Menu Driven Apps

More on loops, the switch statement and menus

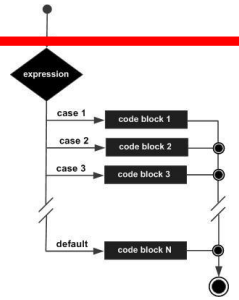
Produced by: Ms. Mairéad Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Topics list

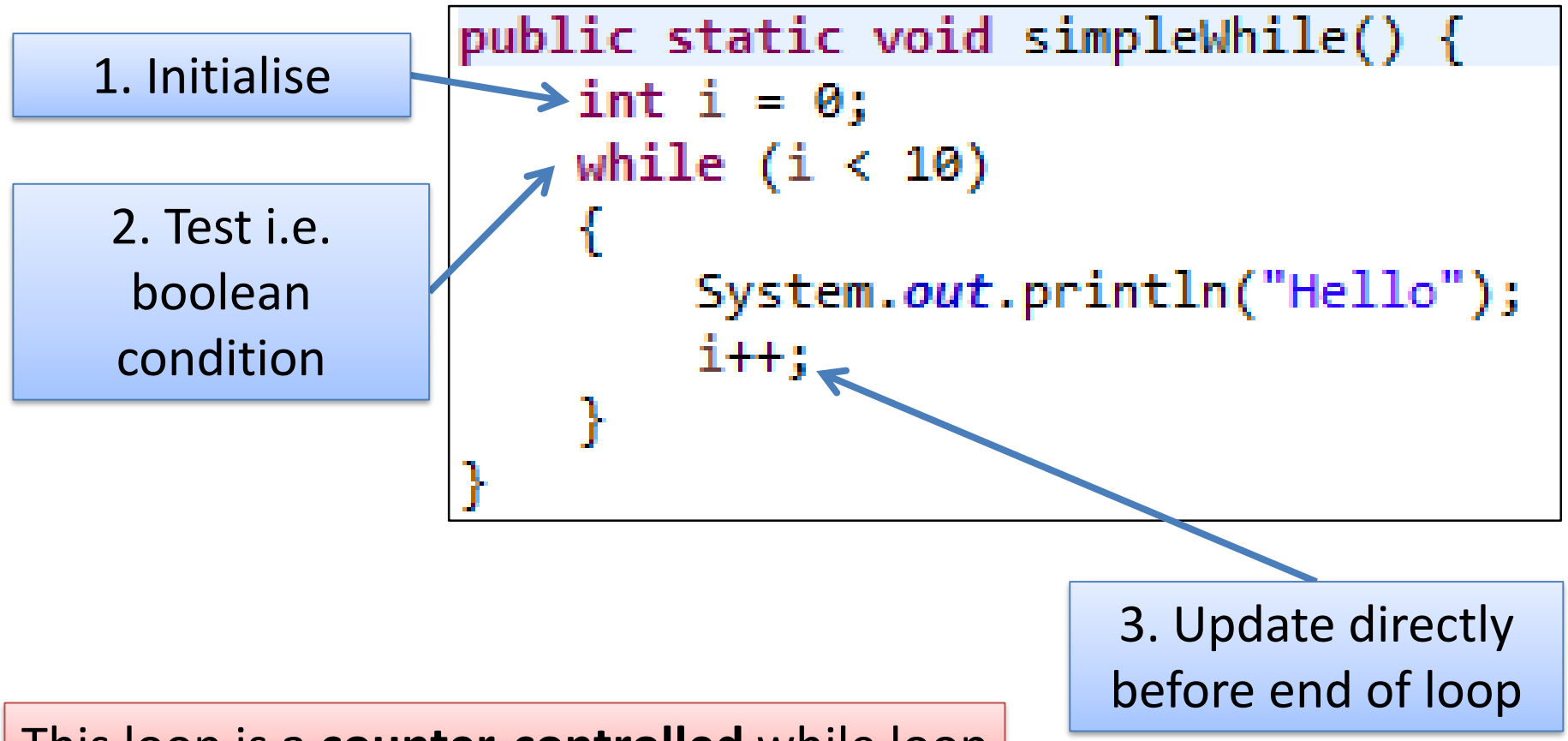
1. **Loops**
   - while, for, for each
   - Loop Control Variables (**LCV**)
   - Arrays and **counter controlled** loops
   - Arrays and **sentinel** based loops
   - Arrays and **flag**-based loops



