CO452
Programming Concepts

Lecture 1

Basics, Main, Input and Output

In this first lecture

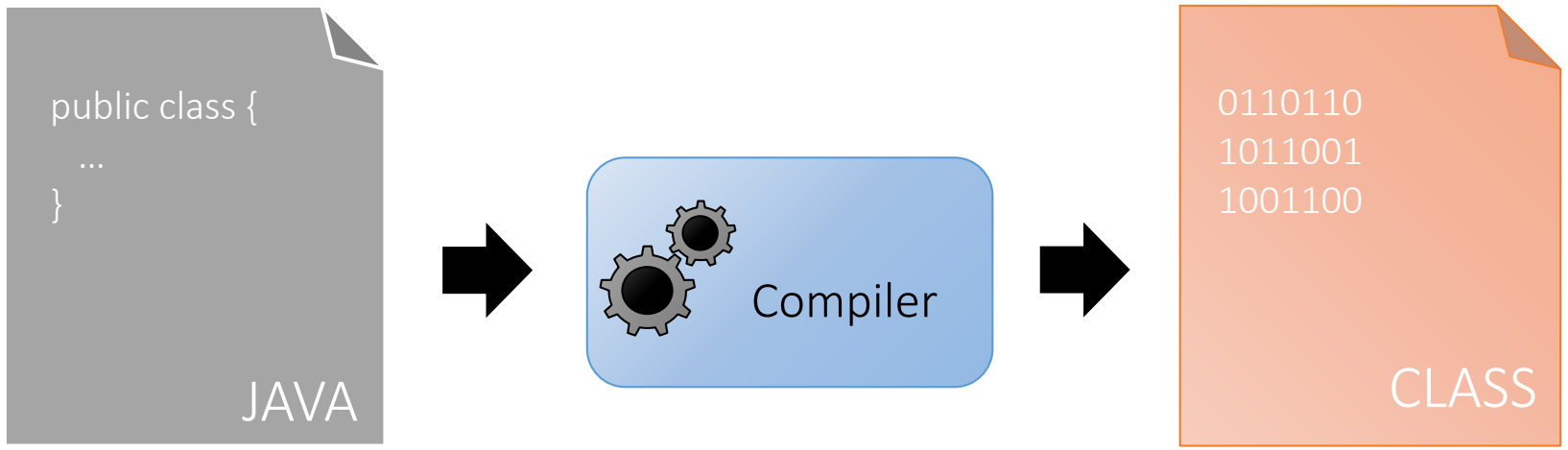
- Writing our First Program (output + main)
- Declaring Variables
- Assigning values to variables
- Constants
- Input: EasyIn + Scanner

Computation

Question!

What is a computer?

Compilation



*.java
source code

javac

*.class
byte code

The main method

Starting our programs

There is a method called **main** which is the defined entry point for starting programs in Java.

Later we will use the main method to create objects of other classes and call their methods.

For now, we will create simple programs within the main.

What is a method

A method is a block of code that performs a well defined task

Could be a single line of code, or could be multiple statements which contribute to the achievement of a task.

In our first week, we shall use the main method to achieve basic tasks, such as outputting a message to the screen, or storing the user's name.

The main method

The main method has to be defined within a class in Java, and usually resides in a standalone class (here: Program)

```
public class Program
{
    public static void main(String[] args)
    {
        …
    }
}
```

The main method is **static**

Dynamic members of a class would be accessed through an object (an instance). However, the main method is defined to be **static**, meaning that the method can be called through the class name (no object required).

```
public class Program
{
    public static void main(String[] args)
    {
        …
    }
}
```

The main method

The main method accepts an argument (parameter) 'args'. The 'args' is an array, and optionally allows data to be passed in at compilation that can be used by main, but this is not required.

```
public class Program
{
    public static void main(String[] args)
    {
        …
    }
}
```

Output

The println() method

Use the **println()** method of the System class to output the text data placed within speech marks " " to screen

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

Hello World

_

The println() method

It is also possible to output digits and symbols, but be aware that they will be formatted as a String data type " "

```
public class Program
{
    public static void main(String[] args)
    {
        System.out.println("Hello Nick");
        System.out.println("Your ID is 12345678");
    }
}
```

```
Hello Nick
Your ID is 12345678
_
```

Variables

What are variables?

- Variables are storage areas for a single value
- Values come in different formats: from whole numbers, decimal numbers, small numbers, large numbers, to words, letters and phrases
- The value is stored at a memory address (RAM, which is temporary memory) and can then be accessed through the variable name
- Whenever we refer to the variable name we refer to the value stored in memory
- Variables can only store one value at a time

How do we define variables?

