the UNIVERSITY of GREENWICH

COMP1618 Software Tools & Techniques

Lecture 1 Introduction

Teaching team

- **Module Leader:**

  Dr. Hai Huang **hai.huang@gre.ac.uk**

- **Module instructor:**

  Yuting Wang. Yuting.Wang@greenwich.ac.uk

Attendance

- **1-hour lecture** (Fridays 1 pm-1:50 pm or 2 pm - 2:50pm, 3-14 weeks)
  - Location: SL101 (LT)
- **2-hour lab** (Fridays, 3-5pm or 4pm-6pm or 6pm-8pm , 3-14 weeks)
  - Location: Please check your individual timetable

Assessment

Coursework: 100% weighting

- **Deadline**: 15 Dec 2023
- **Build an application with GUI using Java**
- **Final deliverables**:
  A demonstration of your application to your tutor
  A zip file containing all code and data files.
  A report containing the evidence of all stages
- **Passing mark**: 50%

Learning outcomes

- Demonstrate a clear and critical understanding of developing software and question its principles and boundaries.
- Appreciate the importance of and demonstrate a critical awareness of user requirements and the impact of these on the design of software.
- Demonstrate the principles and techniques of software design, construction and testing. You will engage with a number of design practices including applying basic UML modelling.
- Comprehension and evaluation of algorithms and data structures

the UNIVERSITY of GREENWICH

Introduction

the UNIVERSITY of GREENWICH

Outline

- Introduction to programming
  - programming / syntax / languages
- Setup Java/ NetBeans
- Basic examples in Java:
  - Hello World / SalesTax
- Basics of Java:
  - Class / Variables / Primitive Data Types / Arithmetic Operators / Input with Scanner / Input & Output with Dialog boxes

What is a computer program?

- A computer program is a set of instructions that tell a computer what to do
- Some examples of computer programs:
  - A web browser (like Firefox or Chrome) is a computer program that can be used to view web pages on the internet
  - An office suite is computer program that can be used to write documents or spreadsheets
  - A video game is a computer program
- Computer programs are often referred to as **code** and programming is sometimes called **coding**

the UNIVERSITY of GREENWICH

Setup Java & NetBeans

- Must have:
  - A text editor (depends on the OS) – for writing code
  - The Java Developer Kit (JDK)
    - Includes the Java Compiler – for compiling Java code
  - The Java Runtime Environment (JRE)
    - Includes Java Virtual Machine (JVM) – for running Java programs
- Recommendation:
  - Java SE: Java Platform, Standard Edition (Java SE) lets you develop Java applications on desktops and servers
  - An Integrated Development Environment (IDE) – a software application that provides comprehensive facilities to computer programmers for software development
    - NetBeans
    - Eclipse

NetBeans (IDE)

- Netbeans is an IDE from Oracle, the current owners of the Java language, and thus, is very easy to use with Java.
  - If you install Netbeans, it will guide you through installing the JDK
- It is free – get it on your laptop/home computer from https://netbeans.org/downloads/
  - you will need NetBeans 15 or above

How to create Java applications

This involves the following steps:

- Write the source code - saved (by you) as **`Filename.java`**
- For this purpose, although you could use any text editor, even NotePad
  - but not Microsoft Word
- We will use NetBeans
- Compile and run the Java source code – in NetBeans you can do this with **Run | Run File** or **Shift+F6**

First example

```
1  /**
2   * First Java program, which says "Hello, world!"
3   */
4  public class Hello { // Save as "Hello.java"
5    public static void main(String[] args) {  // program entry point
6      System.out.println("Hello, world!");    // print message
7    }
8  }
```

- Line 1-3: /* comments */
- Line 4-8: A class called "Hello" is defined via the keyword "class.“ The body is enclosed with {}
  - Line 5-7: main method – entry point of the program execution. Again, enclosed with { }
  - Line 6: programming statement. End with a semi-colon (;)

What Does a Class Look Like?

