Menu Driven Apps

Loops, menus and the switch statement

Produced by:
Ms. Maireád Meagher
Dr. Siobhán Drohan
Ms Siobhan Roche

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Topics list

- Loops
  - 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

This loop is a counter-controlled while loop

1. Initialise

2. Test i.e. boolean condition

3. Update directly before end of loop

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

Topics list

- Loops
  - 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 - 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, 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.

Recap - Counter-Controlled for Loop

Here, 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

- Loops
  - 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

- Use this type of loop when you do not 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: in this exercise, you do not need to store the values in an array…we will do this 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.
- Try to 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

```
private void sentinelInWhileLoopWithCounter()
{
    int sum=0;
    int i=0;
    System.out.print("Please enter a number, -1 ends input ");
    int n = input.nextInt();

    while (n != -1) {
        sum += n;
        i++;
        System.out.print("Please enter a number, -1 ends input ");
        n = input.nextInt();
    }

    System.out.print("The total is : " + sum);
    System.out.print("The number of items entered is : " + i);
}
```

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

Solution – storing inputs

```
private void sentinelInWhileLoopWithArrays()
{
    int sum=0;
    int i=0;
    int size = 100;
    int numbers[] = new int[size];
    System.out.print("Please enter a number, -1 ends input ");
    int n = input.nextInt();

    while (n != -1 && i < size) {
        numbers[i] = n;
        sum += n;
        i++;
        System.out.print("Please enter a number, -1 ends input ");
        n = input.nextInt();
    }

    System.out.print("The total is : " + sum);
    System.out.print("The number of items entered is : " + i);

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

Topics list

- Loops
  - 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 a ***property*** has been established, it cannot be ‘unestablished’.
- ‘Once the flag is raised, it cannot be taken down’
- e.g. to check if any of the numbers in an array is odd.
- Once we have found any odd number the ‘**found odd**’ property is established. We never should change that property back

Flag-Based Loops - example

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

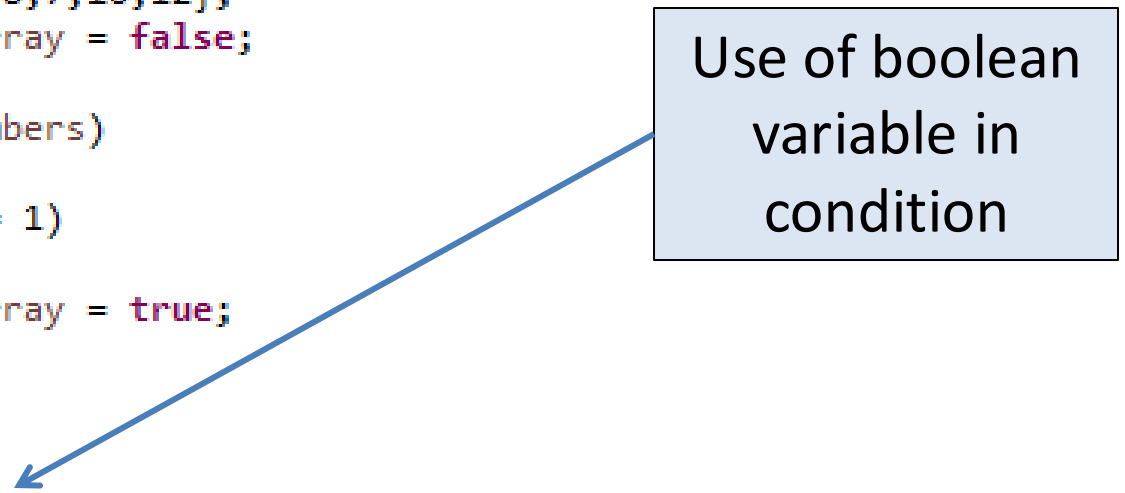
- If you find:
  - At least one odd number, print out to the user that there is at least one odd number.
  - No odd numbers, inform the user of this.

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.");
    }
}
```



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

- Loops
  - 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.