2. **switch** statement

3. **Menus**
   - Shop V3.0 menu system

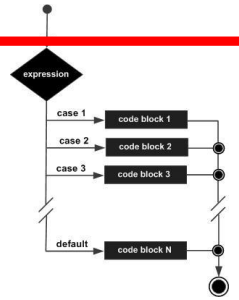
Recap - Loop Control Variable



This loop is a **counter-controlled** while loop

Topics list

1. **Loops**
   - while, for, for each
   - Loop Control Variables (**LCV**)
   - Arrays and **counter controlled** loops
   - Arrays and **sentinel** based loops
   - Arrays and **flag**-based loops



2. **switch** statement

3. **Menus**
   - Shop V3.0 menu system

Recap - Counter-Controlled **for** Loop

```
public static void loopWithArrayExample() {
    int[]numbers = new int[10];     //array is a local variable
    int sum = 0;

    for (int i = 0; i < 5; i++)
    {
        System.out.print ("Please enter a number : ");
        numbers[i] = input.nextInt();
        sum += numbers[i];
    }

    System.out.println("The sum of the numbers you typed in is : " + sum);

}
```

Recap - Counter-Controlled Loops

- Sometimes we know when we are coding **compile-time**, **how many** inputs we will have.
  - The previous slide displays an example of this.

- Other times, we find out at **run-time** **how many** inputs we have
  - an example of this is on the next slide.

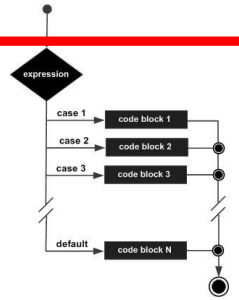
Recap - Counter-Controlled **for** Loop

```
public static void loopWithArrayVarSizeExample() {
    int[]numbers = null;
    int numNumbers = 0;
    int sum = 0;

    System.out.print ("How many numbers would you like to enter?  : ");
    numNumbers = input.nextInt();
    numbers = new int[numNumbers];

    for (int i = 0; i < numNumbers; i++)
    {
        System.out.print ("Please enter a number : ");
        numbers[i] = input.nextInt();
        sum += numbers[i];
    }

    System.out.println("The sum of the numbers you typed in is : " + sum);

}
```

Here, we know at **run-time** how many inputs we have.

Topics list

1. **Loops**
   - while, for, for each
   - Loop Control Variables (**LCV**)
   - Arrays and **counter controlled** loops
   - Arrays and **sentinel** based loops
   - Arrays and **flag**-based loops



2. **switch** statement

3. **Menus**
   - Shop V3.0 menu system

Sentinel-based loops

- Use this type of loop when you **DON'T know how many** inputs you will have.

- The ***end of input*** is signalled by a special value.
  - e.g.
    - if you are calculating the 'average of ages of people in the room':
      - write a program that will continually take in ages until, say, **-1** is entered.
    - **-1** would be the **sentinel value**.

Structure

- Concept of ***Loop Control Variable*** is vital here.
- The loop continuation is solely based on the input, so the variable containing the information is the Loop Control Variable.
- Initialise the Loop Control Variable before entry into the loop.
- Remember to ‘update the Loop Control Variable’ just before the end of the loop.

Try this exercise

- Write a loop to read in and add up a set of integers. Keep going until the value '**-1**' is inputted.

- What is your Loop Control Variable?

Note: *for this exercise, don't store the values in an array… we'll do that in a few slides time.*

Solution

```
public static void sentinelWhileLoop()
{
    int sum = 0;

    System.out.print("Enter a number, -1 ends input: ");
    int n = input.nextInt();

    while (n != -1)
    {
        sum += n;
        System.out.print("Enter a number, -1 ends input: ");
        n = input.nextInt();
    }
    System.out.println("The total is: " + sum);
}
```

1. Initialise

2. LCV Condition

3. Update LCV directly before end of loop

Next step in the exercise

- We need to record how many inputs have happened.

- Change the previous solution
  so that you know at the end,
  how many numbers have been inputted.

- At the end, print the sum and number of inputs.

Code with number of inputs

```
public static void sentinelWhileLoopWithCounter()
{
    int sum = 0, counter = 0;

    System.out.print("Enter a number, -1 ends input: ");
    int n = input.nextInt();

    while (n != -1)
    {
        sum += n;
        System.out.print("Enter a number, -1 ends input: ");
        n = input.nextInt();
        counter++;
    }
    System.out.println("The total is: " + sum);
    System.out.println("The number of items entered is: " + counter);
    }
```

Try this now - using **arrays**

- Re-write the code on the previous slide, but **store the data in an array**.
  - NOTE:
    - Assume the max number of inputs possible is 100 (i.e. size of array).

