CS620
Structured Programming
Introduction to Java

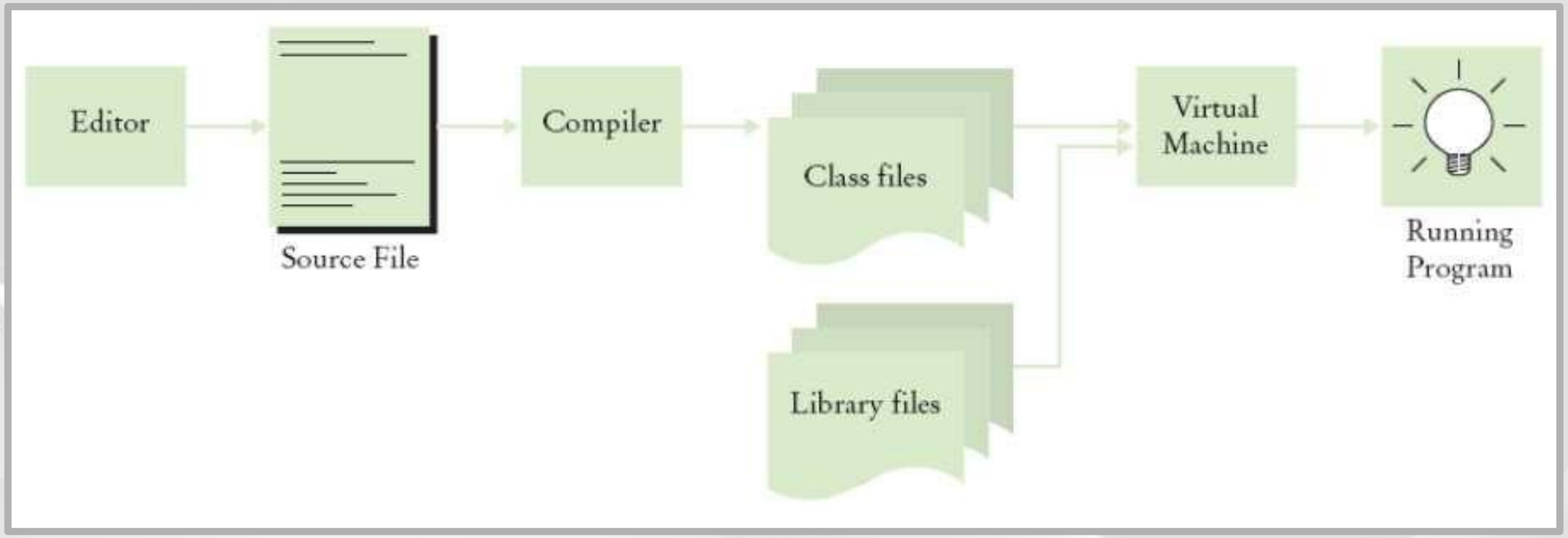
Day 2 - Lecture 1

Recap of Day 1 and Solutions to presented

Recap

- Just to refresh and clarify some of what we covered yesterday..

Recap


Editor
Source File
Compiler
Class files
Library files
Virtual Machine
Running Program

Recap

- The programming process
  - Iterative
  - Trial & Error
  - Write, Compile, Test, Fix, Repeat

- Files:
  - Source Code (ClassName.java)
    - Same name as Class
    - javac ClassName.java
  - Class file (ClassName.class)
    - java ClassName

Recap

- Parts of the text
  - Comments
  - Blocks
  - Class Definition
  - Methods
  - Variables
  - Statements

Recap - Class Definitions

- Class definition
  - ‘class’ is a java keyword
    - Small ‘c’
  - Class name should match file name
- Examples:
  - **`class HelloWorld {…}`**
    - **`javac HelloWorld.java`**
    - **`java HelloWorld`**
  - **`class Integers {…}`**
    - **`javac Integers.java`**
    - **`java Integers`**

Recap - Class Definitions

- Class
  - The actual meaning of the name chosen for your class name doesn’t matter

  - The examples given such as
    - HelloWorldApp
    - HelloWorldAppString
    - HelloWorldAppStringDecl
    - HelloWorldAppStringDeclConcat
    - Were just named to indicate the changes made when putting them together for demonstration

Recap - Class Definitions

- Class
  - You can call your class anything you want
    - Except for keywords, etc. (Standard identifier rules)
  - Try to keep Class names reasonably short but descriptive in some way
  - One practice is to give your classes meaning full names and to put them in to a directory for each new day. I.e. Day one exercises should go into a directory of folder called day1.

