# CS 108 Teaching Staff

**Lecturer:** **Charles Theva**

**Email:** **charles@cs.rmit.edu.au**

**Subject Home Page: (find subject guide there)**

http://goanna.cs.rmit.edu.au/~charles/cs108/

**Head Tutor:** **Craig Hamilton**

**Email:** **craig@cs.rmit.edu.au**

**(Expert in WebLearn)**

**Problems with Tutes and Labs ?**

**Contact Tutor then Head Tutor then Lecturer then Year Coordinator.**

1

# CS - 108

**One of the most important subject in CS as it is prerequisite for many others.**

**A very practical subject requires at least 5 - 8 hours of programming per week.**

**Assessment** **1.** **50% Exam**

**2.** **50% Coursework**

**4 Assignments** **28%**

**2 WebLearn Tests** **10%**

**Mid-semester Test** **12%**

**Note: You need to pass both components ( 1 and 2).**

2

# Student Notes / Book

**Prescribed**: ***CS108 Student Notes*, Department of Computer Science, RMIT. Volumes I and II.**

**Cay Horstmann, *Computing Concepts with Java 2 Essentials*, 2nd Edition, Wiley, 2000, ISBN 0-471-34609-8.**

**(Likely to be Used for both semesters)**

**Recommended Reference: Lewis J., Loftus W., *Java Software Solutions*, 2nd Edition, Addison Wesley, 2000 , ISBN 0-201-61271-2**

3

# How can I do well ?

**Attend all Lectures, Labs and Tutes**

**Read the Subject Notes before going for Lectures. Review it afterwards.**

**Start your assignments early - it always takes longer than you think.**

**Complete all the lab exercises even if you're not required to submit them.**

**Read the text book**

**Start Writing Programs in week 1 itself !!! Those who start early always do well.**

4

# CS108 Syllabus

1. Introduction, programming languages, OO
2. Variables, identifiers, Operators and precedence.
3. Algorithms and Stepwise refinements. Decisions.
4. Repetitions
5. Writing simple classes
6. Arrays, methods and parameters, Applets
7. Inheritance and Polymorphism
8. More modifiers: static and final.
9. Arrays Strings revisited.
10. Exception handling
11. Buiding a Generic Container class
12. File manipulation
13. Enhanced class design

5

# Important Dates (in the Subject Guide)

**Week 1 starting 26/02/2001**

**Intro to WebLearn**

**Week 3 starting 12/03/2001**

**Chapter 2 assignment must be submitted**

**Week 4 starting 19/03/2001**

**First assignment submission due Mon 19/03/2001, 9:30pm.**

**Last week to pick a group for your test.**

**Week 5 starting 26/03/2001**

**Test Week (WebLearn Test 1)**

**Chapter 4 assignment must be submitted.**

**…..**

6

# Web Learn

7

# Origins of Java

- **Originated as part of a Research Project**
- **To build a language/environment for small electronic devices**
- **Embedding networking capabilities and Cross platform portability**
- **Originally called Green, then changed to Oak, finally Java**
- **Design decisions drew from C++, Eiffel, SmallTalk**
- **Launched as an Internet extension for browser early 1995.**

8

## Java's main Philosophy

- **Simple**
- **Familiar to Developers**
- **Object Oriented**
- **Portable**
- **Multi-threaded**
- **Robust and Secure**

9

# Machine language

**Languages that a computer can directly understand. Differs from Machine to Machine.**

**Example:**

```
0010 0000 1100 0100
0011 0111 1100 0100
```

10

## Assembly language

- a level higher than machine languages
- but must be specified in terms of basic operations such as moving data into registers.

```
LOAD Hcost
MOVE Tax
ADD  Profit
```

11

## High- Level languages

- Programs specified in an English-like syntax, and a compiler is used to translate it.
- Each compiler produces a translation for a specific computer.
- Same source program can be run in another computer if it is compiled by the appropriate compiler.