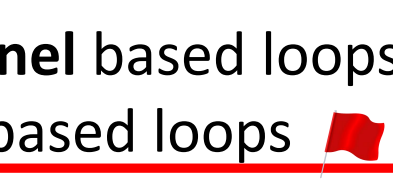
- We also need to know
  - **how many inputs** actually happened.

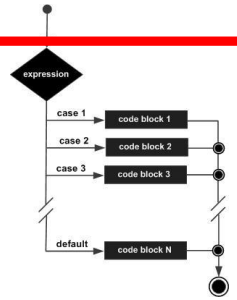
Solution – storing inputs

```
public static void sentinelWhileLoopWithArrays()
{
    int sum = 0, counter = 0, size = 100;
    int numbers [] = new int[size];

    System.out.print("Enter a number, -1 ends input: ");
    int n = input.nextInt();

    while (n != -1 && counter < size) //ensures that you don't go over max size of array
    {
        numbers[counter] = n;
        sum += n;
        System.out.print("Enter a number, -1 ends input: ");
        n = input.nextInt();
        counter++;
    }
    System.out.println("The total is: " + sum);
    System.out.println("The number of items entered is: " + counter);

    for (int i = 0; i < counter; i++)
    {
        System.out.println("    Number entered: " + numbers[i]);
    }
}
```

Topics list

1. **Loops**
   - while, for, for each
   - Loop Control Variables (**LCV**)
   - Arrays and **counter controlled** loops
   - Arrays and **sentinel** based loops
   - Arrays and **flag**-based loops





2. **switch** statement

3. **Menus**
   - Shop V3.0 menu system

Flag-Based Loops

- These are used when you want to
  - examine a collection of data
  - to ***check for a property***.
  - Once a property has been established, it cannot be 'unestablished'.

- 'Once the flag is raised, it cannot be taken down'

Flag-Based Loops - **example**

- Given a populated array of numbers, cycle over the array to see if any numbers are odd.

- If you find:
  - At least one odd number,
    - print out to the user that there is at least one odd number.
  - No odd numbers,
    - inform the user of this.

Solution: check if ‘any numbers odd’

```
public static void flagBasedLoopWithArray()
{
    int numbers[] = {4,6,8,7,10,12};
    boolean oddNumberInArray = false;

    for (int number : numbers)
    {
        if (number % 2 == 1)
        {
            oddNumberInArray = true;
        }
    }

    if (oddNumberInArray == true)
    {
        System.out.println("There is at least one odd number in the array.");
    }
    else
    {
        System.out.println("There is NO odd number in the array.");
    }
}
```

Better code…

```
public static void flagBasedLoopWithArray()
{
    int numbers[] = {4,6,8,7,10,12};
    boolean oddNumberInArray = false;

    for (int number : numbers)
    {
        if (number % 2 == 1)
        {
            oddNumberInArray = true;
        }
    }

    if (oddNumberInArray)
    {
        System.out.println("There is at least one odd number in the array.");
    }
    else
    {
        System.out.println("There is NO odd number in the array.");
    }
}
```

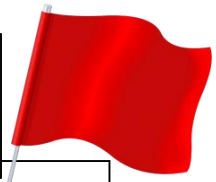
Use of boolean variable in condition

What about having a flag-based loop

in a method

with a boolean return type?

Code with boolean return type



```
public static boolean flagBasedLoopWithArrayReturn()
{
    int numbers[] = {4,6,8,7,10,12};
    boolean oddNumberInArray = false;

    for (int number : numbers)
    {
        if (number % 2 == 1)
        {
            oddNumberInArray = true;
        }
    }

    return oddNumberInArray;
}
```

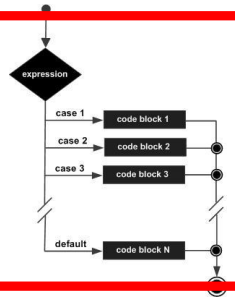
Calling the method and handling the returned boolean

```
if (flagBasedLoopWithArrayReturn())
    System.out.println("There is at least one odd number in the array");
else
    System.out.println("There is NO odd number in the array");
```

