

Menu Driven Systems

Scanner, while loops, menus and switch

Produced Dr. Siobhán Drohan
by:



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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

Topics list

- Scanner (for console input/output)
- while loops
 - recap on structure – Loop Control Variables
 - Sentinel based loops.
- switch statement.
- A simple menu using switch.
- Shop Project – adding a menu.

Input in Java: the Scanner Class

- The Scanner class comes with Java.
- It allows us to take in data from the console / terminal window.
- It is part of the `java.util` package in the Java Application Programming Interfaces (API).

Input in Java: the Scanner Class

- In order to use the Scanner class, place the following line as the first line of code in your file (i.e. before you declare your class):

```
import java.util.Scanner;
```

- You can now use all the input methods that have been defined in the Scanner class.
- There are methods to take in ints, doubles, chars etc.

Input in Java: the Scanner Class

- Having imported the util package, you will need to write the following instruction in your program.

```
Scanner scan= new Scanner(System.in);
```

- This declares a Scanner **object** called **scan** (you can name this object anything you wish).
- You must have this instruction to be able to call the methods in the Scanner class.

Input Methods of the Scanner Class

```
int age = scan.nextInt();           //reads an integer from the keyboard
double wages = scan.nextDouble();  //reads a double from the keyboard
char taxBand = scan.next().charAt(0); //reads a single char from the keyboard
String firstName = scan.next();     //read a String that has no spaces
String fullName = scan.nextLine()   //reads a String that contains spaces
```

Input Methods of the Scanner Class

```
public class MenuController{

    private Scanner input;
    private Store store;

    public MenuController() {
        input = new Scanner(System.in);

        //read in the details....
        System.out.print("Please enter the store location: ");
        String location = input.nextLine();

        store = new Store(location);
        runMenu();
    }
}
```

Input Methods of the Scanner Class

```
private int mainMenu()  
{  
    System.out.println("\fShop Menu");  
    System.out.println("-----");  
    System.out.println("  1) Add a Product");  
    System.out.println("  2) Remove Product (by index)");  
    System.out.println("  3) List the Products");  
    System.out.println("  4) List the cheapest product");  
    System.out.println("-----");  
    System.out.println("  5) View store details");  
    System.out.println("  0) Exit");  
    System.out.print("==>> ");  
    int option = input.nextInt();  
    return option;  
}
```


Input Methods of the Scanner Class

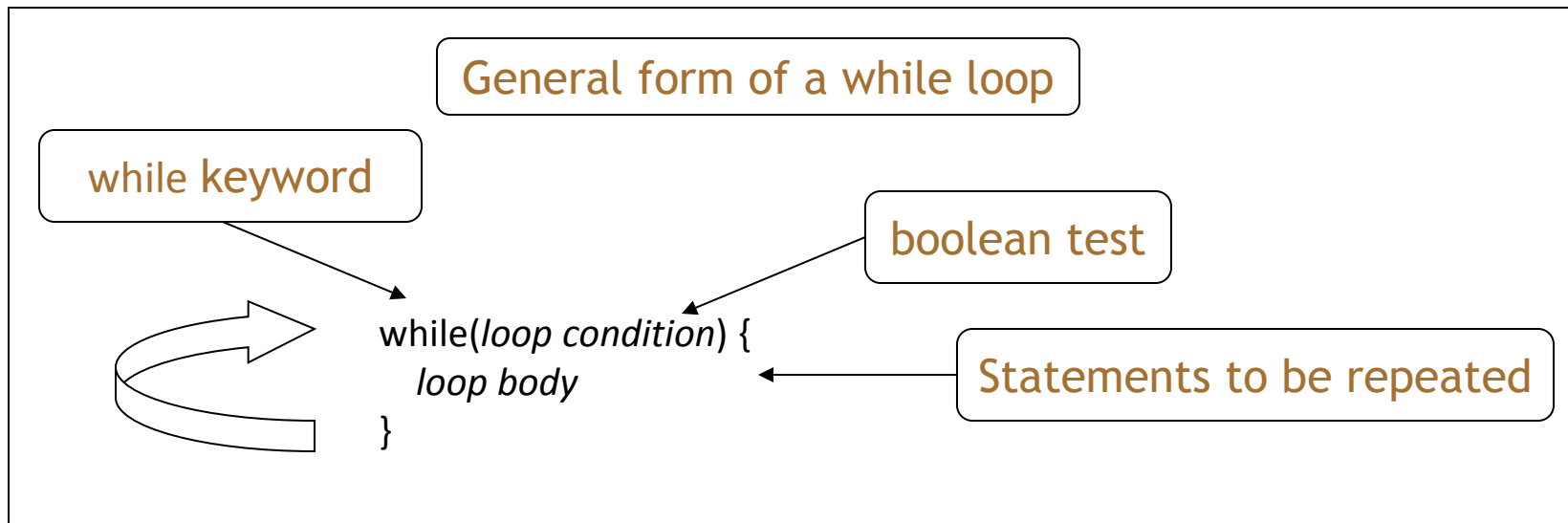
```
//gather the product data from the user and create a new product.
private void addProduct() {
    //dummy read of String to clear the buffer - bug in Scanner class.
    input.nextLine();
    System.out.println("Enter the Product details...");
    System.out.print("\tName: ");
    String productName = input.nextLine();
    System.out.print("\tCode (between 1000 and 9999): ");
    int productCode = input.nextInt();
    System.out.print("\tUnit Cost: ");
    double unitCost = input.nextDouble();
    System.out.print("\tIs this product in your current line (y/n): ");
    char currentProduct = input.next().charAt(0);
    boolean inCurrentProductLine = false;
    if ((currentProduct == 'y') || (currentProduct == 'Y'))
        inCurrentProductLine = true;

    store.add(new Product(productName, productCode, unitCost, inCurrentProductLine));
}
```