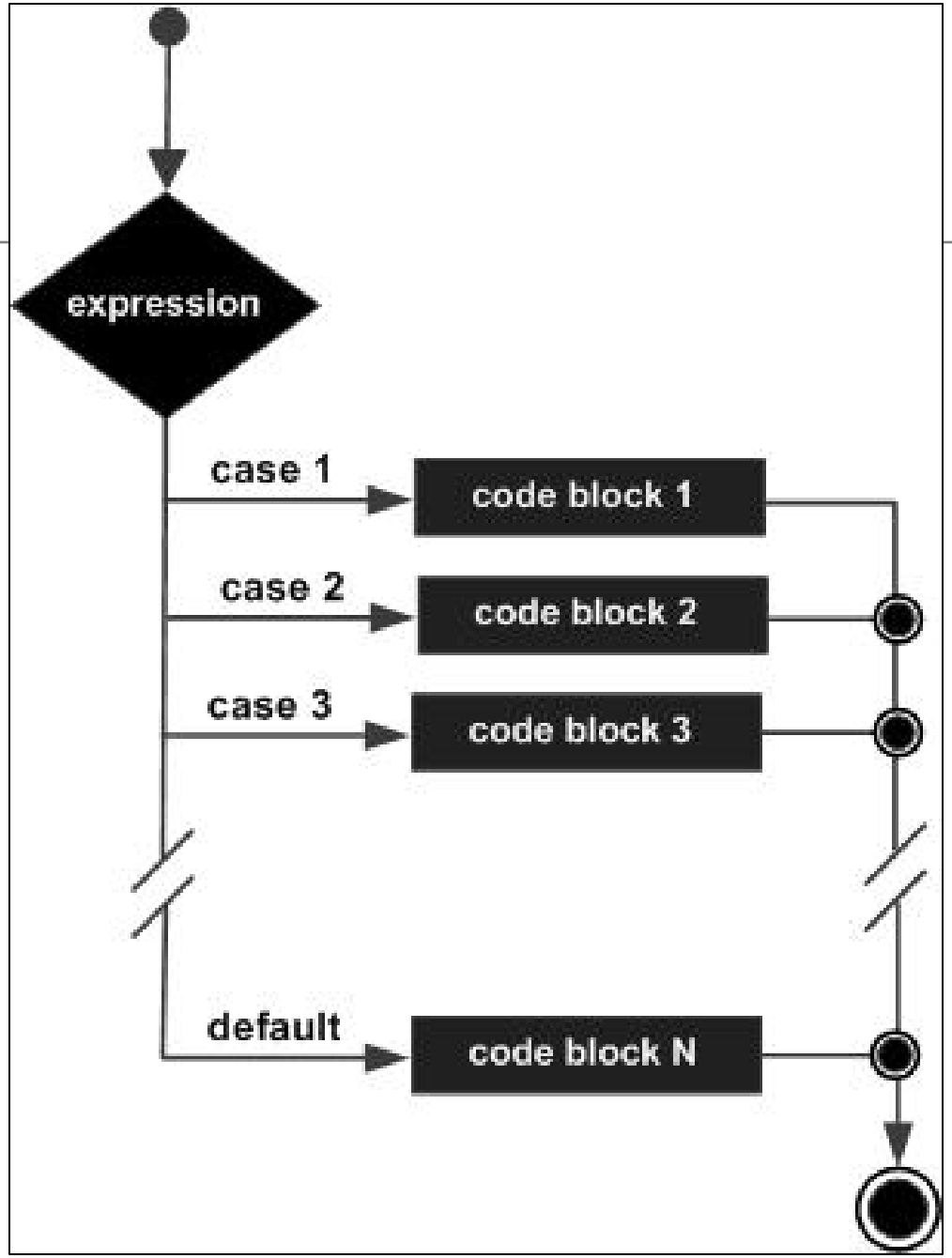
The **switch** statement

- 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**.

- We have used the switch statement before – now we can look at it in more details.