Topics list

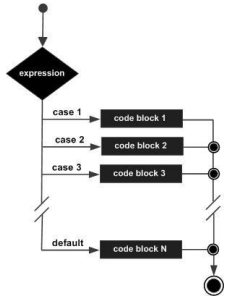
1. **Loops**
   - while, for, for each
   - Loop Control Variables (**LCV**)
   - Arrays and **counter controlled** loops
   - Arrays and **sentinel** based loops
   - Arrays and **flag**-based loops

switch statement



3. **Menus**
   - Shop V3.0 menu system

The **switch** statement

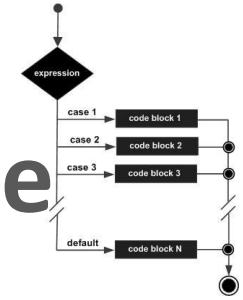


- The switch statement works in exactly the same way as a **set of if** statements, but is more compact and readable.

- The *switch statement* switches on a single **value** to one of an arbitrary number of **cases**.

- Two possible patterns of use are…

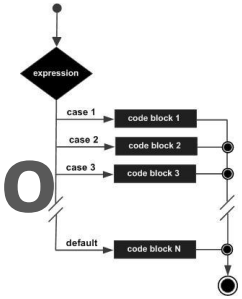
The **switch** statement – pattern **one**




```
switch(expression) {
    case value: statements;
                break;
    case value: statements;
                break;
    further cases possible
    default: statements;
             break;
}
```

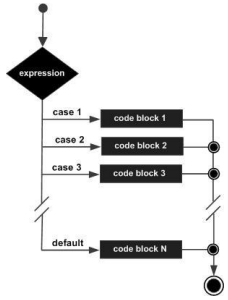
Note: We used this pattern in our ShopV3.0 menu system

The **switch** statement – pattern **two**




```
switch(expression) {
      case value1:
      case value2:
      case value3:
          statements;
          break;
      case value4:
      case value5:
          statements;
          break;
      further cases possible
      default:
          statements;
          break;
 }
```

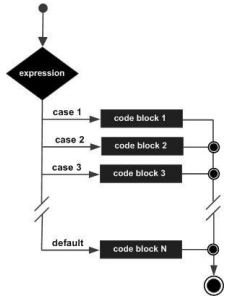
Recap: The switch statement



- A *switch* statement can have any number of **case** labels.

```
switch(expression) {
    case value: statements;
                break;
    case value: statements;
                break;
     further cases possible
     default: statements;
              break;
}
```

The switch statement



The **break** statement after every case is needed,

otherwise the execution *"falls through"* into the next label's statements.

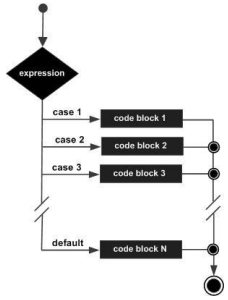
Pattern **two** makes use of this.

All three of the first values (cases) will execute the first *statements* section,

Values (cases) four and five will execute the second *statements* section.

```
switch(expression) {
    case value1:
    case value2:
    case value3:
        statements;
        break;
    case value4:
    case value5:
        statements;
        break;
    further cases possible
    default:
        statements;
        break;
}
```

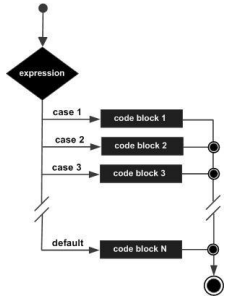
Recap: The switch statement



- The **default** case is optional.
- If no default is given, it may happen that no case is executed.

```
switch(expression) {
    case value: statements;
                break;
    case value: statements;
                break;
    further cases possible
    default: statements;
             break;
}
```

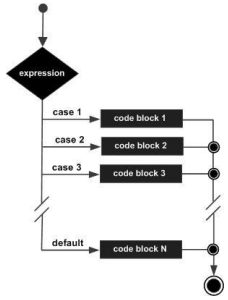
Recap: The switch statement



- The **break** statement after the default
  (or the last case, if there is no default)
  is not needed but is considered good style.

```
switch(expression) {
    case value: statements;
                break;
    case value: statements;
                break;
    further cases possible
    default: statements;
             break;
}
```

Recap: The switch statement

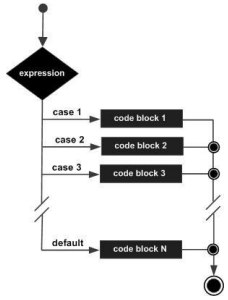


- Pre Java 7,
  the **expression** used to switch on, and the case **labels (value)** are **char** or **int**.