Topics list

- Scanner (for console input/output)
- while loops
 - recap on structure – Loop Control Variables
 - Sentinel based loops.
- switch statement.
- A simple menu using switch.
- Shop Project – adding a menu.

while loop: pseudo code



Pseudo-code expression of the actions of
a while loop

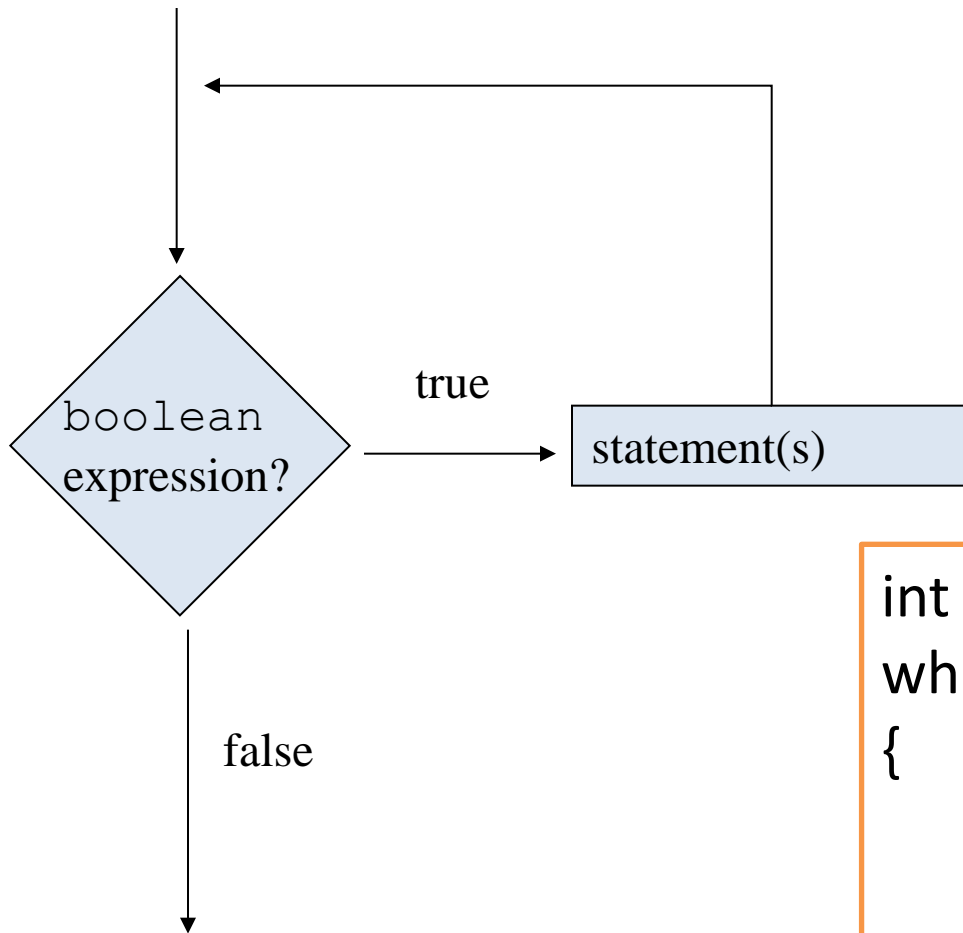
while we wish to continue, do the things in the loop body

while loop: construction

```
Declare and initialise loop control variable (LCV)  
while(condition based on LCV)  
{  
    “do the job to be repeated”  
    “update the LCV”  
}
```

