Computer Science 101 S1

Lecture 3

Contents

Java Syntax
Displaying Output

Coursebook: **§3, §4**

Review

Write the source code for the application (which starts the program).

MyApplication.java

```
public class MyApplication {
        public static void main(String[] args) {
                MyProgram p = new MyProgram();
                p.start();
        }
}
```

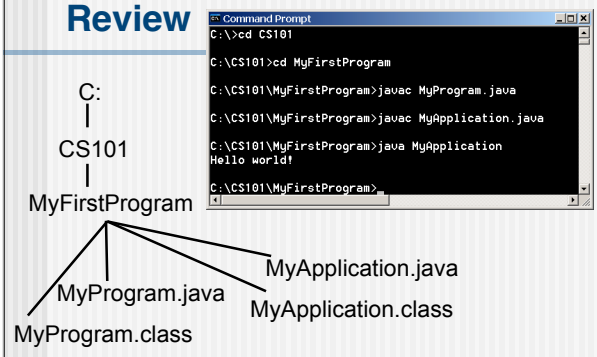
Write the source code for the program:

MyProgram.java

```
public class MyProgram {
        public void start() {
                System.out.println("Hello World");
        }
}
```

instructions which will be executed

Review



During lectures

We will only look at the *program* file:

```
public class MyProgram {
     public void start() {

          System.out.println("Hello World");

     }
}
```

When we want to write a different program, we will define different instructions

Java Syntax

We need to learn:

- the words which have special meaning
- the punctuation and symbols that are required
- the combinations of words and symbols that are allowed

The compiler is very picky

We must be very precise when writing our code

7

Errors

You will make mistakes – don't worry, this helps you learn.

There are three types of errors:

- syntax errors
- runtime errors
- logic errors

8

Style

We are writing our programs for two different audiences:

- other people
- the computer

To improve consistency and readability:

- all programmers should follow the same conventions for the layout and organisation of their code
- for the Java language, the document "Code Conventions for the Java Programming Language" describes these style guidelines:

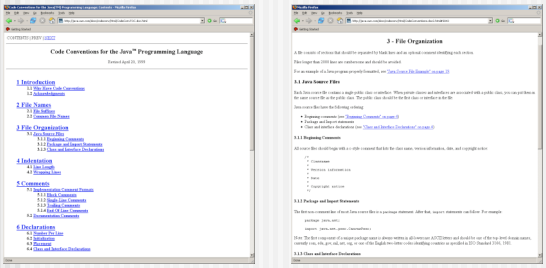
http://java.sun.com/docs/codeconv/

In this course, pay particular attention to *indentation* and *comments*.

9

Code conventions

http://java.sun.com/docs/codeconv/



10

The structure of a Java program

Our programs are made up of *statements*, which are contained inside *methods*, which are contained inside *classes*:

```
public class MyProgram {

    public void start() {

        System.out.println("Hello World");
    }
}
```

Class

Method

Statement

11

Statement

Syntax

<instruction> ;

Description

A single instruction in Java
Always ends with a semicolon

Examples:

```
System.out.println("Hello");
int x;
j = 3;
```

12

Identifiers

An *identifier* is the name that we give to classes, methods and variables in our programs.

- classes
- methods
- variables

```
public class MyProgram {
      public void start() {
            System.out.println("Hello World");
      }
}
```

Valid identifiers:

- only upper case and lower case letters, numbers and underscore
- identifiers must not start with a number
- identifiers must not be the same as a reserved Java keyword

13

Class name identifiers

Class name identifiers:
- start with an upper case letter
- beginning of subsequent words should be upper case
- all other characters lower case

MyProgram
LabOneProg
SpaceInvaders
Assignment1Program

```
public class MyProgram {
      public void start() {
            System.out.println("Hello World");
      }
}
```

Program Layout

The compiler does not care about the layout of our code:

```
public class MyFirstProgram {

   public void start() {
      System.out.println("Hello World");
   }
}
```

```
public class MyFirstProgram { public
void start( ) { System . out . println (
"Hello World" ) ; } }
```

```
public
class
MyFirstProgra
m
{
public
void
start
(
)
{
System
.
out
.
println
(
"Hello World"
)
;
}
}
```

15

Program Layout - indentation

We should always organise our code so that it is easy for us and others to read.

