

Menu Driven Systems

While loops, menus and the switch statement

Produced Mairead Meagher
by: Dr. Siobhán Drohan



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

Recap - Loop Control Variable

```
public static void simpleWhile() {  
    int i = 0;  
    while (i < 10)  
    {  
        System.out.println("Hello");  
        i++;  
    }  
}
```

Initialise

Condition

Update directly
before end of loop

Topics list

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

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);  
}
```

Counter-Controlled Loops

- Sometimes we know when we are coding, we know 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.

We know at run-time how many inputs we have.

```
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);  
}
```

Topics list

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

Sentinel-based loops

- When we do not know how many inputs we will have.
- End of input is signalled by a special value.
- E.g. 'average of ages of people in the room', if you don't know how many are in the room.

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

- 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: you do not need to store the values in an array; we will do this later.

Solution

```
public static void sentinelWhileLoop()
```

```
{
```

```
    int sum = 0;
```

```
    System.out.print("Enter a number, -1 ends input: ");
```

```
    int n = input.nextInt();
```

Initialise

```
    while (n != -1)
```

```
    {
```

```
        sum += n;
```

```
        System.out.print("Enter a number, -1 ends input: ");
```

```
        n = input.nextInt();
```

```
    }
```

```
    System.out.println("The total is: " + sum);
```

LCV Condition

```
}
```

Update LCV directly
before end of loop

Next step

- We need to record how many inputs have happened.
- Try to change the previous example 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

- Same structure as before, but now we want to store the input.
- 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).
 - Then 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

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

Flag-Based Loops

- These are used when you want to examine a collection of data to check for a property. Once this property has been established, it cannot be 'unestablished'.
- E.g. test an array of numbers to see if any numbers are odd
- 'Once the flag is raised, it cannot be taken down'

Code to check '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.");
    }
}
```

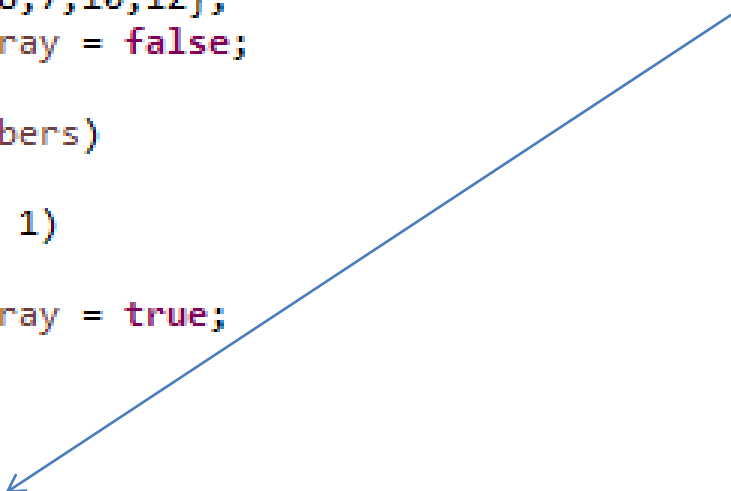
Better code..

Use of boolean variable in condition

```
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.");
    }
}
```



*What about having a
flag-based loop 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

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

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

- One possible pattern of use

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

The switch statement

- Second possible pattern of use

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

The switch statement

- A *switch* statement can have any number of **case** labels.
- The **break** statement after every case is needed, otherwise the execution “falls through” into the next label’s statements. The second form above makes use of this. In this case, all three of the first values will execute the first *statements* section, whereas values four and five will execute the second *statements* section.

The switch statement

- The **default** case is optional. If no default is given, it may happen that no case is executed.
- The **break** statement after the default (or the last case, if there is no default) is not needed but is considered good style.
- From Java 7, the expression used to switch on and the case labels may be strings.

The switch statement - 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 - example

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

The switch statement - 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!");  
}
```

Topics list

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

A simple menu using switch

