CS1PR1: Introduction to Programming

http://www.dcs.kcl.ac.uk/local/teaching/units/cs1pr1/

July 27, 2002

- Why Program?
- Why Program in Java?

Textbook: K. Lano, J. Fiadeiro, L. Andrade, "Software Design in Java 2", Palgrave, 2002.

Also useful: Cay Horstmann, "Computing Concepts with Java 2 Essentials", 2nd Edition, Wiley, 2000.

Webpage (http://www.dcs.kcl.ac.uk/local/teaching/units/cs1pr1/) contains copy of all code used in lectures + slides, etc.

Why Program?

- Starting salaries in IT range from 20,000 to 28,000+
- Programs/software essential in many areas, from mobile phones, internet, through to aeroplane and car control systems
- A challenging but rewarding skill – building a significant program (eg, 2nd year projects last year constructed an internet-based course registration system) is substantial achievement

Why Program in Java?

- Java is rapidly becoming most widely used general purpose programming language in industry
- High quality language – object-oriented structuring, good support tools, etc
- Can quickly build sophisticated graphical interfaces and 'applets' (programs which run in a web browser)

Java originally began as language to add animation, dynamic content, etc to web pages – but now used in every field. Still closely linked with internet programming however (see later CS1 and CS2 courses).

Programming Fundamentals

- A program is usually written to solve a particular information management or device control problem
- Example: recording details of undergraduate students and their exam performances so Dept. can make decisions on progression, references, etc
- System should manage set of student data (name, year, set of courses sat with results) for each student, allow addition of data, printing, etc
- Data storage functionality could be done using databases like ACCESS, but complex calculations (eg, estimating students final degree grade) better done within a program.

Programs manipulate data which represent real-world entities, in order to help us manage these entities.

The Course

- How to write computer programs (using object-oriented techniques)
- Introduction to Java: applets, methods, arrays, inheritance, etc

More advanced capabilities of Java covered in future courses (CS1DST, CS3GRS, CS2SAP, etc): networking, web servers, 3D graphics, etc.

Essential book: K. Lano, J. Fiadeiro, L. Andrade, "Software Design in Java 2", Palgrave, 2002.

Other texts: C. S. Horstmann, Computing Concepts with Java 2 Essentials (2nd Edition), Wiley, 2000. Winder and Roberts, Developing Java Software (2nd Edition), Wiley, 2000.

Lectures and Courseworks

Two lectures each week: Tuesday 12.00 (11F), Tuesday 14.00 (11F).

Each fortnight courseworks are given (simple Java programming problems). Must be handed in by Friday 4pm. Plus mini group project.

At least 5 out of these must be passed for you to pass course!

Courseworks can be done in **Laboratory Sessions** (2 hours each week), with demonstrators on hand to answer your questions.

** Hand-in is to the coursework office/pigeonholes, Mon/Wed/Fri 2pm-4pm. ** (may change)

Students have been assigned to lab groups based on their tutor.

Doing the Practical Work is Important!

- Best way to learn programming is to practise writing programs
- Even if early courseworks seem very easy, still need to do them to pass course. Also helps you get used to programming environment here.
- Java together with its library is a huge language designed for professional programmers. Every Java programmer is somewhere along path from beginning to the end. (In fact, it is a growing path because Java library is continually being extended.) Work hard and patiently, and you will make steady progress.

Computer Components

- A *memory* in which information of all kinds is stored. The information can be numbers, text, photographs, music, etc. These items can be erased and replaced by new information.
- Devices for reading and writing information. In particular a keyboard which a user can use to input information into memory. Also a screen on which computer can display items of information held in memory
- A *processor* which operates under the control of a *program*. The program describes a sequence of operations which are to be carried out by the processor. These actions include things like reading, writing and modifying information in the memory, carrying out calculations with values in the memory, etc.

Programming Languages and Java

- Java is a 'High level programming language', ie, intended for humans to read and write, as opposed to machine code (for direct execution by a computer).
- A Java program file $ff.java$ is *compiled* by the Java compiler (here, we use $javac$) into machine code files $ff1.class$, $ff2.class$ (in fact we get a machine code file for each class defined in $ff.java$).
- Java is an object-oriented language, like C++. This means programs consist of class definitions and execution involves the application of *methods* (operations/functions) to *objects* of these classes.

Java Program Structure

Java program files consist of a list of *class* definitions, each class definition has a list of local data declarations and *method* definitions. Each method definition consists of a method *header* and a method *body.* A method body consists of a list of *statements.*

```
class Name
{ <definitions of local data>

  <method definition 1>
  ...
  <method definition n>
}
```

At least one class in each file must be declared as a `public class`. Name of file must be same as this class.

Very Simple Java Program

```
public class HelloWorld
{  /* Writes a simple message on the screen. */
   public static void main(String[] args)
   {  System.out.println("Hello, World!");
   }
}
```

This could be written in a file *HelloWorld.java*. It consists of one class definition, which has a comment (between `/*` and `*/`) and a single method definition inside it. In turn, this method has a single statement in its body:

```
System.out.println("Hello, World!");
```

When this statement is obeyed, the message `Hello, World!` will appear on the screen (in the DOS window that we executed it in). Program terminates at end of *main* method.

To run program, type:

```
javac HelloWorld.java
```

to compile it, then, if no compilation errors occur, type:

```
java HelloWorld
```

in DOS window to execute the compiled code.

The method *println* "print line" always adds a new line after the text it displays.