Recap - Methods

- Methods
  - Parts of a method declaration:
    - Modifiers: ‘public static’
    - Return type: ‘void’
    - Name/Identifier: ‘main’
    - Arguments: ‘(String[] args)’
  - **`public static void main(String args[]);`**
- Other examples:
  - **`private static String someFunction(int arg1, String arg2);`**
  - **`public int someFunction(int arg1, String arg2);`**
  - **`protected static someFunction(double aDecimal);`**

Recap - Variables

- Variables
  - Declare
    - `<variable type> <variable name> ;`
    - `int someNumber;`
    - `String someString;`
  - Initialize / Assign
    - `int someOtherNumber = 15;`
    - `String someOtherString = “Hello”;`
    - `double someDecimal = 0.005;`

Recap - Variables

- Variables
  - Pass to Methods
    - `System.out.println( someInt + someString );`
    - `System.out.println("A variable: " + someVariable);`
    - `SomeFunction(foo, bar, stuff);`
  - Expressions
    - `int someNumber = 10;`
    - `int someOtherNumber = someNumber + 5;`
    - `int answer = someNumber + someOtherNumber ;`
    - `int remainder = someNumber % someOtherNumber ;`
    - `double anotherDecimal = 11 / 13;`

Recap - Types

- int
  - For whole numbers
  - ‘Truncates’ decimals to the next-lowest whole number
- double
  - For decimal values
  - Holds a huge range of decimals
- String
  - For characters/text
  - Numbers stored in strings become like ‘letters’, they lose their ‘numberness’

Recap - Types

- boolean
  - For boolean algebra / conditionals
  - Only holds **true** or **false**

- Byte / Short / Long / Float
  - For optimizing data storage
  - Be aware of these types and what they are; but don’t worry about using them yet.

Recap - Operators

- +, -, *, /
  - Basic mathematical operators
  - Remember that ‘+’ can be used to join strings together (*concatenation*)
- % (Modulo)
  - Gives the remainder of one number divided by another
  - 10 % 5 gives 0
  - 13 % 5 gives 3
  - 16 % 3 gives 1
  - Useful for checking even/odd numbers

Recap - Operators

- `++`
  - Increment
  - ‘x++’ is the same as ‘x = x + 1’
  - **`int x = 4;`**
  - **`x++;`** `(x is now 5)`
  - **`x++;`** `(x is now 6)`

- `--`
  - Decrement
  - ‘x--’ is the same as ‘x = x - 1’
  - **`int x = 7;`**
  - **`x--;`** `(x is now 6)`
  - **`x--;`** `(x is now 5)`

- `==`
  - Equality
  - Only use this when checking equality
  - i.e.; When **asking** if two things are equal
  - Remember that ‘=‘ is the **assignment** operator

Pseudo-Code

- Getting away from the compiler for a bit
- **Thinking** about programming
- **Planning** programs in advance
- Working outside of the constraints of the language
  - At least at first!

Pseudo-Code

- ‘Pseudo-Code’ means writing code that doesn’t actually work, but gets the idea across

- It’s a smart way to design algorithms

- It lets you ‘program’ loosely without worrying about the exact details of the language

Pseudo-Code

- We’ve already written some pseudo-code without realising it.
- Comments that we put into our code to describe what we’re doing can be considered pseudo-code of a sort
- Writing pseudo-code is good practice both for algorithm-design **and** for commenting

Algorithms

- Algorithms are a way of describing what a program or function should do.
  - Can be written in code, or in plain language, spoken aloud, shown in images, drawn as a circuit, etc.
- They can also describe a general way of achieving a particular task
- Sorting algorithms are commonly studied. Finding the most efficient way to sort a data set is an important topic in computer science.