12

# Divisions of High- Level Languages

- Procedural - C, Pascal
  divided into procedures
- Object-oriented - Java, C++, Small
  based on interaction of objects
- Logic languages - prolog
  setting up goals

13

# Traditional Steps:
## Compiling Linking and Executing

**Standard way of producing and creating an application**

**1.Create the program.**

```
Int add(int x, int y)
{ int z = ...
```

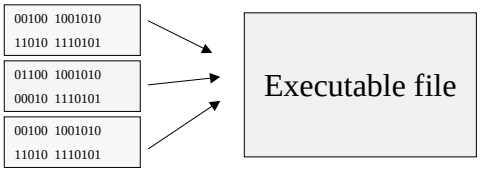
**2. Traditional compilers translate source code into machine language directly.**

```
Int add(int x, int y)
{ int z = ...
```

→

```
01100  1001010
00010  1110101
```

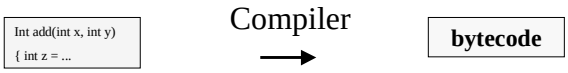
**3.Linkers link Standard code and library modules into executables**



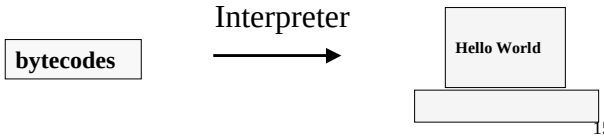

14

# What's different about Java ?

**Java compilers translate source code into an intermediate language called (similar to machine code but platform independent) bytecode.**




**The Java interpreter reads the bytecodes (loads the necessary library bytecodes) and executes it on a specific machine. Unlike machine code Java bytecode is not tied to any particular machine making it architecture neutral.**




15

# Editing, Compiling and Interpreting

**Step1 - Create the code using an editor**

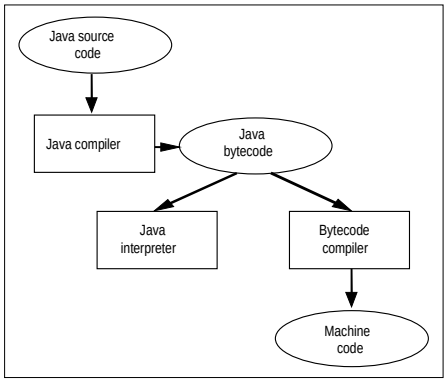
***H:\>edit HelloWorld.java***

**Step2 - Compile the code using javac**

**H:\>javac HelloWorld.java**

**Step3 - Interpret and execute the program**

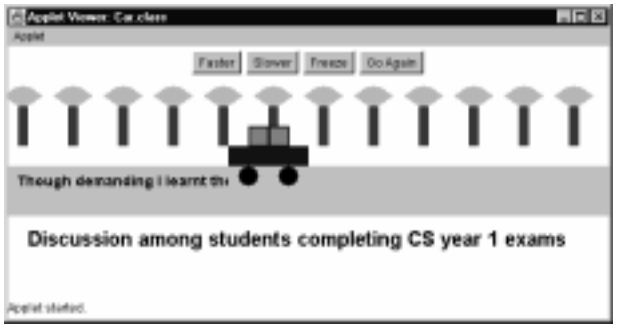
**H:\>java HelloWorld**




16

# Two type of Java Programs Applets

- **Designed to be remotely downloaded as part of an HTML page and executed in the client machine.**
- **They are executed in a browser or appletviewer.**




17

# Stand Alone Applications

- **Stand-alone applications are similar to other applications written in high level language.**
- **Though stand-alone applications can incorporate Graphical user interface and events (mouse-moved, button-pressed) we will concentrate on console based applications in this module.**

**Enter Name** ***Robert Homewood***

**Enter Course** ***CS***

**Hello Robert Homewood Welcome to CS Course**

18

# First Java Program

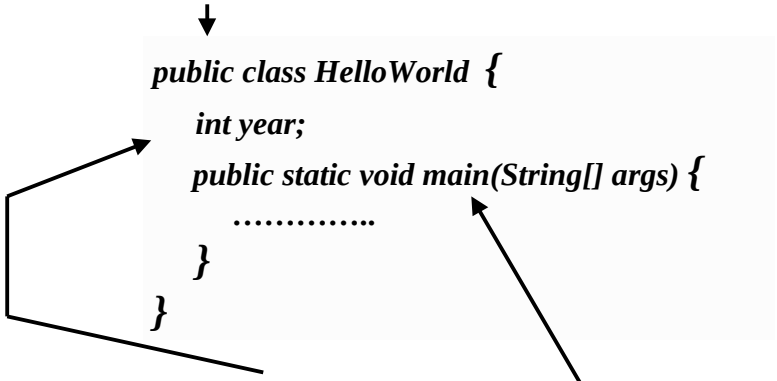
```
// The first Java program
// The famous Hello World!
public class HelloWorld  {
   static int year;
   public static void main(String[] args){
      year = 2001;
     System.out.println(new String("Hello world ") + year);
   }
}
```

```
H:\>javac HelloWorld.java
H:\> java HelloWorld
H:\>Hello World 2001
```