- It has a name which should start with a capital letter

Key word

Name

Indicates the beginning of the class

```
1  /*
2   * Comment to state the purpose of the program
3   */
4  public class Classname {   // Choose a meaningful Classname. Save as "Classname.java"
5      public static void main(String[] args) {   // Entry point of the program
6          // Your programming statements here!
7      }
8  }
```

Indicates the end of the class

Variables & methods go in between the braces

HelloWorld in NetBeans

...ava | EmailManager.java | CinemaTicket.java | MessageData.java | HelloWorld.java

Source | History

```
1  /*
2   * To change this license header, choose License Headers in Project Properties
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7  package helloworld;
8
9  /**
10  *
11  * @author vt12
12  */
13 public class HelloWorld {
14
15   /**
16    * @param args the command line arguments
17    */
18   public static void main(String[] args) {
19     // TODO code application logic here
20     System.out.println("Hello World");
21   }
22
23 }
```

HelloWorld

Output - HelloWorld (run)

```
run:
Hello World
BUILD SUCCESSFUL (total time: 4 seconds)
```

Comment

Class

Main method

Statement

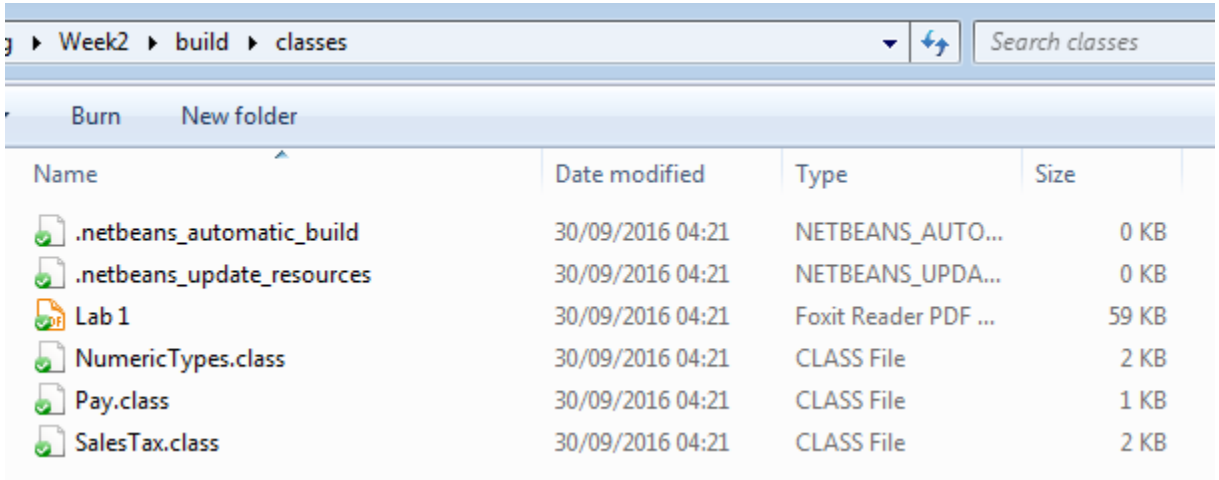
Output

main method

- **public static void *main*(String[] args)**
  - This line must be exactly as shown in the example (except the *args* variable name can be defined).
  - This is the line of code that the *java* command will run first.
  - This method starts the Java program.
  - Every Java *application* must have a `main` method.
    - although applets and servlets don't need one. It is the starting point for the program.
  - When a program starts, the computer looks for the main method and begins execution with the first statement after the main method heading.

Compiling HelloWorld

- When you don't get any error messages, the java compiler will create the *bytecode* file **`HelloWorld.class`** in the folder
- This is the machine-readable file that is actually used when you run your program
- You do not need to use this file yourself but you should know what it is and where it lives



the UNIVERSITY of GREENWICH

Identifiers

- Identifiers are programmer-defined names for:
  - classes
  - variables
  - methods
- Identifiers may not be any of the Java reserved keywords.

Identifiers

- Identifiers must follow certain rules:
  - An identifier may only contain:
    - letters a–z or A–Z,
    - the digits 0–9,
    - underscores (_), or
    - the dollar sign ($) and pound sign (£)
  - The first character may not be a digit.
  - Identifiers are case sensitive.
    - `itemsOrdered` is not the same as `itemsordered`.
  - Identifiers cannot include spaces.

Variable

TYPE	NAME	VALUE	
int	number	1	Stored only Integer
int	sum	500500	Stored only Integer
double	radius	5.5	Stored only floating-point number
double	area	95.0334	Stored only floating-point number
String	greeting	Hello	Stored only texts
String	statusMsg	Game Over	Stored only texts

A *variable* has a ***name***, stores a ***value*** of the declared ***type***.