This structure should always be used

while loop: flowchart



```
int i = 1;
while (i <= 10)
{
    System.out.println(i);
    i++;
}
```

while loop: iterating over a collection

```
/**
 * List all notes in the notebook.
 */
public void listNotes()
{
    int i = 0;
    while(i < notes.size()) {
        System.out.println(notes.get(i));
        i++; ←
    }
}
```

Increment *i*
by 1

while the value of *i* is less than the size of the collection, print the next note, and then increment *i*

Sentinel-based loops

- We will signal the end of input with a special value i.e. a sentinel value.

e.g. the code on the next slide continually asks the user to enter a person's age. When the user enters a value of -1, the loops ends and the total of all the ages is printed to the console.

Solution

Initialise

```
public void sentinelBasedWhileLoop()
{
    int sum = 0;

    System.out.print("Please enter an age (-1 ends input): ");
    int ageInput = scanner.nextInt();

    while (ageInput != -1)
    {
        sum += ageInput;
        System.out.print("Please enter an age (-1 ends input): ");
        ageInput = scanner.nextInt();
    }

    System.out.println("The sum of the ages in the room is: " + sum);
}
```

LCV Condition

Update LCV directly
before end of loop

Sentinel-based loops - 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.

Topics list

- Scanner (for console input/output)
- while loops
 - recap on structure – Loop Control Variables
 - Sentinel based loops.
- switch statement.
- A simple menu using switch.
- Shop Project – 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

- Scanner (for console input/output)
- while loops
 - recap on structure – Loop Control Variables
 - Sentinel based loops.
- switch statement.
- A simple menu using switch.
- Shop Project – 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

```
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

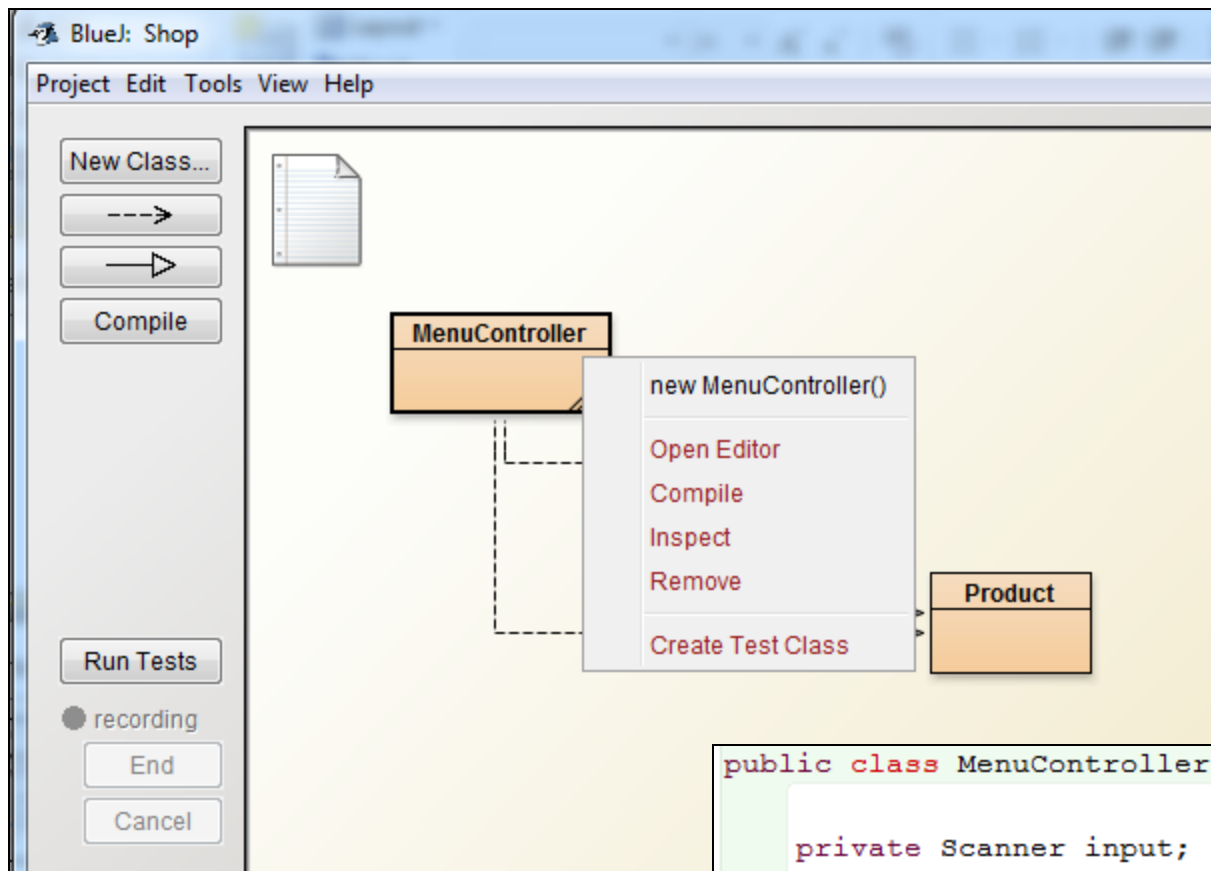
- Scanner (for console input/output)
- while loops
 - recap on structure – Loop Control Variables
 - Sentinel based loops.
- switch statement.
- A simple menu using switch.
- Shop Project – adding a menu.

Adding a basic menu to Shop...