The **switch** statement



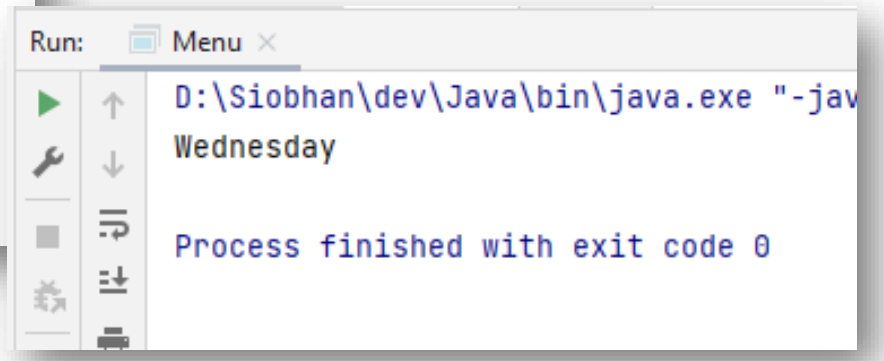
The switch statement

- A *switch* statement can have any number of **case** labels.
- The **default** case is optional; if no default is given, it may happen that no case is executed.
- Can *switch* on **int**, **char** or **String**.

Good Resource: https://blog.codeleak.pl/2020/05/switch-as-an-expression-in-java.html

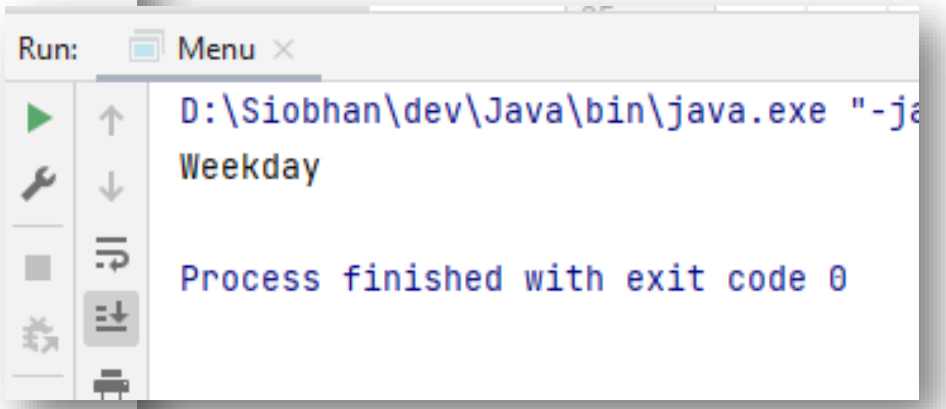
The switch statement – int example

```
public static void main(String args[]){
    int day = 3;

    switch (day){
        case 1 -> System.out.println("Monday");
        case 2 -> System.out.println("Tuesday");
        case 3 -> System.out.println("Wednesday");
        case 4 -> System.out.println("Thursday");
        case 5 -> System.out.println("Friday");
        case 6 -> System.out.println("Saturday");
        case 7 -> System.out.println("Sunday");
        default -> System.out.println("Invalid day: " + day);
    }
}
```



The switch statement – String example

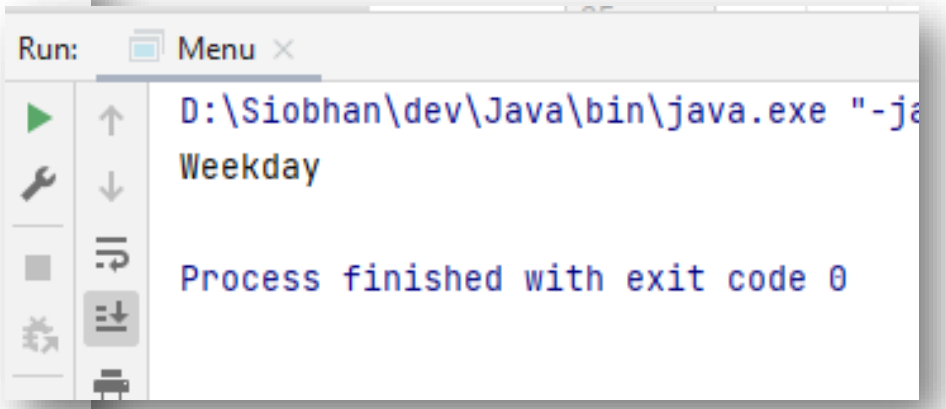
```java
public static void main(String args[]){
    String dayOfWeek = "Mon";

    switch (dayOfWeek.toLowerCase()){
        case "mon"   -> System.out.println("Weekday");
        case "tues"  -> System.out.println("Weekday");
        case "wed"   -> System.out.println("Weekday");
        case "thurs" -> System.out.println("Weekday");
        case "fri"   -> System.out.println("Weekday");
        case "sat"   -> System.out.println("Weekend");
        case "sun"   -> System.out.println("Weekend");
        default -> System.out.println("Invalid day: " + dayOfWeek);
    }
}
```