19

# Java is about creating and using classes !!!

All Java programs must have at least one of this




## Classes contain variables and methods

**Classes, methods and related statements are enclosed between { ... }**

20

# How to make my code readable ?

- **Comments make your programs readable**
- **They can appear anywhere in a program**
- **They are ignored by compiler**
- **Type 1 // up to the end of the line is a comment**
- **Type 2 /* all character enclosed between these are comments - hence ignored */**
- **But don't overdo it x = y; // assigning y to x**

```
/* My first Java program prints the famous Hello World!
   The tradition says all great programmers must do this */
public class HelloWorld {
  int year;          // current year
  public static void main(String[] args){
    ……..
  }
}
```

21

# What is the output ?

```
// The first Java program
// The famous Hello World!
public class HelloWorld {
  static int year;
  public static void main(String[] args){
    //  year = 2001;
    System.out.println(new String("Hello world ") + year);
  }
}
```

(a) Hello World

(b) Hello World 0

(c) Hello World 2000

22

# Where does the execution begin ?

**Recall how we ran the program HelloWorld with:**

***H:\> java HelloWorld***

**Q. Where does the execution begin ?**

**A. With the main() method of that class (HelloWorld)**

```
public class HelloWorld {
    int year;
    public static void main(String[] args) {
        ……..
    }
}
```

23

# What is the output when we compile and execute ?

```
public class HelloThere {
  public static void anotherMethod(String[] args){
    System.out.print("There ");
  }
  public static void main(String[] args){
    System.out.print("Hello ");
  }
}
```

**(A) Hello**

**(B) There**

**(C) Hello There**

**(D) There Hello**

24

# Terminating statements

**Each statement must ends with a semicolon ';'**

```
System.out.println(new String("Hello ”));
System.out.println(new String(”World ”));
```


**The effect is still the same with two statements in the same line as in:**

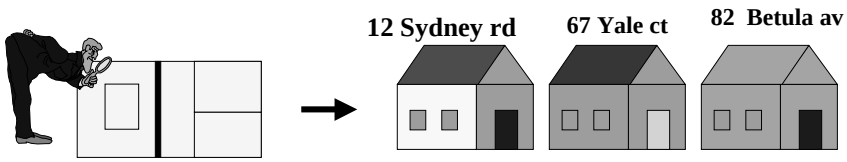
```
System.out.println(…); System.out.println(…);
```

25

# Class and Objects

**A class can be compared to a blueprint created by an architect when designing a house. It defines the important characteristics of the house such as walls, windows, electrical outlets etc.**

**Once we have the blueprint several houses can be built using that blueprint. They may have different addresses, furniture, colors etc.**




**Similarly a class is a blueprint for objects.**

# Creating new Objects with new

**In Java we are provided with a number of pre-created classes such as String and Rectangle, which allows us to create new String and Rectangle objects using the operator new.**

**The statement below creates a new Rectangle object and passes it to the println() method which is a method of the pre-defined PrintStream class.**

```
System.out.println(new Rectangle(10,5,20,30))
```

**This method prints the details of the Rectangle object to the terminal.**

```
java.awt.Rectangle[x=10,y=5,width=20,height=30]
```

27

# Getting a handle to an Object

**What if we want to manipulate the Rectangle object created using one of its own methods - before printing its details ?**

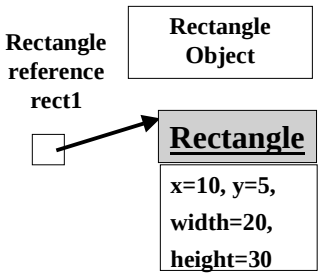
**For example, we may want to apply the translate(int x, int y) method of Rectangle class on that object.**

**In order to call one if it's own methods, somehow we need to get a handle to the object !**

**Next program declares a Rectangle reference and sets it to point (refer) to the newly created Rectangle object with:**

```
Rectangle rect1 = new Rectangle(10,5,20,30);
```

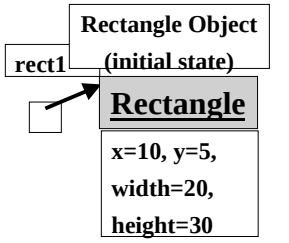
**Subsequently its own methods are called through that reference.**