Writing Pseudocode

- Hello World pseudocode example:

```
– Print “Hello World!” to the screen
– End
```

- Just like that! But let’s expand it just a little..

```
– In the HelloWorld Class:
    • Store “Hello World!” in a string var
    • Print the string var to the screen
    • End
```

Writing Pseudocode

- Another example, the numbers task from yesterday:
  - In the NumbersTask Class:

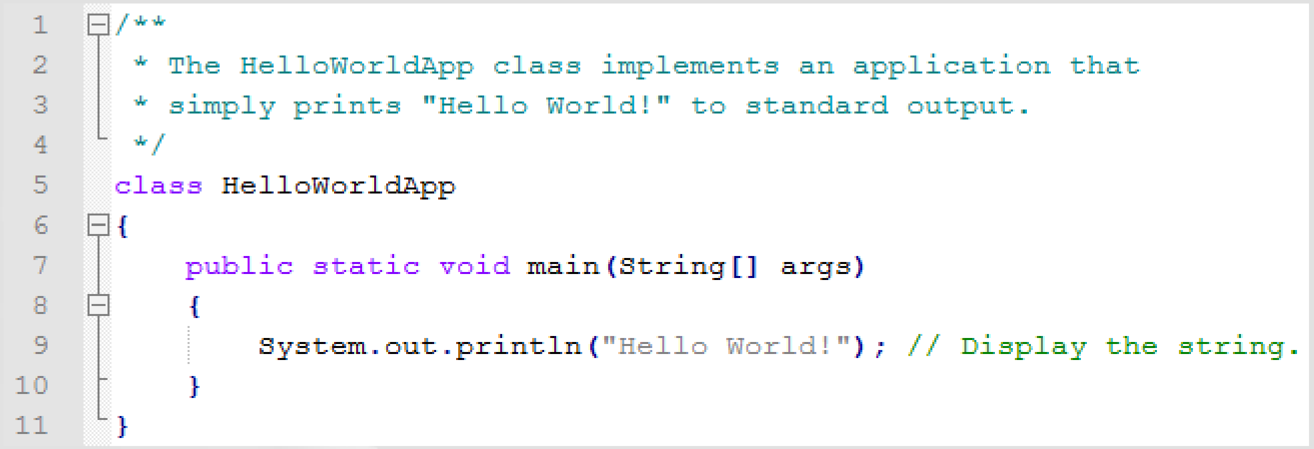
```
• Hold 3 numbers in variables, x (10), y (5), z (6)
• Print ( x )
• Print ( y )
• Print ( z )
• Print ( x + y + z )
• Print ( x * y * z )
• End
```

- You might notice it bears a similarity to the instructions in the practical exercises
- (Relatively) Plain English!

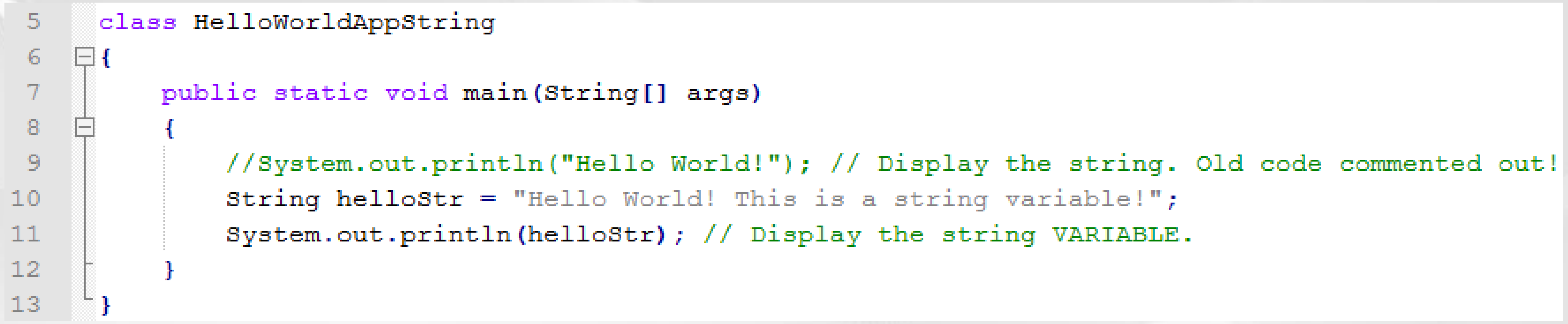
Lab Exercise Walkthroughs

- Task by task
- Ask questions!

