Why Use Separate Classes and a Driver in Java

Building clear, modular Java programs

Produced by: Ms Mairead Meagher
Ms Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Department of Computing and Mathematics
setu.ie

Organising Your Code

- Everything in Java is a class
- Class with methods → contains logic, tasks, reusable functionality
- Driver class → contains main(), runs the program
- Separating them keeps responsibilities clear and code structured

Why Separate Them

- Cleaner, modular code
- Easier testing of methods
- Methods can be reused in other programs
- main() becomes a simple controller

“Think of your class as the **toolbox**, and the Driver as the **person using the tools**.”

- The Toolbox must exist before the person can use the tools

Class

= Toolbox

Driver

= Person using the tools

How They Interact (Code Example)

Calculator.java

```
public class Calculator {
    public int add(int x, int y)
{
        return x + y;
    }
}
```

CalculatorDriver.java

```
public class CalculatorDriver {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        int result = calc.add(5, 3);
        System.out.println("Sum = " +
result);
    }
}
```

1. Create Object
2. Invoke Method

Why Create an Object?

- Methods belong to the class

- You need an object to access those methods
  - Class = blueprint
  - Object = usable thing

Another way to set up Driver – calling Internal methods in Driver

```
public class Driver {

   private Scanner input = new Scanner (System.in);

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

   public Driver(){
//Uncomment the line with the method you want to run.
      exercise1();
      //exercise2();

      exercise16();
   }
```

```
private void exercise1() { //also written in Driver
      System.out.println("Exercise 1: reads in 6 integers into
an integer array and prints the values to the screen.");

      int[] numbers = new int[6];

      for (int i = 0; i < numbers.length ; i++) {
         System.out.print("Enter value : ");
         numbers[i] = input.nextInt();
      }

      for (int i = 0; i < numbers.length ; i++) {
         System.out.println("value " + (i+1)  + " is: " +
numbers[i]);
      }
   }
```

1. Call Driver (private) methods in the constructor

2. Simply call the constructor in the main method.

Questions?