CSY2030
Systems Design & Development
Revision of Java 2

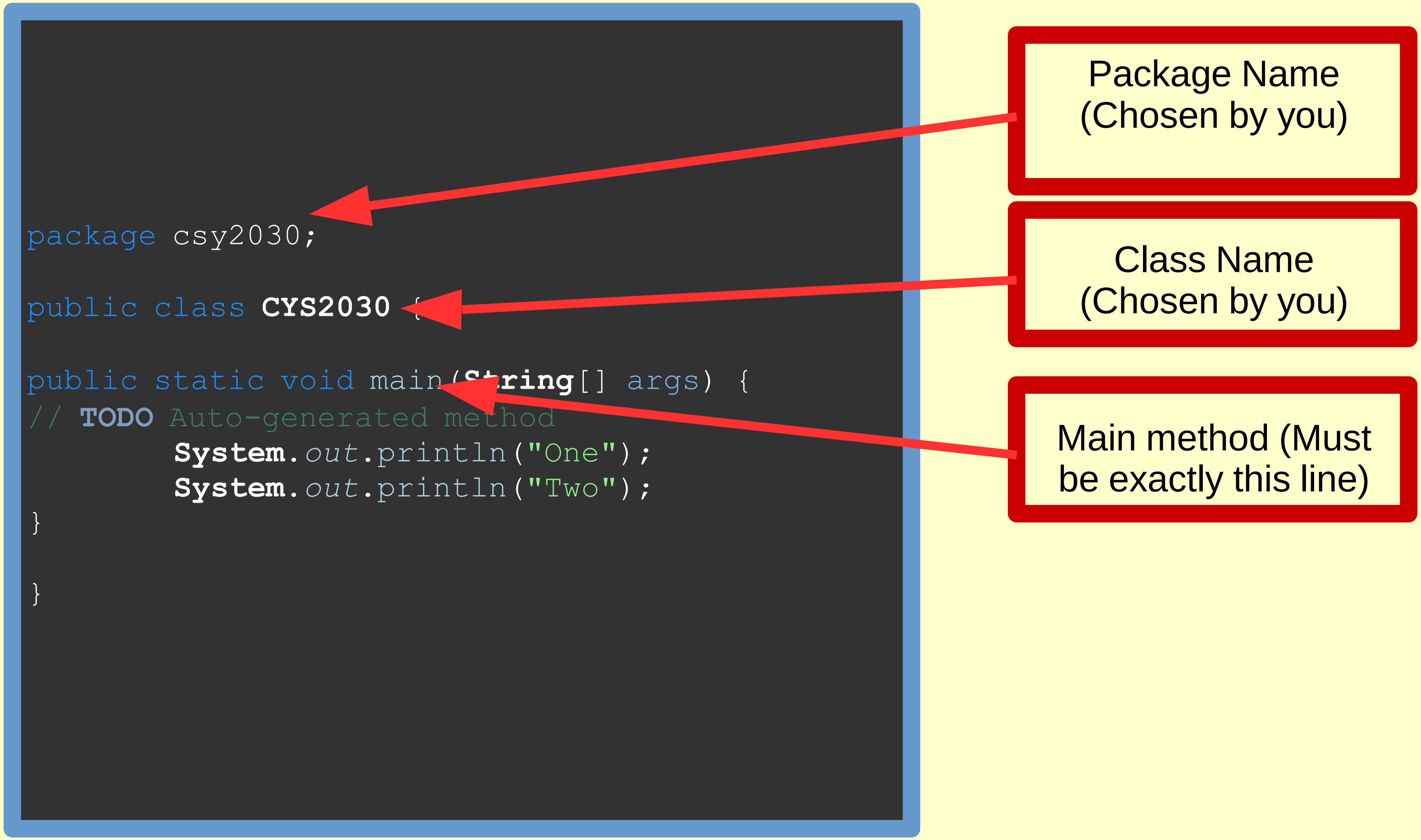
Overview of Lecture

- Today we will revise the following in Java:
  - Structure
  - Types of errors
  - Names and identifiers
  - Comments
  - Primitive data types
  - Constants
  - Methods