The switch statement – String example

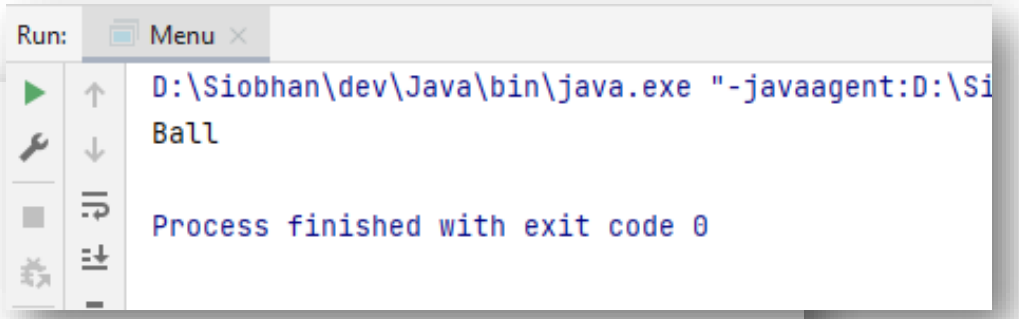
DRY code – Don't Repeat Yourself!

```
public static void main(String args[]){
    String dayOfWeek = "Mon";

    switch (dayOfWeek.toLowerCase()){
        case "mon", "tues", "wed", "thurs", "fri"
                -> System.out.println("Weekday");
        case "sat", "sun"
                -> System.out.println("Weekend");
        default -> System.out.println("Invalid day: " + dayOfWeek);
    }
}
```



The switch statement – char example

```java
public static void main(String args[]){
    char alphabetChar = 'B';

    switch (alphabetChar){
        case 'A' -> System.out.println("Apple");
        case 'B' -> System.out.println("Ball");
        case 'C' -> System.out.println("Cat");
        case 'D' -> System.out.println("Dog");
        case 'E' -> System.out.println("Elephant");
        default  -> System.out.println("Char not on our list yet: " + alphabetChar);
    }
}
```



Topics list

- Loops
  - 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.

