CSCU9A2
Programming and User Interface Design

- Lecturers:
  Dr Simon Jones - Java Programming, Course organizer
  Dr David Cairns - User Interface Design
- Topics:
  - Continuing Java – algorithms, data structures, GUIs and introduction to OO
  - User Interface Design - The principles of good, useable design… for programs, for the Web, for the disabled
- Resources on the Web:
  Succeed, or http://www.cs.stir.ac.uk/courses/CSCU9A2/
- Campus network: My Computer\Groups on Wide\CSCU9A2
- Email: Remember to check regularly
- Computing Science Advisory Team: see Web page:
  http://www.cs.stir.ac.uk/courses/advisers.html

Organization

- Detailed schedule:
  - Accessible via the CSCU9A2 Web page

Home page

- Three lectures per week
- Two 1 hour practicals per week
  - Starting on Monday 23 January
  - Worksheets and checkpoints
  - BRING LECTURE NOTES/TEXTBOOK
- One 1 hour tutorial per week
  - Starting Monday 30 January
  - Problem sheets given out in previous week
  - Attempt problems before the tutorial
- Practical and tutorial sign-up on Succeed
  - Need to change? DIY
- Difficulties with clashes? See/email Course organizer

Assessment

- Achieving “checkpoints” in practicals
  - One checkpoints in (most) practical worksheets
  - Worth 20% of the final grade – IMPORTANT!
  - Checkpoints to be completed ***no later* than the end of the second week following**
- One Java programming assignment: an interactive GUI application
  - Worth 40% of final grade
  - Not attempted: No Mark for the module
- One two hour exam
  - Worth 40% of final grade
  - Not attempted: No Mark for the module
- To get a view of where we are going, take a look at last year’s assignment:

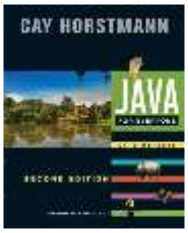
Demo

Software

- You will be using the "public domain" Java Development Kit (JDK):
  - Downloadable free from Oracle (was Sun Microsystems)
  - Not "user friendly" – will be the "engine under the hood"
- The main tool will be the *BlueJ* Integrated Development Environment (“IDE”)
  - A user friendly ‘front end’ for the *JDK*
  - BlueJ is also “public domain”, free for non-commercial use
  - Good for novices, not so helpful for advanced users
- Both are available from our divisional server via the University campus network, see: http://www.cs.stir.ac.uk/courses/software.html

Try JCreator,
Eclipse, NetBeans

The Java Book



- "Java for Everyone: Late Objects",
  2nd Edition, Cay Horstmann, John Wiley
  - Will continue to be useful
- Only some lectures are organized around the chapters of the book - see schedule
- It is excellent reference material
  - You need reference material beside you
- We won't be covering all the book
- It is not specific either to the PC, nor to the Java JDK nor to BlueJ
- There is now a 3rd edition *as an e-book only – very recent*
  - We will not be referring to this

The Java part of the course is about Programming

- Definition from the Shorter OED: (paraphrased)

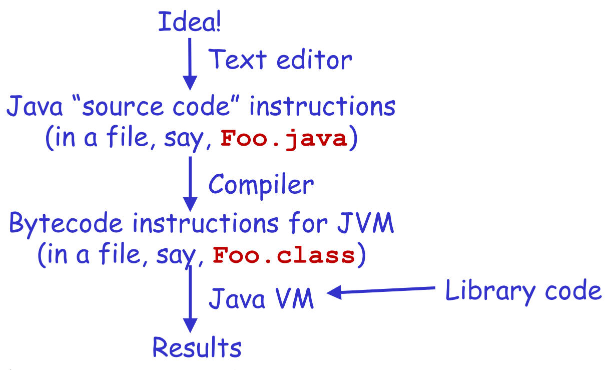
  "Programming: Preparing a fully explicit series of instructions (a program) which when fed into a computer will automatically direct its operation in carrying out a specific task." (1946!)
- Unpacking this, there are many important phrases:
  - "fully explicit"
  - "series of instructions"
  - "automatically direct"
  - "specific task"
- Important to keep these in mind when understanding the *activity* of *programming*
- The *programming language* in which we will be expressing our instructions will be **Java**

About Java

- Java is:
  - a "high level" programming language
  - an *imperative* programming language ("command" oriented)
  - an *object-oriented* programming language
  - a core language plus extensive *libraries* containing facilities for: *graphical user interfaces, communicating over the Internet, interacting with databases, mobile phones, …*
- Java is a member of the C family of languages:
  - C → C++ → Java
- It is also a full industry-strength programming language
  - We must tread carefully through the complexity!