Structure of a Java Program

- A Java program consists of at minimum:
  - One class
  - A class must have the same name as the file it's stored in. e.g. *MyClass.java* contains a class named *MyClass*.
    - If the class name and the file name do not match, the program will not run
  - A static method called main (Eclipse can generate this method for you)

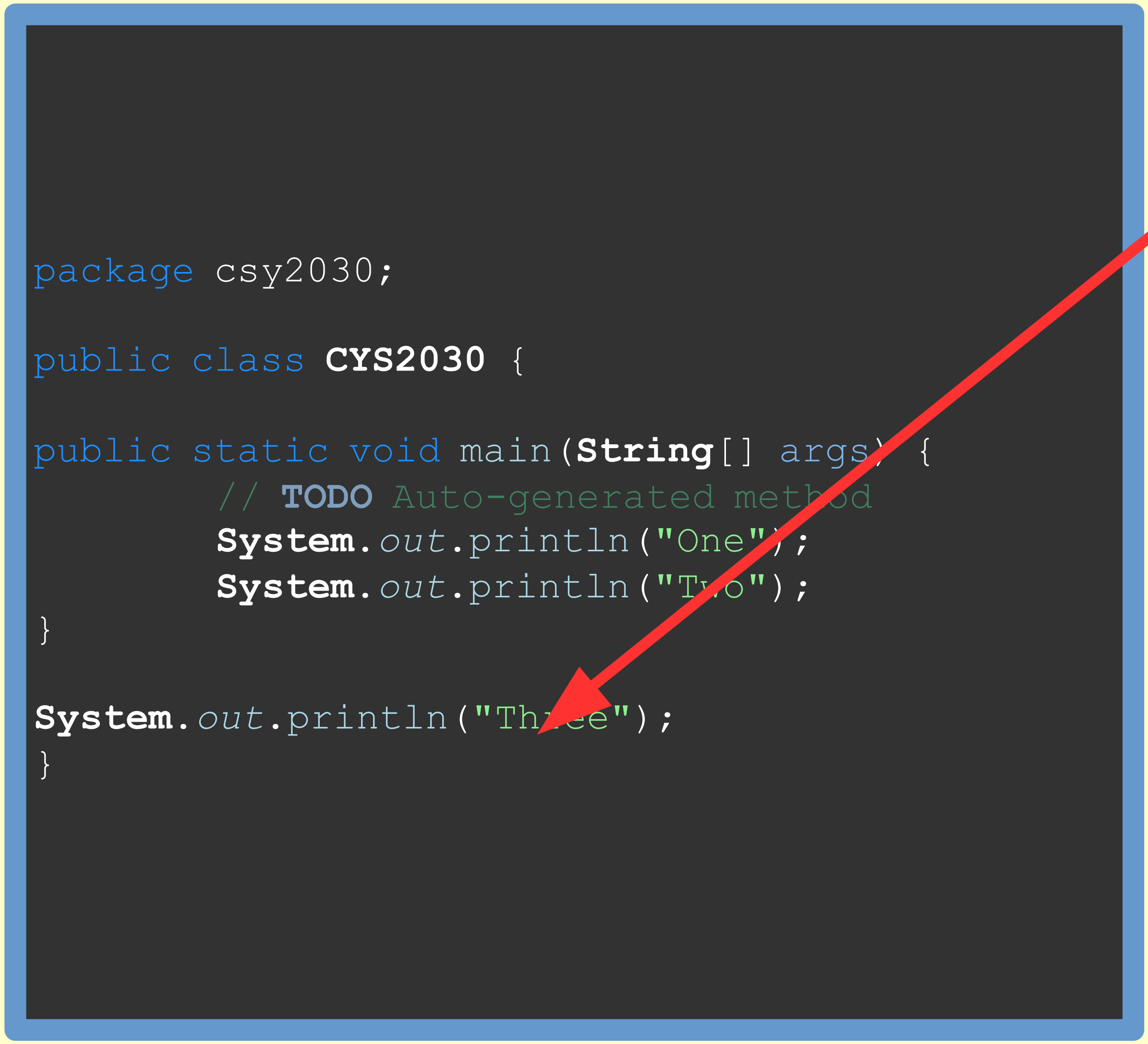
Structure of a Java Program



```
package csy2030;

public class CYS2030 {

public static void main(String[] args) {
// TODO Auto-generated method
        System.out.println("One");
        System.out.println("Two");
}

}
```

Package Name (Chosen by you)
Class Name (Chosen by you)
Main method (Must be exactly this line)

Structure of a Java Program



```
package csy2030;

public class CYS2030 {

public static void main(String[] args) {
        // TODO Auto-generated method
        System.out.println("One");
        System.out.println("Two");
}

System.out.println("Three");
}
```

Statements must be be placed inside a method. This line is valid code but it cannot be placed here as it's not in a method.