- From Java 7 onwards,
  you can switch on **String**.

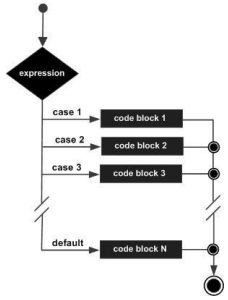
```
switch(expression) {
    case value: statements;
                break;
    case value: statements;
                break;
    further cases possible
    default: statements;
             break;
}
```

The switch statement – **int** example



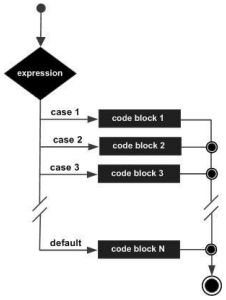

```
switch(day) {
    case 1:  dayString = "Monday";
             break;
    case 2:  dayString = "Tuesday";
             break;
    case 3:  dayString = "Wednesday";
             break;
    case 4:  dayString = "Thursday";
             break;
    case 5:  dayString = "Friday";
             break;
    case 6:  dayString = "Saturday";
             break;
    case 7:  dayString = "Sunday";
             break;
    default: dayString = "invalid day";
             break;
}
```

The switch statement – **char** example




```
switch (group){
    case 'A':
            System.out.println("10.00 a.m ");
            break;
    case 'B':
            System.out.println("1.00 p.m ");
            break;
    case 'C':
            System.out.println("11.00 a.m ");
            break;
  default:
            System.out.println(“Enter option A, B or C only!");
}
```

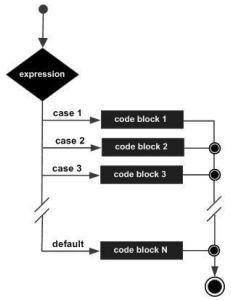
The switch statement – **String** example




```
switch(dow.toLowerCase()) {
    case "mon":
    case "tue":
    case "wed":
    case "thu":
    case "fri":
        goToWork();
        break;
    case "sat":
    case "sun":
        stayInBed();
        break;
}
```

Topics list

1. **Loops**
   - while, for, for each
   - Loop Control Variables (**LCV**)
   - Arrays and **counter controlled** loops
   - Arrays and **sentinel** based loops
   - Arrays and **flag**-based loops



2. **switch** statement

3. **Menus**
   - Shop V3.0 menu system

```
Shop Menu
---------
  1) Add a product
  2) List the Products
  --------------------
  3) List the current products
  4) Display average product unit cost
  5) Display cheapest product
  6) List products that are more expensive than a given price
  --------------------
  0) Exit
==>>
```

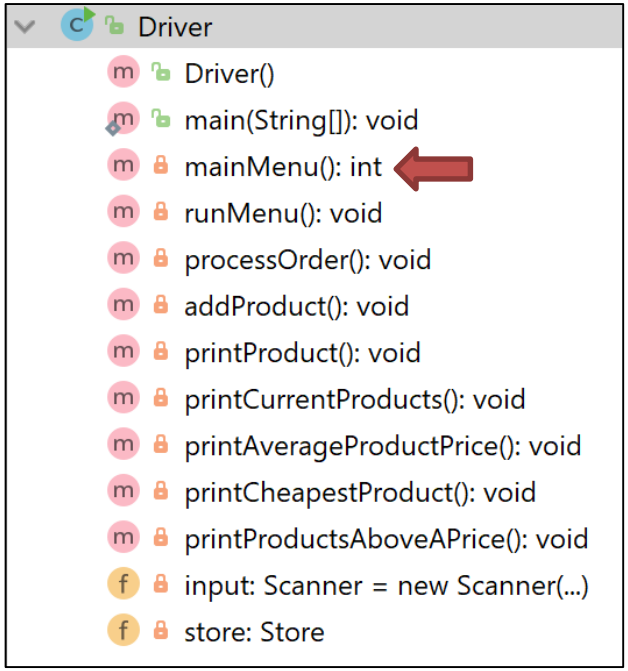
Shop V3.0 – menu to be displayed.