Let's add a few numbers

```java
/*
 * Add four integers and display their sum
 */
public class FourNumberSum {   // Save as "FourNumberSum.java"
    public static void main(String[] args) {
        int number1 = 30;  // Declare 4 integer variables and assign a value
        int number2 = 9;
        int number3 = 16;
        int number4 = 3;
        int sum;  // Declare an int variable called sum to hold the sum
        sum = number1 + number2 + number3 + number4 ;  // Compute sum
        System.out.print("The sum is ");  // Print a descriptive string
        System.out.println(sum);  // Print the value stored in variable sum
    }
}
```

- What is the output?

Primitive Data Types

- Primitive data types are built into the Java language and are not derived from classes.
- There are 8 Java primitive data types.

```
-byte
-short
-int
-long
-float
-double
-boolean
-char
```

Integer Data Types

- `byte`, `short`, `int`, and `long` are all integer data types.
- They can hold whole numbers such as 5, 10, 23, 89, etc.
- Integer data types cannot hold numbers that have a decimal point in them.
- Integers embedded into Java source code are called *integer literals*.
- See Example: IntegerVariables.java

Floating Point Literals

- When floating point numbers are embedded into Java source code they are called *floating point literals*.
- The default type for floating point literals is `double`.
  - 29.75, 1.76, and 31.51 are `double` data types.
- Java is a *strongly-typed* language.
- See example: Sale.java

The `boolean` Data Type

- The Java `boolean` data type can have two possible values.
  - `true`
  - `false`

The `char` Data Type

- The Java `char` data type provides access to single characters.
- `char` literals are enclosed in single quote marks.
  - ‘a’, ‘Z’, ‘\n’, ‘1’
- Don’t confuse `char` literals with string literals.
  - `char` literals are enclosed in single quotes.
  - String literals are enclosed in double quotes.
- See example: Letters.java

Variable Declarations

- Variable Declarations take the following form:
  - *DataType VariableName;*
    - `byte inches;`
    - `short month;`
    - `int speed;`
    - `long timeStamp;`
    - `float salesCommission;`
    - `double distance;`

Constants

- Once initialized with a value, constants cannot be changed programmatically.
- By convention, constants are all upper case and words are separated by the underscore character.

```
final double CAL_SALES_TAX = 0.725;
```

Initialization Statements

- Initialization statement:
  - When you assign a value to a variable as part of the variable's declaration.
- Initialization statement syntax:

  <type> *<variable>* = *<value>*;

- Example initializations:

```
int totalScore = 0; // sum of all bowling scores
int maxScore = 300; // default maximum bowling score
```

Initialization Statements

- Example initializations (repeated from previous slide):

```
int totalScore = 0; // sum of all bowling scores
int maxScore = 300; // default maximum bowling score
```

- Here's an alternative way to do the same thing using declaration and assignment statements (instead of using initialization statements):

```
int totalScore;     // sum of all bowling scores
int maxScore;       // default maximum bowling score
totalScore = 0;
maxScore = 300;
```

Strings

- If you want to store more than on character
- Use the **S**tring data type

```
String name;
name = "Hai  Huang";
or
String name = "Hai Huang";
```

Basic Arithmetic Operators

Operator	Meaning	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus (Remainder)	x % y
++	Increment by 1 (Unary)	++x or x++
--	Decrement by 1 (Unary)	--x or x--

Input – the Scanner Class

- A pre-written class named `Scanner`, which allows you to get input from a user.
- To tell the compiler you want to use the `Scanner` class, insert the following `import` statement at the very beginning of your program (right after your prologue section and above the `main` method):

```
import java.util.Scanner;
```

- At the beginning of your `main` method, insert this initialization statement:

```
Scanner stdIn = new Scanner(System.in);
```

- After declaring `stdIn` as shown above, you can read and store a line of input by calling the `nextLine` method like this:

```
<variable> = stdIn.nextLine();
```

Input – the Scanner Class

```
import java.util.Scanner;

public class FriendlyHello
{
  public static void main(String[] args)
  {
    Scanner stdIn = new Scanner(System.in);
    String name;
    System.out.print("Enter your name: ");
    name = stdIn.nextLine();
    System.out.println("Hello " + name + "!");
  } // end main
} // end class FriendlyHello
```

These two statements create a keyboard-input connection.

This gets a line of input.

Input – the Scanner Class

- In addition to the `nextLine` method, the `Scanner` class contains quite a few other methods that get different forms of input. Here are some of those methods:

  `nextInt()`

  Skip leading whitespace until an `int` value is found. Return the `int` value.