This program will not run

Structure of a Java Program

```
package csy2030;

public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("One");
        System.out.println("Two");
}
```

Methods must be placed inside a class. This is not a valid program because there is no class

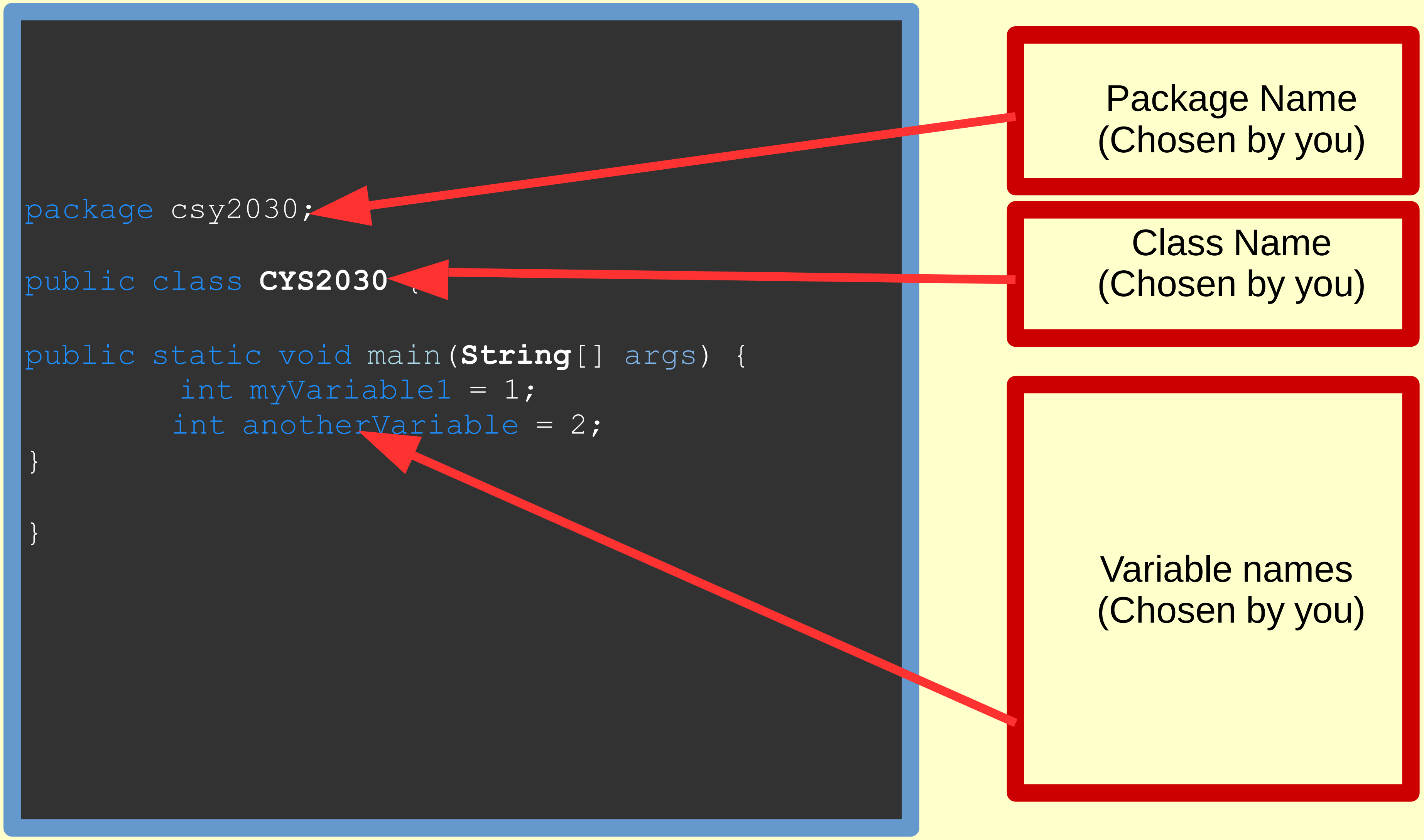
Type of Errors in Java

- There are three main types of error that will stop your program working as you intended:
  - Syntax errors
    - These are errors in the code. E.g. missing braces, missing semicolons, and trying to use a variable that hasn't been defined.
  - Run-Time errors
    - A run-time error is an error that occurs while the program is running and usually causes the program to crash
      - e.g. dividing by zero
  - Logical errors
    - A logical error is an error in the logic of the code. The computer is doing exactly what you told it to but what you told it to do was wrong. E.g. trying to store a decimal in an int e.g *int x = 5/2;*

Names and Identifiers

- As a programmer, you declare a lot of the vocabulary
- Some keywords are defined for you (E.g. *if*, *for*, *while*, *class*, *int*, *String*, etc)
- Mostly you will name everything you use.
- All classes, variables and methods must have a name.
  - The name is decided by the programmer (you)

Names and Identifiers



```
package csy2030;

public class CYS2030 {

public static void main(String[] args) {
          int myVariable1 = 1;
          int anotherVariable = 2;
}

}
```

Names and Identifiers

- Class names should start with a capital letter and an uppercase letter for each word.
  - E.g use **MyPrinterClass** instead of **myprinterclass** or **myPrinterClass**
  - This is a convention and not enforced by Java itself
- Variables should start with a lowercase letter and capitalise the first letter of each word. E.g. **myVariable**

Names and Identifiers

- Identifiers cannot start with a number
  - Identifiers may not use any symbols that have special meaning in Java:

    -

    +

    /

    \

    #

    *

    (

)

    {

    }

    .

Comments

- Comments can be included in your code to explain what it is doing
- Comments are not processed by the computer
- The contents of comments don't have a strict structure and can contain any text
- Comments do not have an impact on the program.