Indentation rule

Indent any code appearing between curly braces (use tab)

Use this style all the time

```
public class IndentationExample {
   public void start() {
      System.out.println("Hello World");
   }
}
```

16

Comments

Comments

Ignored by the compiler

Aimed at **people** who read the code

Inline comments

All text until end of line is ignored e.g.

```
// comment goes here
```

Multi-line comments

All text between start and end of comment is ignored e.g.

```
/* comment goes
   here */
```

17

Comments

Remember, comments are ignored by the compiler.

Good code should be "self-documenting", so use comments sparingly.

```
/*
 *  Author:    Adriana Ferraro
 *  Date:      March 2007
 *  Purpose:   To illustrate the use of comments
*/
public class MyFirstProgram {
      public void start() {
                  // prints message
            System.out.println("Ciao Mondo");
      }
}
```

18

Case sensitivity

Java is case-sensitive.

```
System.out.println("hi");
```

is correct and will compile:

```
System.Out.println("hi");
```

is NOT correct and will NOT compile.

19

Displaying output

`System.out.println()`

This statement can be used to print text to the screen as output:

```
System.out.println("Some text");
```

- this prints out whatever is inside the speech marks
- a newline character is printed at the end of the line
- if nothing is inside the parentheses, a blank line is printed

20

Displaying output

Example:

```
public class Printing {
     public void start() {
          System.out.println("abc");
          System.out.println();
          System.out.println("123");
     }
}
```

produces:

```
abc

123
```

21

Literals

Integer: whole numbers (exact values) e.g.
3, -2045, 74

Floating point: numbers which include decimal point (approximate) e.g.
3.65, 0.0, -1267.692, 3.333333333

boolean: true or false e.g.
true, false

22

More Literals

Character: single symbol from Unicode character set e.g.
'A', 'b', 'C', '5'

String: sequence of characters surrounded by quotes e.g.
"Hello world", " ", "", "BLING"

Printing literals

```
-45
0.034
true
g
This is a String
```

Use System.out.println()

Any literal value can be used in parentheses

```
public class PrintingLiterals {
     public void start() {
          System.out.println(-45);
          System.out.println(0.034);
          System.out.println(true);
          System.out.println('g');
          System.out.println("This is a
                              String");
     }
}
```

24

Escape sequences

Suppose you want to print:

```
Hello, "World"!
```

Will the following work?

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

To indicate we want to print the quotation mark, we must put a backslash character in front of it:

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

25

Escape sequences

Another common escape sequence is:

```
\n
```

which prints a new line.

Some other escape sequences:

\"	quotation mark
\\	backslash
\n	new line
\t	tab

26

System.out.print()

Prints output (just like println())

Does NOT add a carriage return (newline) at the end of the text.

```
public class PrintExample{
  public void start(){
    System.out.print("My name is ");
    System.out.print("Adriana");
    System.out.println("Where is this
                                printed?");
    System.out.print("and this?");
  }
}
```

```
My name is AdrianaWhere is this printed?
and this?
```

27

What you need to know

Java statements end with a semicolon.

Identifiers - names given to classes, methods and variables. What is a valid identifiers.

Identifier for a class name.

Indentation rule.

Inline comments, multiline comments.

28

What you need to know

Java is case sensitive.

Types of literals: Integer, Floating-point, boolean, character, String.

System.out.println() and System.out.print() are used to print somthing to the standard output.

System.out.println() and System.out.print() can be used to print any literal value to the standard output.

Escape sequences inside a String.

29

Ex01 – What is the output

```
public class OutputProgram1 {
  public void start() {
     System.out.println("       4");
     System.out.println("plus   6.45");
     System.out.println("  is 10.45");
     System.out.println();
     System.out.println(true);
  }
}
```

30

Ex02 – What is the output

```
public class OutputProgram2 {
  public void start() {
     System.out.println("\"Chocolate\\");
     System.out.println("\nnn\\n\"");
     System.out.println();
     System.out.println("Yum");
  }
}
```

31

Ex03 – What is the output

```
public class OutputProgram3 {
  public void start() {
     System.out.print("one");
     System.out.print(" two");
     System.out.println();
     System.out.print(" three\n");
     System.out.println("zero");
    }
}
```

```
public class                              {

        public static void main(String[ ] args) {
                        p = new                   ();
                p.start();
        }
}
```

```
public class                              {
        public void start() {









        }
}
```

34