1

```java
private int mainMenu(){
    System.out.println("Shop Menu");
    System.out.println("---------");
    System.out.println("  1) Add a product");
    System.out.println("  2) List the Products");
    System.out.println("  --------------------");
    System.out.println("  3) List the current products");
    System.out.println("  4) Display average product unit cost");
    System.out.println("  5) Display cheapest product");
    System.out.println("  6) List products that are more expensive than a given price");
    System.out.println("  --------------------");
    System.out.println("  0) Exit");
    System.out.print("==>> ");
    int option = input.nextInt();
    return option;
}
```

Shop V3.0 - mainMenu() method.

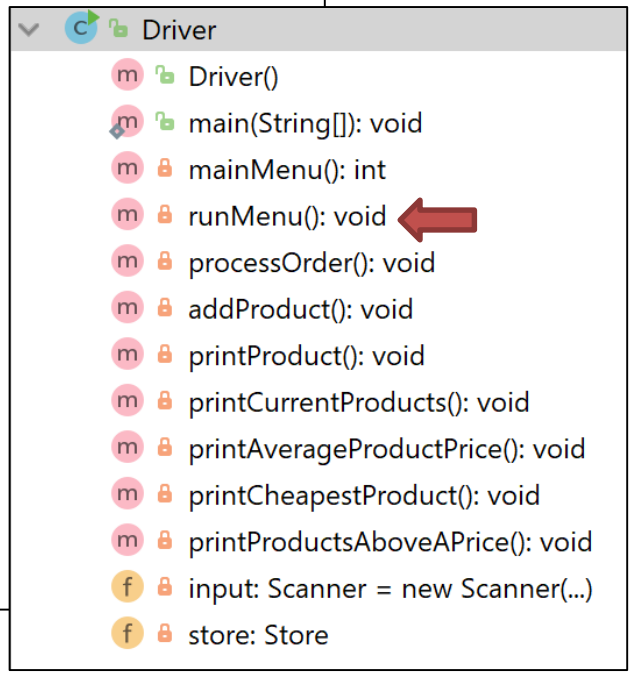
2



Shop V3.0 - runMenu() method.

3

```
private void runMenu(){
    int option = mainMenu();
    while (option != 0){

        switch (option){
            case 1:     addProduct();
                break;
            case 2:     printProduct();
                break;
            case 3:     printCurrentProducts();
                break;
            case 4:     printAverageProductPrice();
                break;
            case 5:     printCheapestProduct();
                break;
            case 6:     printProductsAboveAPrice();
                break;
            default:    System.out.println("Invalid option entered: " + option);
                break;
        }

        //pause the program
        System.out.println("\nPress any key to continue...");
        input.nextLine();
        input.nextLine();  //for the bug in Scanner

        //display the main menu again
        option = mainMenu();
    }

    //the user chose option 0, so exit the program
    System.out.println("Exiting... bye");
    System.exit(0);
}
```



```
private void runMenu(){
    int option = mainMenu();
    while (option != 0){

        switch (option){
            case 1:     addProduct();
                break;
            case 2:     printProduct();
                break;
            case 3:     printCurrentProducts();
                break;
            case 4:     printAverageProductPrice();
                break;
            case 5:     printCheapestProduct();
                break;
            case 6:     printProductsAboveAPrice();
                break;
            default:    System.out.println("Invalid option entered: " + option);
                break;
        }

        //pause the program
        System.out.println("\nPress any key to continue...");
        input.nextLine();
        input.nextLine();  //for the bug in Scanner

        //display the main menu again
        option = mainMenu();
    }

    //the user chose option 0, so exit the program
    System.out.println("Exiting... bye");
    System.exit(0);
}
```

LCV initialised

LCV tested

LCV changed

3

Shop V3.0 - runMenu() method.

Loop Control Variable is **option**

Shop V3.0 - runMenu() method.

3

```
private void runMenu(){
    int option = mainMenu();
    while (option != 0){

        switch (option){
            case 1:     addProduct();
                break;
            case 2:     printProduct();
                break;
            case 3:     printCurrentProducts();
                break;
            case 4:     printAverageProductPrice();
                break;
            case 5:     printCheapestProduct();
                break;
            case 6:     printProductsAboveAPrice();
                break;
            default:    System.out.println("Invalid option entered: " + option);
                break;
        }

        //pause the program
        System.out.println("\nPress any key to continue...");
        input.nextLine();
        input.nextLine();  //for the bug in Scanner

        //display the main menu again
        option = mainMenu();
    }

    //the user chose option 0, so exit the program
    System.out.println("Exiting... bye");
    System.exit(0);
}
```

Boolean evaluates to false, loop exited

Code for exiting the program

Option 0 selected

Shop V3.0 - runMenu() method.