  - It will run exactly the same whether they are included or not

Comments

- Comments serve two main purposes:
  1) To allow you to document the code
  2) To easily “turn off” lines of code without removing them entirely - this is known as “commenting out”

Comments - To document code

```
package csy2030;

public class CYS2030 {
    public static void main(String[] args)  {
    //      This is a single line comment
        /* This is

             a
  multi-line comment
  */

  }

  }
```

Comments – Commenting out code

```
package csy2030;

public class CYS2030 {

public static void main(String[] args) {
// System.out.println(“test”);


}


}
```

This line will not be executed

Primitive Data Types

- There are eight primitive data types in Java
  - Four of them represent integers
    - *byte, short, int, long*
  - Two of them represent floating point numbers
    - *float, double*
  - One of them represents characters
    - *char*
  - And one of them represents boolean values
    - *boolean*

Primitive Data Types

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

Type	Storage	Min Value	Max Value
`byte`	8 bits	-128	127
`short`	16 bits	-32,768	32,767
`int`	32 bits	-2,147,483,648	2,147,483,647
`long`	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
`float`	32 bits	$+/- 3.4 \times 10^{38}$ with 7 significant digits	
`double`	64 bits	$+/- 1.7 \times 10^{308}$ with 15 significant digits	

Primitive Data Types

- A variable can only store a value of its own type.

```
int x = 2.5; // ERROR: incompatible types
```

- However, an *int* value can be stored in a *double* variable.
- The value is converted into the equivalent real number.

```
Double myNum = 5;
System.out.println(myNum);
```

```
5.0
```

Primitive Data Types

- Casting can be used to convert what was invalid:

```
double myDouble = 5.77;
int myInt = myDouble;
```

- Into a valid expression

```
double myDouble = 5.77;
int myInt = (int) myDouble;
```

- However, this will cause a loss of precision
  - myInt will store 5, the .77 will be lost. Note that it does not get rounded, the numbers after the decimal point are just removed

Converting Strings to Integers

- Any value can be stored in a string. e.g.

```
String myString = "1.23";
```

- However, you cannot perform arithmetic on a number when it is stored inside a string