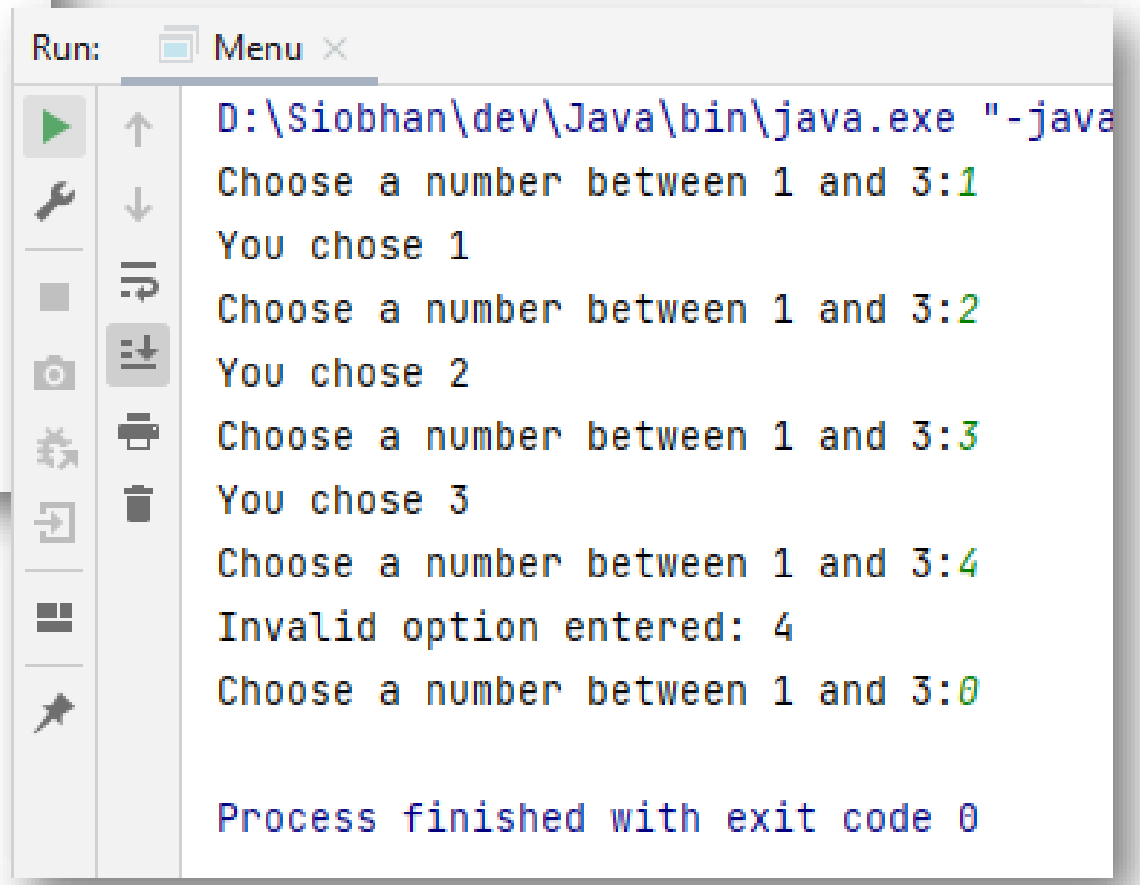
Now loop on the switch statement

Note the use of the Loop Control Variable, **option**.

```
private void runMenu(){

    int option = ScannerInput.readNextInt( prompt: "Choose a number between 1 and 3:");

    while (option != 0){

        switch (option){
            case 1 -> System.out.println("You chose 1");
            case 2 -> System.out.println("You chose 2");
            case 3 -> System.out.println("You chose 3");
            default -> System.out.println("Invalid option entered: " + option);
        }

        option = ScannerInput.readNextInt( prompt: "Choose a number between 1 and 3:");
    }
}
```

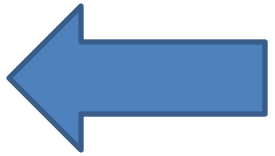
This gives the following output

```java
private void runMenu(){

    int option = ScannerInput.readNextInt( prompt: "Choose a number between 1 and 3:");

    while (option != 0){

        switch (option){
            case 1 -> System.out.println("You chose 1");
            case 2 -> System.out.println("You chose 2");
            case 3 -> System.out.println("You chose 3");
            default -> System.out.println("Invalid option entered: " + option);
        }

        option = ScannerInput.readNextInt( prompt: "Choose a number between 1 and 3:");
    }
}
```



If you wanted more than one line of code associated with a case, just declare a code block:

```
private void runMenu(){

    int option = ScannerInput.readNextInt( prompt: "Choose a number between 1 and 3:");

    while (option != 0){

        switch (option){
            case 1 -> {
                System.out.println("You chose 1");
                System.out.println("You chose One");
            }
            case 2 -> System.out.println("You chose 2");
            case 3 -> System.out.println("You chose 3");
            default -> System.out.println("Invalid option entered: " + option);
        }

        option = ScannerInput.readNextInt( prompt: "Choose a number between 1 and 3:");
    }
}
```



Topics list

- Loops
  - 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…

```
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
==>>
```

Menu to be displayed…

```
private int mainMenu(){
    int option = ScannerInput.readNextInt( prompt: """
            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
            ==>>""");
    return option;
}
```

Driver Class

```
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
==>>
```

Menu to be displayed…

```
private int mainMenu(){
    int option = ScannerInput.readNextInt( prompt: """
            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
            ==>>""");
    return option;
}
```

Driver Class

We are using Raw Strings (the triple double quotes) to format the output for the menu.

This is a newish feature in Java – if it doesn't work, update your JDK.

Handling the menu input…

Driver Class

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

    while (option != 0){

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

        //pause the program so that the user can read what we just printed to the terminal window
        ScannerInput.readNextLine( prompt: "\nPress enter key to continue...");

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

Calling the menu on startup…

Driver Class

```
public class Driver{

    private Store store = new Store();

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

    public Driver() {
        runMenu();
    }
```

```
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
==>>
```

NOTE: A simple menu using the older format of the switch statement

```
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;
    }
}
```

A **break** statement is needed after each case, otherwise execution "falls" into the next statement regardless.

You should use the newer version

Questions?