Lab Exercises – Day 1

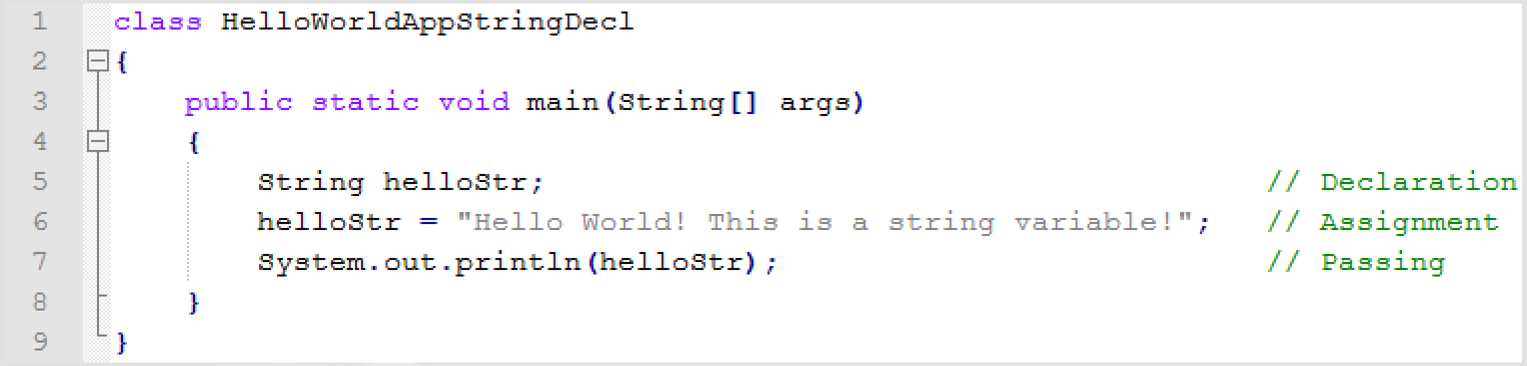


```java
/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

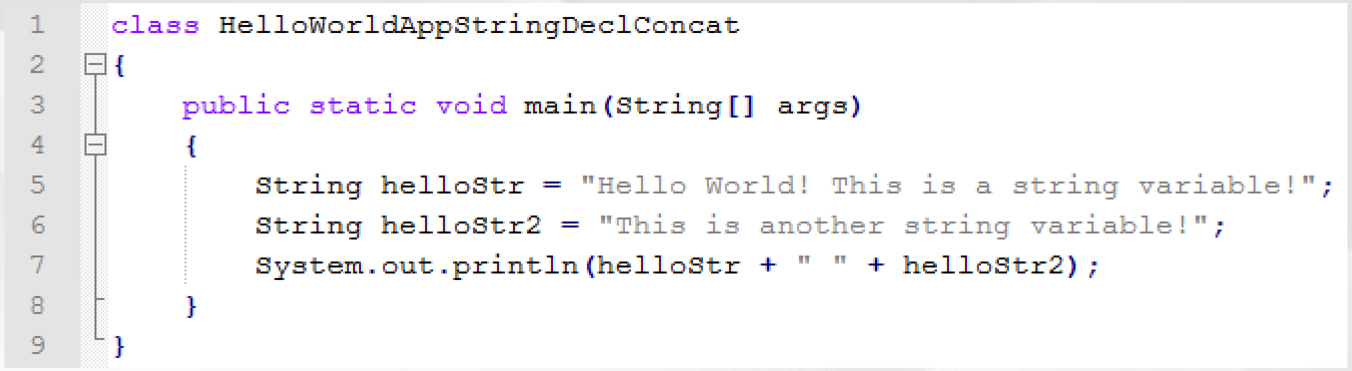


```java
class HelloWorldAppString
{
    public static void main(String[] args)
    {
        //System.out.println("Hello World!"); // Display the string. Old code commented out!
        String helloStr = "Hello World! This is a string variable!";
        System.out.println(helloStr); // Display the string VARIABLE.
    }
}
```

Lab Exercises – Day 1



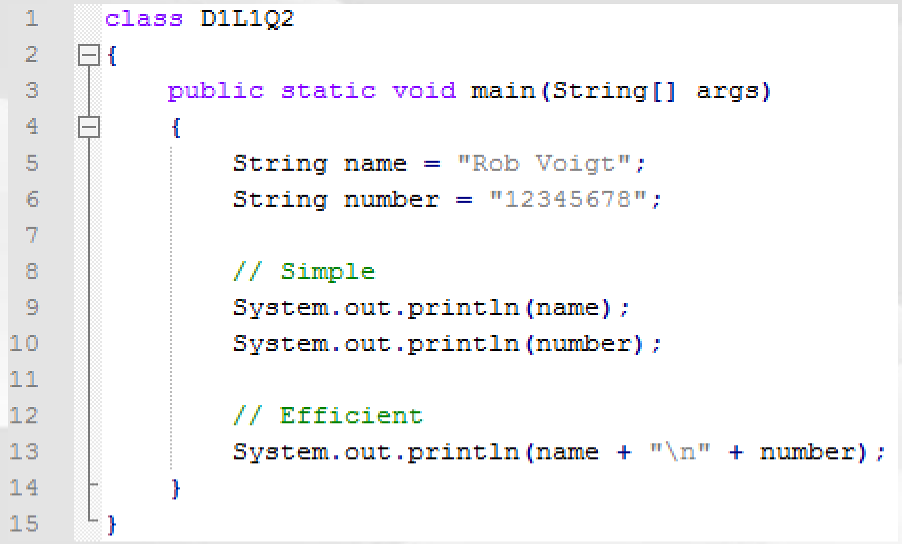
```
class HelloWorldAppStringDecl
{
    public static void main(String[] args)
    {
        String helloStr;                                        // Declaration
        helloStr = "Hello World! This is a string variable!";   // Assignment
        System.out.println(helloStr);                           // Passing
    }
}
```