Converting strings to other types

```java
// Store 1 in bVar.
byte bVar = Byte.parseByte("1");

// Store 2599 in iVar.
int iVar = Integer.parseInt("2599");

// Store 10 in sVar.
short sVar = Short.parseShort("10");

// Store 15908 in lVar.
long lVar = Long.parseLong("15908");

// Store 12.3 in fVar.
float fVar = Float.parseFloat("12.3");

// Store 7945.6 in dVar.
double dVar = Double.parseDouble("7945.6");
```

Constants

- Constants, like variables, store values
- However, the value of a constant cannot be changed once it's been defined
- By convention, constants are declared in uppercase with underscores separating the words.
  - e.g. MY_CONSTANT
  - This is so anyone looking at the code can quickly tell if the code is referencing a variable or a constant but has no meaning to the computer
- Constants are declared as part of the class
- Syntax is:

```
      public static final data-type constant-name = value;
e.g
      public static final int MY_CONSTANT = 5;
```

Constants

```
package    ConstantExample;

public      class     ConstantExample      {
public      static      final int  MY_CONSTANT = 5;

public      static void main(String[] args) {
            System.out.println(MY_CONSTANT);
}
}
```

```
5.0
```

Methods

- You must provide a main method as an entry point
- You can also define your own methods
- To declare a method, use the code:

```
public    static    void    methodName()    {
    //Code    goes    here
}
```

Methods

- A method is "called"
  - this is done using the method name followed by opening and closing brackets e.g *methodName();*

```
public  class  MethodExample   {

public  static void main(String[] args) {
        methodName();
}

public  static  void methodName() {
        System.out.println("Method Called");
}

}
```

```
Method Called
```

Methods

- Methods can be called multiple times e.g

```
public  class  MethodExample   {

public  static void main(String[] args) {
        methodName();
        methodName();
        methodName();
}

public  static void methodName() {
        System.out.println("Method  Called");
}
}
```

```
Method Called
Method Called
Method Called
```

Methods

- You can declare multiple methods e.g

```
public     class     MethodExample {

public     static    void main(String[] args) {
           methodName1();
           methodName2();
}

public     static     void  methodName1() {
           System.out.println("Method1 Called");
}

public     static     void  methodName2() {
           System.out.println("Method2 Called");
}

}
```

```
Method1 Called
Method2 Called
```

Methods and Variables

- Class variables can be declared that can be used in any method in the class

```
public     class    MethodExample{

public     static   int myVariable = 3;

public     static   void main(String[] args) {
           methodName1();
           methodName2();
}

public     static    void  methodName1() {
           System.out.println(myVariable);
}

public     static    void  methodName2() {
           System.out.println(myVariable);
}

}
```

```
3
3
```

