errors

Hello

A comment

```
// Say hello!
class Welcome
{
  public void sayHello()
  {
    System.out.println("Hello World");
  }
  public static void main(String[] args)
  {
    Welcome welcome = new Welcome();
    welcome.sayHello();
  }
}
```

Our *class* is called Welcome.

Save in a file called Welcome.java

Which editor?

- There are several choices:
  - JEdit
  - BlueJ
  - Emacs (or XEmacs) + command line
- BlueJ or JEdit are preferred but Emacs is also very powerful.
- Experiment.
- Take the time to learn to use your editor effectively.

Java Compiler

- The Java compiler is called javac.
  - To compile use:
    javac Welcome.java

Java source code file names must always end with .java

This will create a file called Welcome.class

Running a Java Program

- To run a Java program use java.
  - java Welcome

Name the program you want to run.

This is the Java interpreter which actually runs the program.

Hello World appears in xterm.

Let's Try This

- Cue the demos...
  - JEdit + command line

Being efficient

- Use one or more xterm windows for typing commands.
- Don't close the editor every time you edit a file, simply save the file.

Remember

- Use JEdit to type and edit source code.
- Compile program via xterm window, or
  - Use the JEdit Console.
- Run the program via xterm window, or
  - Use the JEdit Console.

Being even more efficient

- Typing !!<return> repeats the previous command.
- This means you can type:
  javac Hello.java
  once.
- And repeat it using !!
- Works for any command.

Being yet more efficient

- The command history will display a numbered list of the past commands you have used.
- !5 will repeat command 5, !10 command 10 and so on.
- !ji will repeat the last command starting with the letters ji

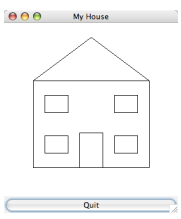
Questions?

Drawing Shapes and Pictures

- A number of exercise questions ask you to write programs that draw pictures.
- You are given the source code of a complete program to copy and edit.
  - Another class.

An Example Dawing

Made up of lines and rectangles.



Let's see the program

- OK, another demo

How do you draw?

```
// This part of the program does the actual drawing.
 public void doDrawing(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.

A Drawing Object?

- What does g.drawLine() mean? What is g?

- g is a reference to an object.
- The object knows how to draw.
- You tell it what to draw.

Your own drawings

- Select a new class name.
  - The name of your drawing.
- Copy the template file and save as <yourname>.java
- Make the changes described next.
- Compile, run.

Changes

- Change the name of the class.
- See the line that says:
- class Drawing extends JFrame

Here's the name. Change it!

- Don't forget: save the class to a new .java file.

Yet more to change?

- Yes.
- Wherever you see the name Drawing in the program, replace it with the new name.
- Drawing drawing = new Drawing("MyDrawing");

Change these as well.

I wonder if the editor does search and replace?

Drawing something different

```
// This part of the program does the actual drawing.
 public void doDrawing(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, width, height, startAngle, arcAngle);
- g.drawLine(x1,y1,x2,y2);
- g.drawOval(x,y,width,height);
- g.drawRect(x,y,width,height);
- g.drawString(text, x,y);

A String is a line of text (sequence of characters).

Drawing a more complicated picture

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

Questions?

A Closer Look

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

Infrastructure

Statement we want to execute.

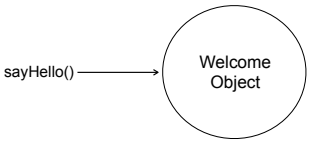
Infrastructure

Infrastructure

- Necessary to support the code we actually want to run.
- Its full purpose will become clear as the course proceeds.
- For now “cut and paste” job.
- But will elaborate a bit…

What happened?

- The program created an object.
- The object was asked to say Hello.



Class and Object?

- Our program actually consisted of a single *class declaration*.
- The class is a "template" that describes what a Welcome object is and does.
- Running the program creates the object and asks it to sayHello.

Objects

- Have responsibilities
  - to carry out actions (e.g., say Hello)
  - to know things (e.g., what to actually say)
- Can collaborate with other objects to perform more complex tasks.
  - Our Welcome object actually uses another object, System.out, which has the responsibility of directing text to the screen.

Objects with BlueJ

- BlueJ is another programming tool you will be using.
- It allows you to directly manipulate objects.
- Here is the demo…

Questions?

Huh?

- These programs seem more complicated than necessary!
- Why not just a simple statement on its own?
  - print("Hello World");

Because…

- The programming language works this way.
- Any non-trivial program needs structure to have any chance of being manageable.
- Classes and objects provide that structure.
- You need to learn to do things properly from the start!

And…

- Classes and objects provide the components, or building blocks, to construct a program from.
- Program statements provide the detail describing how objects perform operations.
  - Like saying Hello.
- Think about levels of detail, or abstraction.

But!

The program could have been written like this:

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

Why not?

Well…

- We want to emphasise *objects* from the start.
- The simpler program form is only of any use for very small examples.
  - Chap. 2 of the text book uses the simple form but you should use objects from the start!

Before we finish…

- If the compiler translates to processor instructions, then what is the java interpreter doing?
- You need to ask: which processor?

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 that is, in turn, run by the real processor.

Why?

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

So, where did Java come from?

- Designed by a group at Sun Microsystems Inc.
- Originally called Oak - a language for programming consumer devices (Talkie Toaster!).
- Renamed to Java and moved to the web.
- Developed into a full scale application programming language.
- Continues to evolve.

James Gosling and the Duke Mascot

Find out more

- Visit the official Sun Java web site:
- java.sun.com
  or
- www.java.net

Summary

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