Source Code to Running Program

- High level languages must be compiled (translated) for the computer
- The compiler also makes consistency and validity checks
- The Java compiling scheme

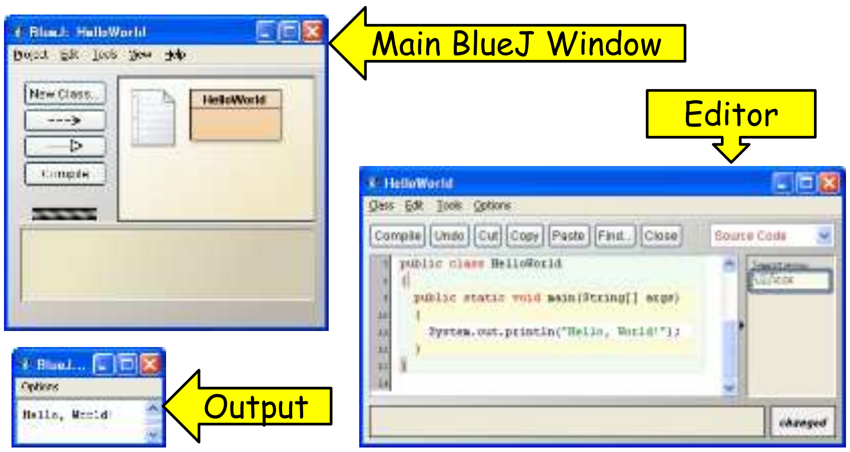

Idea!
Text editor
Java "source code" instructions
(in a file, say, Foo.java)
Compiler
Bytecode instructions for JVM
(in a file, say, Foo.class)
Java VM
Library code
Results

- BlueJ help us to organize these steps

The BlueJ Programming Environment

- To simplify the process, programs are usually created using an Integrated Development Environment (IDE)
  - In CSCU9A2 we use the BlueJ IDE
  - (Later modules use the Eclipse IDE)
- Components of an IDE:
  - Source code editor helps programming by:
    - Showing line numbers of code
    - Colouring lines of code (comments, text…)
    - Automatically formatting source code
    - Inserting coding templates (limited in BlueJ)
    - Continuously checking source code (not BlueJ)
  - Output window
  - Debugger

The BlueJ IDE



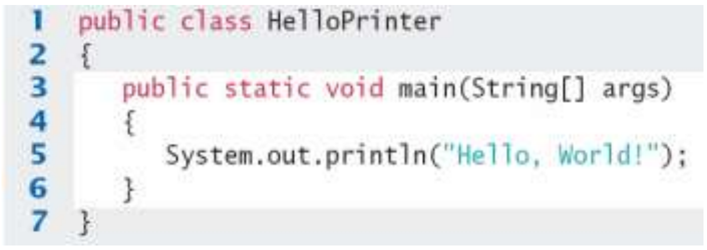
```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

```
Hello, World!
```

BlueJ was designed to help students to learn Java.

A First Program

Recall the main parts of the traditional ‘Hello World’ program in Java



```
1 public class HelloPrinter
2 {
3    public static void main(String[] args)
4    {
5       System.out.println("Hello, World!");
6    }
7 }
```

Neither line numbers nor the colouring are part of Java

Be careful of spelling

JaVa iS CaSe SeNsItiVe

Java uses special characters, e.g. { } ( ) ; in a very precise way

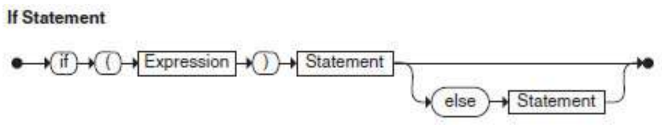
Other topics we will look at

- We will look at how formal "syntax rules"/"diagrams" can be used as a reference for the details of a programming language, e.g

```
identifier ::= letter {letter | digit}
```

OK: `x` `count` `A1b2C3` `x1234`

But not: `123x` `hello-there`



(from http://markettorrent.com/topic/9359)

```
if ( x == 3 + y )
{
  System.out.println( "Result!" );
}
```

- And we will look at how high level programming constructs can be implemented at the machine level, e.g a simple "assembly language" translation of an assignment statement:

```
MOV [a] -> R1
MOV [b] -> R2          for    a = a + b;
ADDI R1, R2 -> R3
MOV R3 -> [a]
```

  where **a**, **b** represent *memory addresses/variables*

  and **R1**, **R2**, **R3** are *CPU registers*

  (we will simplify, and ignore that Java compiles to bytecode)

- To illustrate this, we will use a simple hypothetical computer called the Brookshear Machine
  (from Computer Science: An Overview by J. Glenn Brookshear)
  - Next lecture…

End of section