Methods and Variables

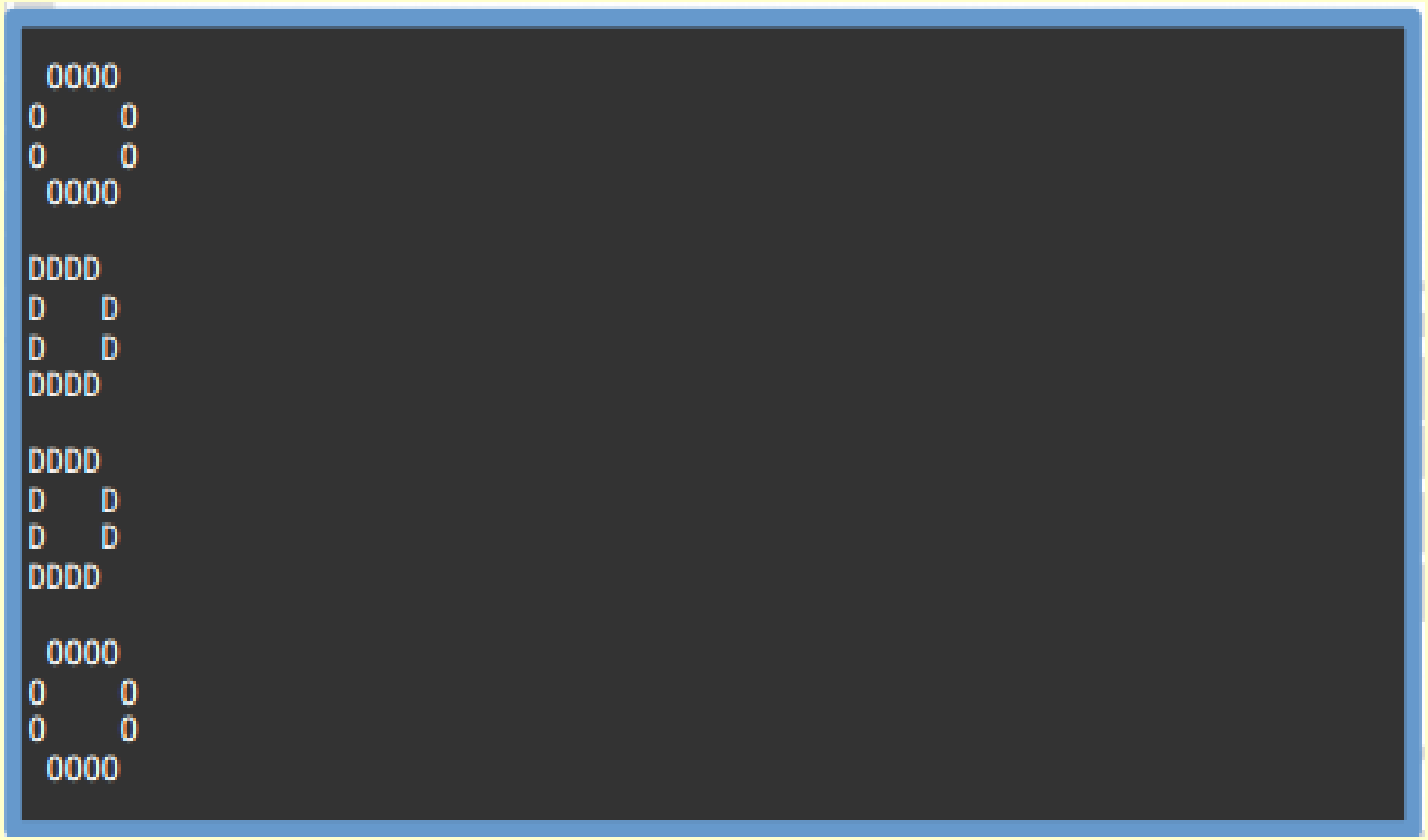
- If the variable is changed in a method, it will be reflected in other methods

```
public     class    MethodExample        {

public     static   int myVariable = 3;

public     static   void main(String[]  args) {
           methodName1();
           methodName2();
           methodName1();
}

public     static    void  methodName1() {
           System.out.println(myVariable);
}

public     static    void  methodName2() {
           myVariable = myVariable + 2;
}

}
```

```
3
5
```

Methods and Variables

- Methods can be used to remove the repeated code. How could you display the following output?



```
 OOOO
O    O
O    O
 OOOO

DDDD
D   D
D   D
DDDD

DDDD
D   D
D   D
DDDD

 OOOO
O    O
O    O
 OOOO
```