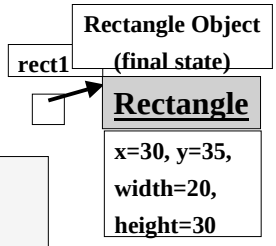



28

# Manipulating the Rectangle Object

```
import java.awt.Rectangle;
public class HelloRectangle  {
  public static void main(String[] args){
    Rectangle rect1 = new Rectangle(10,5,20,30)
   System.out.println(rect1);
   rect1.translate(20,30);  // shifting the location
   System.out.println(rect1);
  }
}
```




**Output from the program**

```
java.awt.Rectangle[x=10,y=5,width=20,height=30]
java.awt.Rectangle[x=30,y=35,width=20,height=30]
```

29

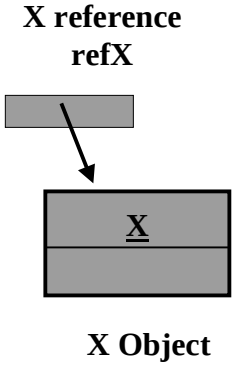
# Creating and manipulating Objects Summary

**To access an object of class X say, we must declare a reference to that class X. Then this reference must be set to refer to that object with a statement of the form**

**X refX = new X(…);**

**Now we can easily manipulate the X object using one of its own methods say meth1() as in**

**refX.meth(…..);**



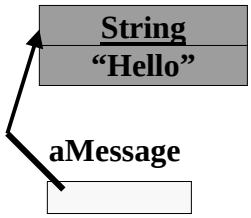

30

# String class - a special one

**In the program below we have declared a String reference aMessage and set it to point to to a newly created String object.**

**Subsequently we output that String object using the String reference aMessage .**

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




**As String objects are commonly used they need not be created them explicitly using the operator new. Hence the 3rd line can be replaced with**

**String aMessage = "Hello";**

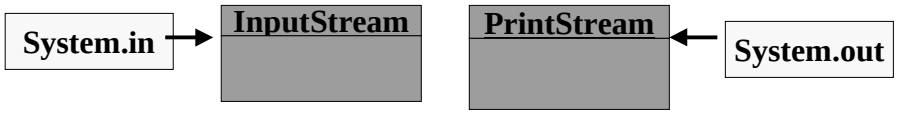
31

# Java provides pre-created objects for Input/Output

**System.out refers to an object of PrintStream class.**

**As console input/output is common in Java they have pre-created objects of type PrintStream (for output) and InputStream (for input).**

**To make them easily accessible the System classes contains references to these objects as shown here.**




**Pre-created InputStream Object (associated with keyboard)**

**Pre-created PrintStream Object (associated with terminal)**

32

# So what do I need to know about these precreated objects ?

**You are free to write to the terminal by writing directly to the object System.out.**

**You are free to use any of the methods of PrintStream class**

**(Keyboard input requires little more processing - we will have to wait until next week)**

33

# Using packages

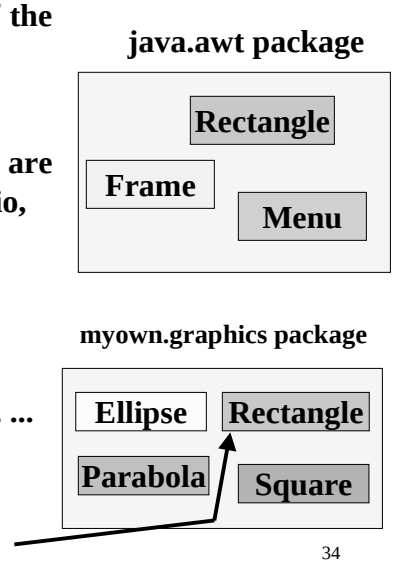
- **In the last program we specified that we are using the Rectangle class of the java.awt package.**

  **import java.awt.Rectangle;**

- **All classes in standard library are placed in packages: java.awt, java.io,**
- **To import all classes use:**

  **import java.awt.*;**

- **Package java.lang is automatically imported - contains String, System, ...**
- **You can place your own classes in packages too - to avoid name clash.**