```
class HelloWorldAppStringDeclConcat
{
    public static void main(String[] args)
    {
        String helloStr = "Hello World! This is a string variable!";
        String helloStr2 = "This is another string variable!";
        System.out.println(helloStr + " " + helloStr2);
    }
}
```

Lab Exercises – Day 1

- L1Q2 - Write a program that prints your name and student number to the standard output when it runs.
- The output should look something like:
  - `"Robert Voigt"`
  - `"12345678"`



```java
class D1L1Q2
{
    public static void main(String[] args)
    {
        String name = "Rob Voigt";
        String number = "12345678";

        // Simple
        System.out.println(name);
        System.out.println(number);

        // Efficient
        System.out.println(name + "\n" + number);
    }
}
```

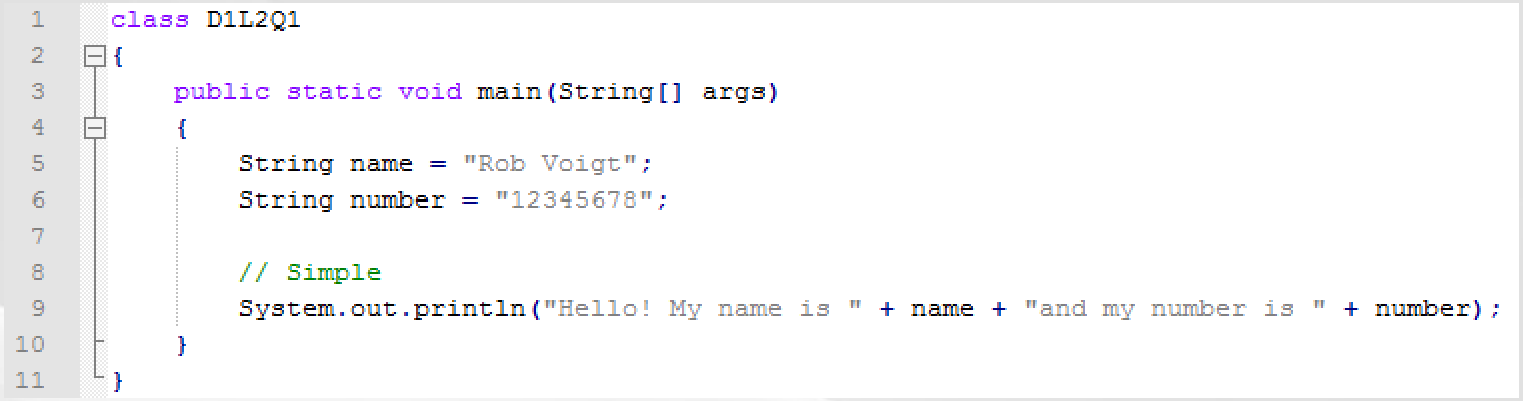
Lab Exercises – Day 1

- L2Q1 - Write a program that prints your name and student number to the standard output when it runs, but this time print the output on just one line by concatenating the variables and putting some extra text with them.

- The output should look something like:

- `"My name is Robert Voigt and my student number is 12345678"`

Lab Exercises – Day 1



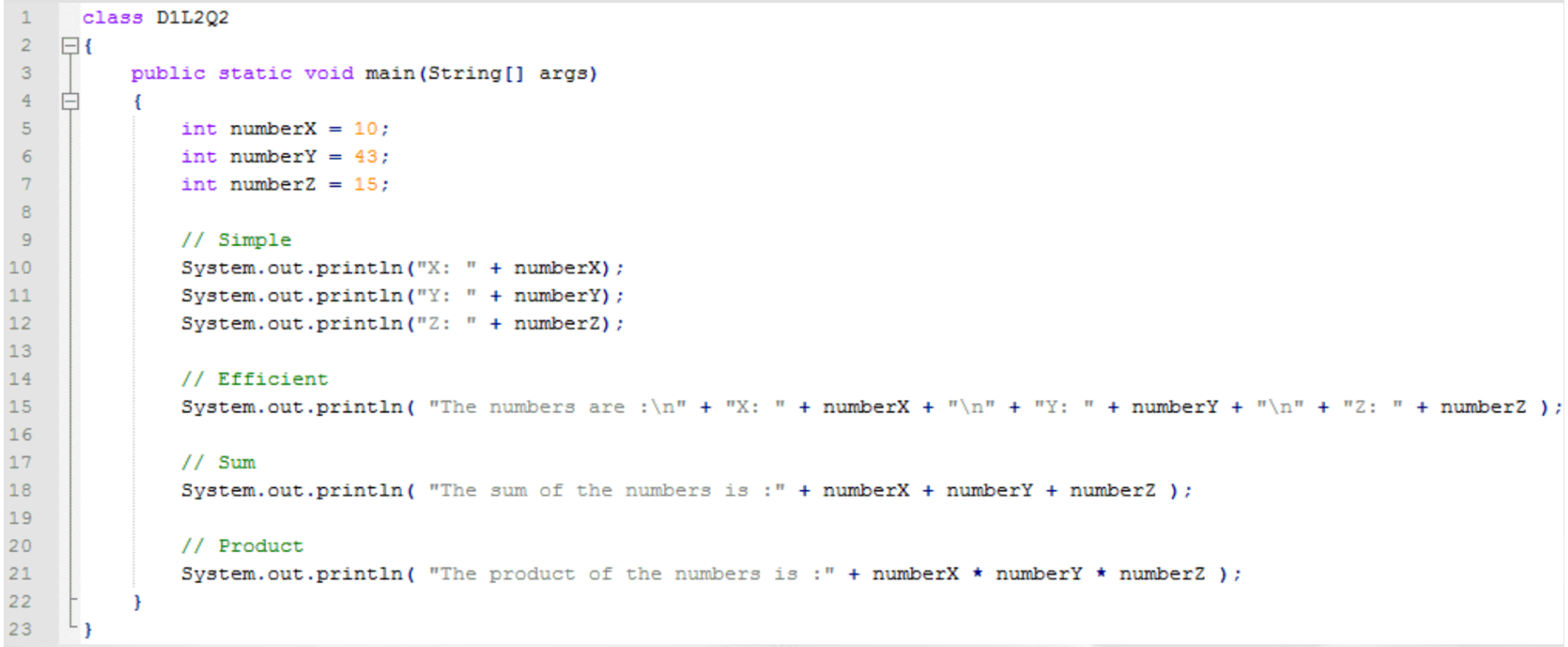
```java
class D1L2Q1
{
    public static void main(String[] args)
    {
        String name = "Rob Voigt";
        String number = "12345678";

        // Simple
        System.out.println("Hello! My name is " + name + "and my number is " + number);
    }
}
```

Lab Exercises – Day 1

- L2Q2 - Write a program that contains the following integer variables and prints each of them on a separate line, one after another:
  - numberX with a value of 10
  - numberY with a value of 43
  - numberZ with a value of 15

- L2Q3 - Use the code from part 2 and print the sum and product of the three variables.

Lab Exercises – Day 1



```java
class D1L2Q2
{
    public static void main(String[] args)
    {
        int numberX = 10;
        int numberY = 43;
        int numberZ = 15;

        // Simple
        System.out.println("X:  " + numberX);
        System.out.println("Y:  " + numberY);
        System.out.println("Z:  " + numberZ);

        // Efficient
        System.out.println( "The numbers are :\n" + "X:  " + numberX + "\n" + "Y:  " + numberY + "\n" + "Z:  " + numberZ );

        // Sum
        System.out.println( "The sum of the numbers is :" + numberX + numberY + numberZ );

        // Product
        System.out.println( "The product of the numbers is :" + numberX * numberY * numberZ );
    }
}
```