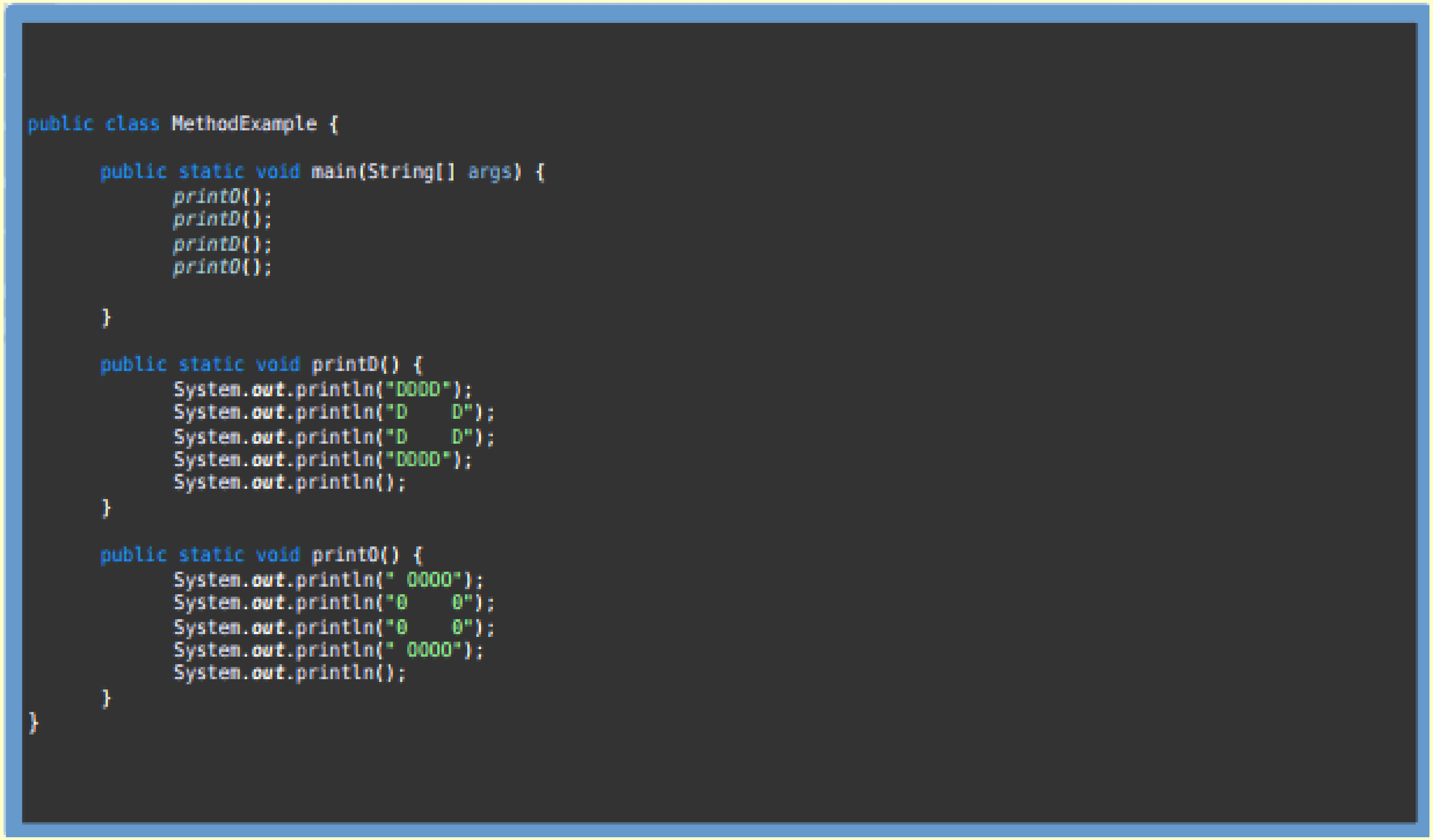
Methods and Variables

- One way is to have multiple print statements but this results in repeated code i.e

```java
public class MethodExample {

public static void main(String[] args) {
            System.out.println(" OOOO");
            System.out.println("0  0");
            System.out.println("0  0");
            System.out.println(" OOOO");
            System.out.println();


            System.out.println("DDDD");
            System.out.println("D D");
            System.out.println("D D");
            System.out.println("DDDD");
            System.out.println();

            System.out.println("DDDD");
            System.out.println("D D");
            System.out.println("D D");
            System.out.println("DDDD");
            System.out.println();

            System.out.println(" OOOO");
            System.out.println("0  0");
            System.out.println("0  0");
            System.out.println(" OOOO");
            System.out.println();
            }
            }
```

Methods and Variables

- Better to have a method for each letter and make multiple calls i.e



```java
public class MethodExample {

	public static void main(String[] args) {
		printO();
		printD();
		printD();
		printO();

	}

	public static void printD() {
		System.out.println("DDDD");
		System.out.println("D    D");
		System.out.println("D    D");
		System.out.println("DDDD");
		System.out.println();
	}

	public static void printO() {
		System.out.println(" OOOO");
		System.out.println("O    O");
		System.out.println("O    O");
		System.out.println(" OOOO");
		System.out.println();
	}
}
```