34

# Stages in Software Development

**Specify**

**What must it accomplish ?**

**Analyze**

**Refine the requirements.**

**Design**

**How to do it ?**

**Implement**

**Code the programs**

**Test**

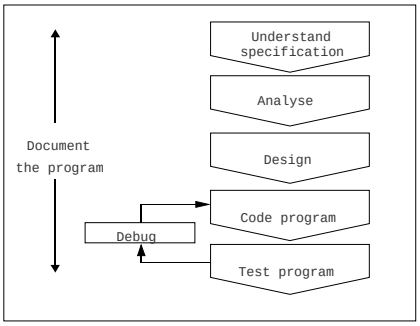
**Run program with different inputs and verify results**

**Document**

**Make it maintainable**

**Write a program to find the roots (real) of a quadratic equation of the form:**

$$ax^2 + bx + c = 0.$$




35

# Some Reserved Words

**Examples:** **class, static, public, void.**

**They have been reserved by the designers and they can't be used with any other meaning.**

**For a complete list see the prescribed book.**

36

# Programming style

- **Java does not impose a particular style but there are rules most Java programmers follow.**
- **One statement per line**

  ***System.out.println("Welcome to RMIT");***

  ***System.out.println("Welcome to CS");***
- **Indentaion (3-4 spaces) show dependency**

  ***public static void main(String[] args) {***

  ***String aMessage = new String("Hello");***
- **Comments help clarify the code**
- **Use blank lines to separate logical sections**
- **Use meaningful identifiers but not verbose**

37

```
/* Written by CT 27/02/2001
   This program finds the roots of a quadratic equation
   To run this program you need the ConsoleReader class  */

import java.io.*;
public class FindRoots {
    public static void main (String[] args) throws IOException {

        double a,b,c;               // coefficients of the equation
        double r1=0.0, r2 = 0.0;  // real roots of the equation
        double disc;                // discriminant
        ConsoleReader console = new ConsoleReader(System.in);

        // Getting user input for coefficients
        System.out.println ("Enter value for a");
        a = console.readDouble();
        System.out.println ("Enter value for b");
        b = console.readDouble();
        System.out.println ("Enter value for c");
        c = console.readDouble();
```

```
    // Computing results
    disc = b*b - 4*a*c;
    // no real roots exist if discriminant is negative
    if (disc > 0.0) {
        r1 = ( -b + Math.sqrt(disc)) / (2*a);
        r2 = ( -b - Math.sqrt(disc)) / (2*a);
    }

    // Displaying results
    if (disc >= 0)
        System.out.println(”Roots are " + r1+ " and " +r2);
    else  System.out.println("No real roots exist");
  }
}
```

39

# Sample User Interface for FindRoots program

```
Enter the value for a
1.0
Enter the value for b
3.0
Enter the value for c
2.0
The roots are -1.0 and -2.0
```

```
Enter the value for a
4.0
Enter the value for b
2.0
Enter the value for c
6.0
No real roots exist
```

**What if I input an alphabet such as ‘s’ as input to a ?**

40

# Choosing Identifiers

- **Identifiers should be meaningful, but not verbose.**
- **Any combination of letters, digits, dollar signs '$' underscore characters '_', not beginning with a digit, is legal.**
- **Legal: Total, lastWord, TaxCalculation**
- **Illegal: 3rdAmmendment, you too, you#too**
- **Avoid: s1, theFirstOfTheStudentsInTheClass**
- **Acceptable : student1, stud_1, firstStudent**

41

## Programming Errors: Compilation Errors

- **detected compiler at compile time ( javac hello.java)**
- **caused by undeclared variables, missing semicolons, incompatible types**

```
disc = b*b - 4*a*c   // missing semicolon
```

```
double r1 =0.0, r2=0.0;
….
root1=(-b + Math.sqrt(disc))/(2*a);  // undeclared variable root1
```

```
int disc;
..
disc = b*b-4*a*c;  // double value cannot be assigned to int
```

42

## Programming Errors: Execution Errors

- **appear when the program runs (java Hello).**
- **Examples are a division by zero, an input of the wrong type, finding the square root of a negative number etc.**
- **Typically execution stops when an exception such as these happens.**
- **For example if in the FindRoots program we input an alphabet (such as s) execution will stop with an error.**

```
Enter value for coefficient a
s
Exception in thread main java.NumberFormatException: s
```

43

## Programming Errors: Logic Errors

- **program compiles and runs, but the results are not what they should be.**
- **For example in the FindRoots program if we write**

***r1 = -b + Math.sqrt(disc) / (2*a);***

**instead of**

***r1 = (-b + Math.sqrt(disc)) / (2*a);***

**the wrong roots will be printed.**

44