1. First, we have to define the type of data that we want to store – **data type**

2. Then we have to select a **meaningful name** which represents that data well

```
String name;
```

1. Data types

int	whole numbers e.g. 10, 75, 200
double	decimal numbers e.g. 2.56, 0.314, 20.75
String	a **class** that represents more than one letter
char	single letter or number or symbol
boolean	true or false

2. Name

- Important to select names that describe the values they hold
- Best practice to start variable names with a letter or '_'
- Can’t have spaces in the variable name (use camel**C**ase or '_')
- Can’t be a reserved word (public, int, class)
- Beware of casing: ‘NAME’ and ‘name’ are different variables!

Assigning values to variables

The assignment operator (=) is used to assign a value to a variable

The value on the right hand side of the '=' sign is ‘assigned’ to the storage area on the left hand side of the '=' sign

```
String name = "Nick";
int id = 21015672;
boolean isComplete = true;
```

Different to maths with trying to balance right and left hand sides of an equation…

Constants

Constants

Constants are initialised with a value, but it’s not possible to overwrite that value later.

Setting a value to be constant prevents it from being overwritten.

In Java, use the keyword **final**

It’s also conventional to use **UPPERCASE** letters for an identifier.

```
final int FEET_IN_MILES = 5280;
final int HOURS_IN_DAY = 24;
final int MAX_MARK = 100;
```

Output values to screen

Output values of variables

Placing the variable name in-between the parentheses of the **print()** method will output the value stored at the memory address

```
public class Program
{
    public static void main(String[] args)
    {
        String name = "Nick";
        System.out.print("Hello ");
        System.out.print(name);
    }
}
```

Hello Nick

_

Concatenation (+)

We can also use the **+** sign (also known as **concatenation**) to join data together as one String to be output

```
public class Program
{
    public static void main(String[] args)
    {
        String name = "Nick";
        System.out.println("Hello " + name);
    }
}
```

Hello Nick

_

Input via Scanner

Input via Scanner

Scanner is Java's defined class which handles the connection between the keyboard and the program: known as the input stream.

It does utilise Classes and Objects – don't worry if you don't understand the concepts right now – we'll revisit this topic later in the module.

Input via the Scanner

```
//1. import the Scanner class
import java.util.Scanner;
//2. create an object of the Scanner class
private Scanner reader = new Scanner(System.in);
//3. call methods on the object
String input = reader.nextLine(); //returns a string
int number = reader.nextInt(); //returns an int
```

Portion of the Scanner class

```
public class Scanner…
{
   /**
    * Advances this scanner past the current line and returns the input that was skipped.
    */
   public String nextLine()
   {
      …
   }
   /**
    * Scans the next token of the input as an int.
    */
   public int nextInt()
   {
      …
   }
}
```

Full documentation available at:
https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html

InputReader

Input via InputReader

To simplify this process of input, we're going to call methods of the InputReader class. This saves us having to create an instance of the Scanner class ourselves.

The InputReader class contains code which defines the Scanner object and has methods to return data entered by the keyboard.

We simply call the method which has the name matching the type of data we want to return.

Using methods of the InputReader Class

A value can be input from the keyboard by accessing the appropriate method as follows:

someVariable = **InputReader.methodName();**

String name;

name = InputReader.**getString();**

InputReader class

Java type	InputReader method
String	getString()
int	getInt()
double	getDouble()

Input example: String data

We can use the getString method of InputReader to capture data from the keyboard.

```
public class Program
{
    public static void main(String[] args)
    {
        String name;
        System.out.println("Hello, what's your name?");
        name = InputReader.getString();
        System.out.println("Hello " + name + "!");
    }
}
```

```
Hello, what’s your name?
_
```

```
Hello, what’s your name?
Nick_
```

```
Hello, what’s your name?
Nick
Hello Nick!
_
```

Input example: int data

We can use the getInt method of InputReader to capture data from the keyboard.

```
public class Program
{
    public static void main(String[] args)
    {
        int id;
        System.out.print("Enter your ID: ");
        id = InputReader.getInt();
        System.out.println("Your ID is " + id);
    }
}
```

Enter your ID: _

Enter your ID: 20122019_

```
Enter your ID: 20122019
Your ID is 20122019
_
```

Next lecture

Sequence, Selection and Iteration

- **Sequence** mandates that statements be executed in order (line by line)
- **Selection** (conditional) statements will execute a **block of code once** when the condition is true
- **Iteration** allows us to **repeat** statements within a block **whilst** the condition is **true**