```
private int mainMenu()
{
    System.out.println("\fShop Menu");
    System.out.println("-----");
    System.out.println("  1) Add a Product");
    System.out.println("  2) Remove Product (by index)");
    System.out.println("  3) List the Products");
    System.out.println("  4) List the cheapest product");
    System.out.println("-----");
    System.out.println("  5) View store details");
    System.out.println("  0) Exit");
    System.out.print("==>> ");
    int option = input.nextInt();
    return option;
}
```



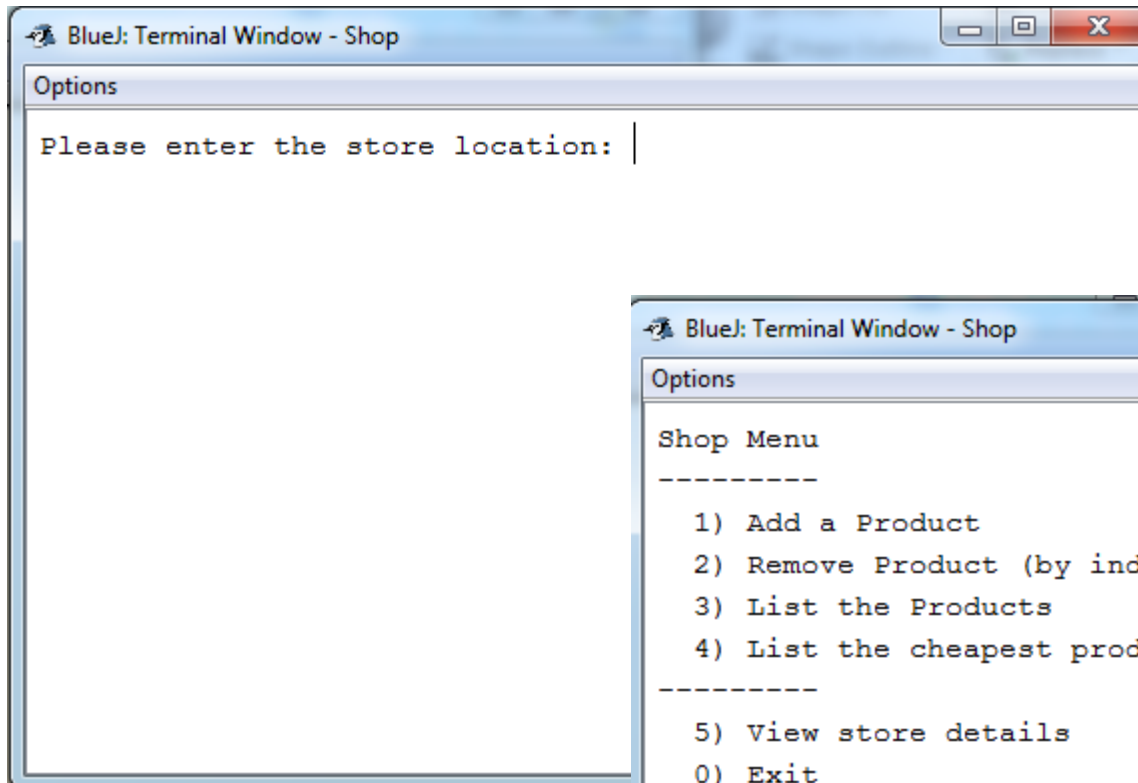
```
private void runMenu() {
    int option = mainMenu();
    while (option != 0) {
        switch (option) {
            case 1:    addProduct();
            break;
            case 2:    System.out.println(store.listProducts());
            if (store.size() > 0) {
                System.out.print("Please enter the index for the product you wish to delete: ");
                int index = input.nextInt();
                store.remove(index);
            }
            break;
            case 3:    System.out.println(store.listProducts());
            break;
            case 4:    System.out.println(store.cheapestProduct());
            break;
            case 5:    System.out.println(store.toString());