```
public void run()
{
    System.out.println("Choose a number between 1 and 3");
    int choice = input.nextInt();

    switch(choice)
    {
        case 1:
            System.out.println("You chose 1");
            break;
        case 2:
            System.out.println("You chose 2");
            break;
        case 3:
            System.out.println("You chose 3");
            break;
        default:
            System.out.println("You chose an invalid number");
            break;
    }
}
```

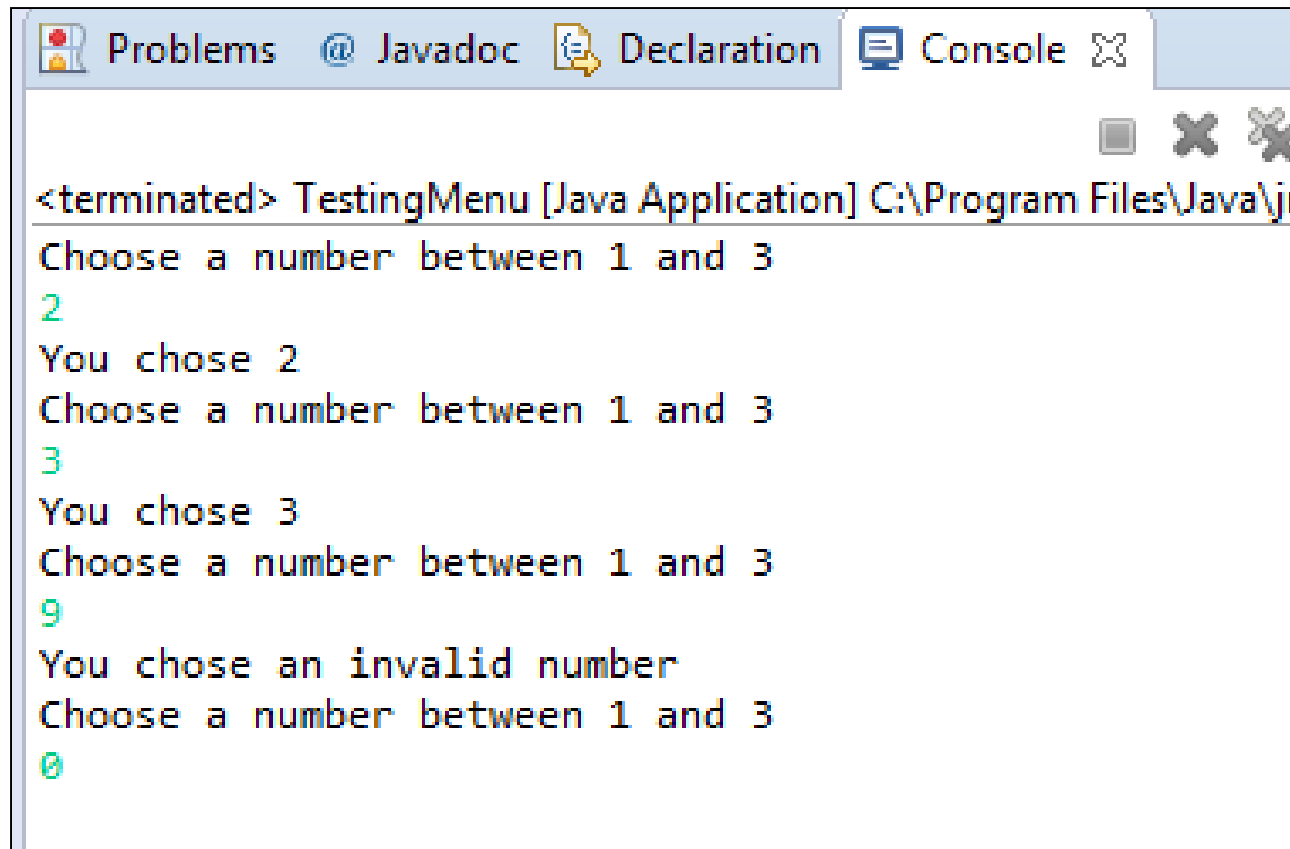
Now loop on the switch statement

```
public void run()
{
    System.out.println("Choose a number between 1 and 3");
    int choice = input.nextInt();

    while (choice != 0)
    {
        switch(choice)
        {
            case 1:
                System.out.println("You chose 1");
                break;
            case 2:
                System.out.println("You chose 2");
                break;
            case 3:
                System.out.println("You chose 3");
                break;
            default:
                System.out.println("You chose an invalid number");
                break;
        }
        System.out.println("Choose a number between 1 and 3");
        choice = input.nextInt();
    }
}
```

Note the use of the
Loop Control Variable

This gives the following output



The screenshot shows a Java IDE window with a tab labeled "Console". The console output is as follows:

```
<terminated> TestingMenu [Java Application] C:\Program Files\Java\j
Choose a number between 1 and 3
2
You chose 2
Choose a number between 1 and 3
3
You chose 3
Choose a number between 1 and 3
9
You chose an invalid number
Choose a number between 1 and 3
0
```

Topics list

- while loops
 - recap on structure – Loop Control Variables
 - Arrays and counter controlled loops
 - Arrays and sentinel based loops
 - Arrays and flag-based loops
- switch statement.
- A simple menu using switch.
- ShopV3.0 – adding a menu.

Adding a basic menu to Shop...