Another Very Simple Java Program

```
public class HelloWorld
{  /* Writes a simple message on the screen. */
   public static void main(String[] args)
   {  System.out.println("Hello, World!");
      System.out.println("See you later.");
   }
}
```

Prints `Hello, World!` on one line, `See you later.` on the line beneath, then terminates.

We could put as many statements as we need in the *main* method. Each statement must end with a ';'.

Objects and Methods

Java provides many built-in objects. *System.out* is an object – it represents the output display for the program.

But strings such as "Hello, World!" are also objects (of class *String*).

Java programs deal with objects and also 'primitive values' such as integers and real numbers: 165, 45.565, etc.

We can also define our own new classes (like $HelloWorld$ above) and create objects of these classes. For example in a program to hold course registration data, would have classes $Course$ and $Student$, at least.

Example of Method Invocation

```
System.out.println("Hello, World!");
```

This statement (remember, method bodies consist of a sequence of statements) is an application of the method *println* to the object *System.out*.

A class lists all the methods that its objects can respond to. *HelloWorld* responds to only one method *main* – this method is automatically invoked when a Java program is started (by *java HelloWorld* on the command line).

Note: *print* method doesn't add a new line at end of string.

Example of Method Definition

```
public class HelloWorld
{
   public static void main(String[] args)  /* Method header. */
   {  System.out.print("Hello, ");         /* Method */
      System.out.println("World!");        /* body.  */
   }
}
```

Header gives information on how method can be used ('public' means any other class can invoke it, for example) and name *main* of method, plus parameters *args* and their type *String*[] – array of *String*.

All this means is that a Java program can accept a list of command line arguments, like *copy* in DOS.

Method body is between curly brackets { and }. A list of statements.

Coursework 1

Hand-in: Friday 4th October, 4pm. Room 539, Strand.

1 Type in, compile and run the following *HelloWorld* program that outputs two lines of output. Use a different message if you prefer.

```
public class HelloWorld
{  /* Output a simple message on the screen. */
   public static void main(String[] args)
   {  System.out.println("Hello, World!");
      System.out.println("See you later.");
   }
}
```

When you have finished, print out your program file, *HelloWorld.java* and hand it in.

2 Write, compile and run a program which displays a 'heart' symbol using the |, \ and / characters, eg:

```
/\/\
|   |
\   /
 \/
```

Print out your program file and hand it in.

How to do the coursework

Suppose you want to run the *HelloWorld* program.

You must go to a PC. If you are working in room 7D or 24D it will be part of the Department's PC Network, and you must log on. Then you will carry out the following steps.

1. Type the program and store it in a file called *HelloWorld.java*.
   Use a text editor program to do this, such as Notepad. Alternatively, you can use the text editor called *edit* by typing command *edit* followed by the file name, in the DOS window. Word is more trouble than it's worth for program editing!
   Normally, if your program is made up of a single class, it will be stored in a file that has same name as class, followed by extension '.java'. Since the program consists of a class called *HelloWorld*, it will be stored in a file called *HelloWorld.java*.
   When you have written the program, do not close the

   Notepad/editor window. You may want to return to it later to make changes to the program.

2. Open a DOS window and move to folder where you have put program file. All the commands you are going to input will be typed in this window, and the message produced by the program will appear in the window too.
   You now have two windows open: the Notepad window containing the program, and the DOS window for communicating with the Java Software Development Kit (also referred to as the Java Development Kit, or JDK).

3. If you are working on the PC Network, there is an initial step to carry out before you start using the JDK system. You must type the command

   `usejdk`

   in the DOS window. If you are working on your own computer at home using a copy of the JDK you have downloaded from the Internet, you may not need to type the usejdk command.

4. To compile the program type

   `javac HelloWorld.java`

   If there are no mistakes in your program, the compiler will produce a bytecode version of it in a file called $HelloWorld.class$.

   If there are any compiler errors, e.g. you have typed Public instead of public or missed a semicolon, compiler will refuse to compile program. It will give error messages.

   If any error messages are given, you will have to correct the program. To do this, click on the Notepad window and make the necessary changes to the program. Then save the program so that the file HelloWorld.java contains the corrected program.

5. Get the Java Virtual Machine to run the compiled bytecode

program in the file HelloWorld.class by typing

`java HelloWorld`

Do not type the .class extension. The JDK will take it for granted that you mean HelloWorld.class.

The program should run and produce its message in the DOS window. You may want to modify the program at this point. If so, go back to the Notepad window to make the necessary changes. Then save the program, compile it, and run it again.

Computer-Aided Learning Tool

This takes you through simple exercises in each of the topics of the course.

It is located in: K:/web/local/teaching/units/cs1pr1/CAL and consists of the *JavaTutor* program. To get your own copy, copy the *class* files to a directory in your own file system. Create an *output* directory in the same directory.

To run the CAL, type: *java JavaTutor*.

Summary of what we've covered:

- Java is one of the most popular languages in industry, and is relatively easy to learn (compared to C++, for example).
- Java program files `ff.java` are compiled into object code files `ff.class` by Java compiler, can then be executed on any machine with a bytecode interpreter.
- Java program files have a structure which is a list of class definitions, each class definition has form:

```
class name
{ <definitions of local data>

  <method definition 1>
  ...
  <method definition n>
}
```

- Each method definition consists of a header giving method name and parameters, and method body, a sequence of statements.

- All statements end with a semicolon ';'.
- One kind of statement is a method invocation: `object.method(arguments);`
- All methods operate on objects.
- `System.out` is object representing output display of program. `println(s)` is a method that operates on it to output a string `s`.