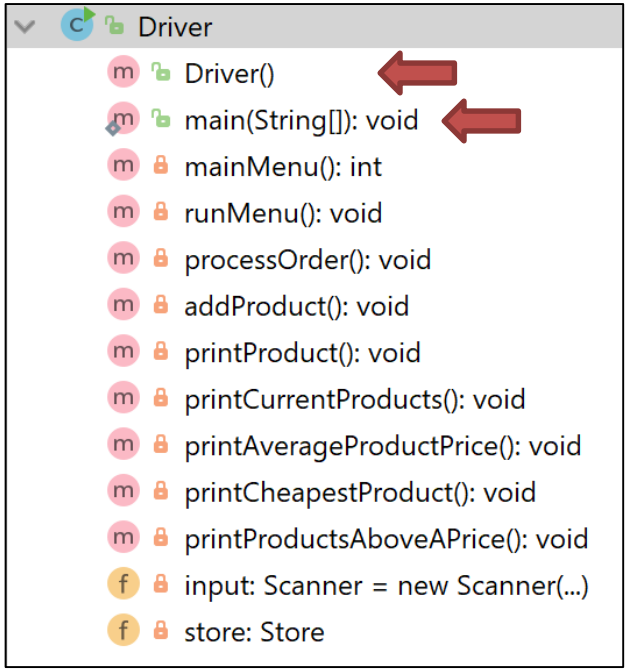
3

```
private void runMenu(){
    int option = mainMenu();
    while (option != 0){

        switch (option){
            case 1:     addProduct();
                break;
            case 2:     printProduct();
                break;
            case 3:     printCurrentProducts();
                break;
            case 4:     printAverageProductPrice();
                break;
            case 5:     printCheapestProduct();
                break;
            case 6:     printProductsAboveAPrice();
                break;
            default:    System.out.println("Invalid option entered: " + option);
                break;
        }

        //pause the program
        System.out.println("\nPress any key to continue...");
        input.nextLine();
        input.nextLine();  //for the bug in Scanner

        //display the main menu again
        option = mainMenu();
    }

    //the user chose option 0, so exit the program
    System.out.println("Exiting... bye");
    System.exit(0);
}
```

Pause the program so the user can read what we just printed to the console

4 Shop V3.0 – call the runMenu() method.

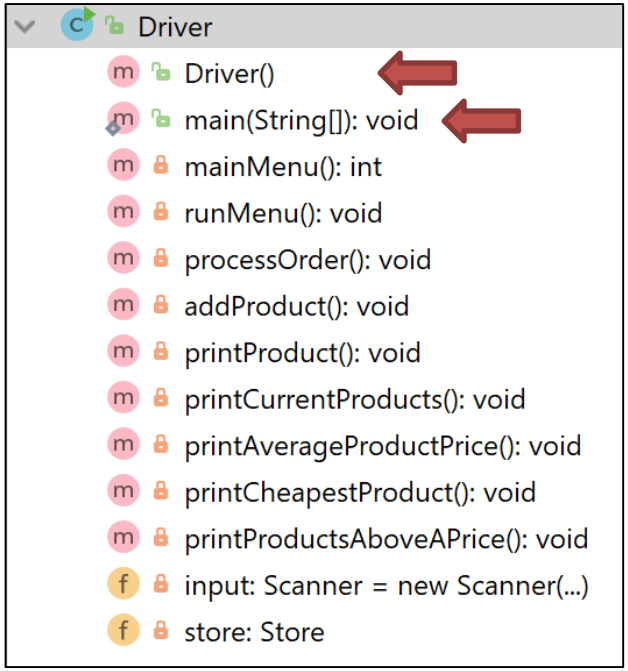
```
public class Driver{

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

    public Driver() {
        store = new Store();
        runMenu();
    }

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



4 Shop V3.0 – call the runMenu() method.

```
Shop Menu
---------
  1) Add a product
  2) List the Products
  --------------------
  3) List the current products
  4) Display average product unit cost
  5) Display cheapest product
  6) List products that are more expensive than a given price
  --------------------
  0) Exit
==>>
```

```
public class Driver{

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

    public Driver() {
        store = new Store();
        runMenu();
    }

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



Questions?