```
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(" 0) Exit");  
    System.out.print("==>> ");  
    int option = input.nextInt();  
    return option;  
}
```

```
private void runMenu(){
    int option = mainMenu();
    while (option != 0){
        switch (option){
            case 1:  addProduct();
                    break;
            case 2:  System.out.println(store.listProducts());
                    break;
            default: System.out.println("Invalid option entered: " + option);
                    break;
        }
    }
```

//pause the program so that the user can read what we just printed to the console

```
System.out.println("\nPress any key to continue...");
```

```
input.nextLine();
```

```
input.nextLine(); //this second read is required - bug in Scanner class; a String read is  
//ignored straight after reading an int.
```

//display the main menu again

```
option = mainMenu();
```

```
}
```

//the user chose option 0, so exit the program

```
System.out.println("Exiting... bye");
```

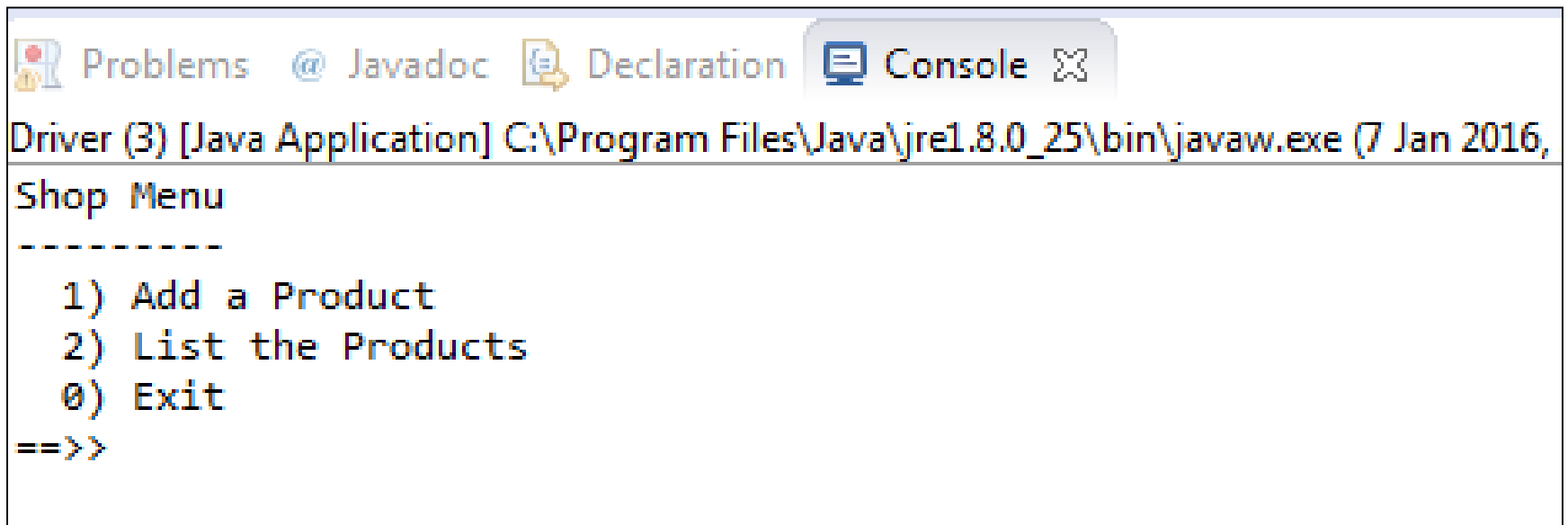
```
System.exit(0);
```

```
}
```

Calling the menu on startup...

```
public static void main(String[] args) {  
    Driver app = new Driver();  
}  
  
public Driver(){  
    input = new Scanner(System.in);  
    store = new Store();  
    runMenu();  
}
```

The displayed menu...



The screenshot shows an IDE window with a tab labeled 'Console'. The console output displays the title bar of a Java application window: 'Driver (3) [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (7 Jan 2016, ...'. Below the title bar, the text 'Shop Menu' is followed by a dashed line separator. A menu is then listed with three options: '1) Add a Product', '2) List the Products', and '0) Exit'. The prompt '==>>' is shown at the bottom of the console.

```
Driver (3) [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (7 Jan 2016, ...
Shop Menu
-----
  1) Add a Product
  2) List the Products
  0) Exit
==>>
```


Questions?