  `nextLong()`

  Skip leading whitespace until a `long` value is found. Return the `long` value.

  `nextFloat()`

  Skip leading whitespace until a `float` value is found. Return the `float` value.

  `nextDouble()`

  Skip leading whitespace until a `double` value is found. Return the `double` value.

  `next()`

  Skip leading whitespace until a token is found. Return the token as a `String` value.

Input – the Scanner Class

- Here's a program that uses Scanner's `nextDouble` and `nextInt` methods:

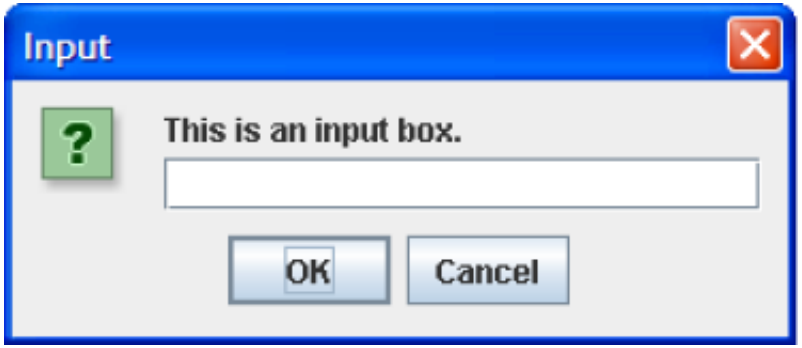
```
import java.util.Scanner;

public class PrintPO
{
  public static void main(String[] args)
  {
    Scanner stdIn = new Scanner(System.in);
    double price;  // price of purchase item
    int qty;       // number of items purchased

    System.out.print("Price of purchase item: ");
    price = stdIn.nextDouble();
    System.out.print("Quantity: ");
    qty = stdIn.nextInt();
    System.out.println("Total purchase order = £" + price * qty);
  } // end main
} // end class PrintPO
```

The `JOptionPane` Class

The `JOptionPane` class provides methods to display each type of dialog box.



Message Dialogs

- The following statement must be before the program's class header:

```
import javax.swing.JOptionPane;
```

- `JOptionPane.showMessageDialog` method is used to display a message dialog.

```
JOptionPane.showMessageDialog(null, "Hello World");
```

- The first argument will be discussed later.
- The second argument is the message that is to be displayed.

Input Dialogs

- The following statement must be before the program's class header:

```
import javax.swing.JOptionPane;
…
String name;
name = JOptionPane.showInputDialog(
          "Enter your name.");
```

- The argument passed to the method is the message to display.
- If the user clicks on the OK button, `name` references the string entered by the user.
- If the user clicks on the Cancel button, `name` references `null`.

Reading Int, Double with an Input Dialog

```
int number;
String str;
str = JOptionPane.showInputDialog(
    "Enter a number.");
number = Integer.parseInt(str);

double price;
String str;
str = JOptionPane.showInputDialog(
    "Enter the retail price.");
price = Double.parseDouble(str);
```

String

Convert to numeric

the UNIVERSITY of GREENWICH

Fixing/Debugging

- We need to fix the problems
- One way is just to look at the program and try to figure out where the error is occurring
  - fine for short programs like the sale taxes example, but imagine something like Microsoft Word which probably has hundreds of thousands of lines of code
- We need a way to check what is going on when we run the program and help find the bugs
  - this is known as **debugging**
- In the past programmers would write output to the console at various points in the program to check everything was correct at each point
  - this is cumbersome and slow

Tutorial: SalesTax Example

- SalesTax allows the user to check the cost of an item with a sale tax
- There is a problem with our program however – some errors
  - Incorrect package
  - Syntax error

```
1   //This program calculates the total pr
2
3
4   import java.util.Scanner;
5
6   public class SalesTax
7   {
8           public static void main(String
9           {
10                  //identifier declaratio
11                  final double TAX_RATE =
12                  double price;
13                  double tax
14                  double total;
15                  String item;
16
17
18                  //create a Scanner obje
19                  Scanner keyboard = new
20
21                  //display prompts and g
22                  System.out.print("Item
23                  item = keyboard.nextLin
24                  System.out.print("Item
25                  price = keyboard.nextDo
26
```

the UNIVERSITY of GREENWICH

Summary

- We covered the Basics of Java.
  - Tools (NetBeans)
  - Terminology and syntax
  - Variable / Data type / Operators / Constant
  - Input / Output with Scanner, Dialog boxes