            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 terminal window
        System.out.println("\nPress any key to continue...");
        input.nextLine();
        input.nextLine(); //2nd read for bug in Scanner; String read is ignored after reading 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);
}
```



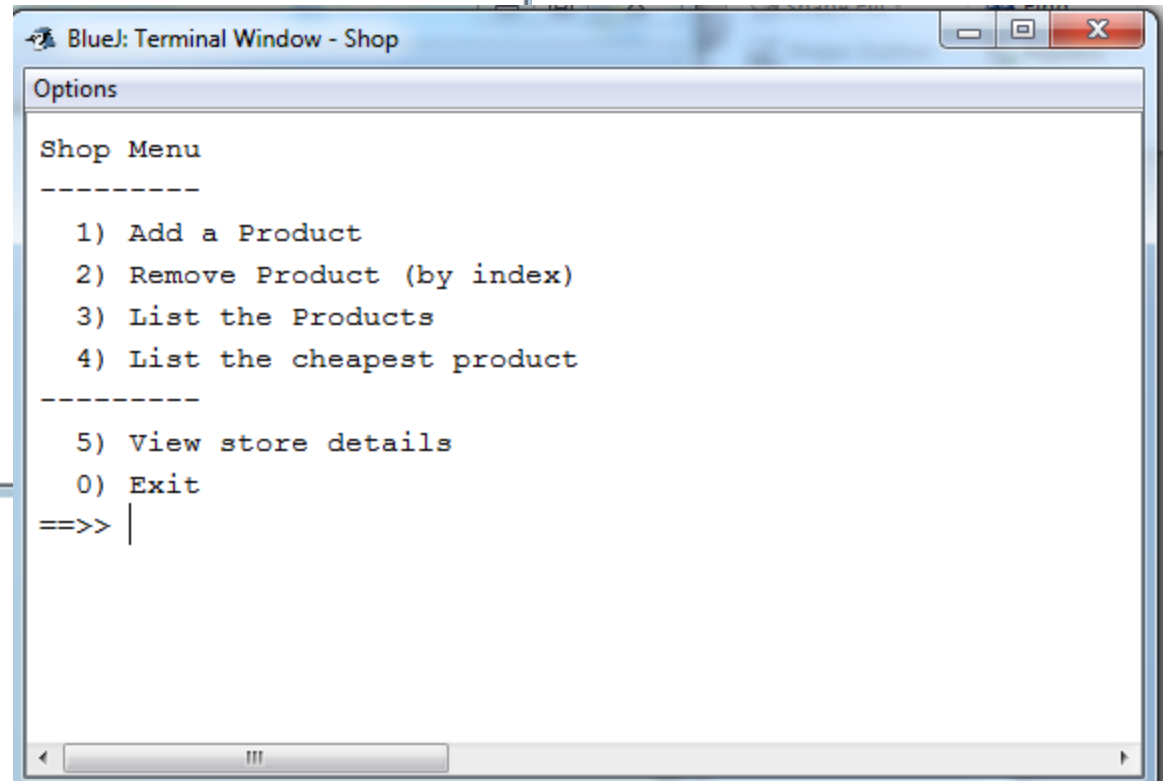
Calling the menu on startup...

```
public class MenuController{  
  
    private Scanner input;  
    private Store store;  
  
    public MenuController(){  
        input = new Scanner(System.in);  
  
        //read in the details....  
        System.out.print("Please enter the store location: ");  
        String location = input.nextLine();  
  
        store = new Store(location);  
        runMenu();  
    }  
}
```

The displayed menu...



```
BlueJ: Terminal Window - Shop
Options
Please enter the store location: |
```



```
BlueJ: Terminal Window - Shop
Options
Shop Menu
-----
  1) Add a Product
  2) Remove Product (by index)
  3) List the Products
  4) List the cheapest product
-----
  5) View store details
  0) Exit
==>> |
```

Questions?





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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