Menu Driven Apps

Loops, menus and the switch statement

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

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

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

Topics list

- Loops
  - Loop Control Variables
  - Arrays and counter controlled loops
  - Arrays and sentinel based loops
  - Arrays and flag-based loops

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 static void main(String[] args) {
    int i = 0;
    while (i <10) {
        System.out.println(i + ": Hello");
        i++;
    }
}
```

```
Driver.main({ });
0: Hello
1: Hello
2: Hello
3: Hello
4: Hello
5: Hello
6: Hello
7: Hello
8: Hello
9: Hello
```

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

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int[] numbers = new int[10];
    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.print("The sum of all the entered numbers is : " + sum);

}
```

```
Driver.main({ });
Please enter a number: 10
Please enter a number: 10
Please enter a number: 5
Please enter a number: 5
Please enter a number: 20
The sum of all the entered numbers is : 50
```

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

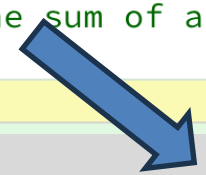
Here, we know at run-time how many inputs we have.

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int[] numbers;
    int numNumbers = 0;
    int sum = 0;
    System.out.print("Please enter the number of numbers you want 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.print("The sum of all the entered numbers is : " + sum);

}
```



```
Driver.main({ });
Please enter the number of numbers you want to enter: 4
Please enter a number: 10
Please enter a number: 5
Please enter a number: 5
Please enter a number: 10
The sum of all the entered numbers is : 30
```

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

```
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 4
Please enter the number, -1 ends input: 3
Please enter the number, -1 ends input: 2
Please enter the number, -1 ends input: -1
The sum of all the entered numbers is : 14
```

```java
public static void main(String[] args) { //sentinel while loop
    Scanner input = new Scanner(System.in);

    int sum = 0;
    System.out.print("Please enter the number, -1 ends input: ");
    int n = input.nextInt();


    while (n != -1) {

        sum+= n;
        System.out.print("Please enter the number, -1 ends input: ");
        n = input.nextInt();
    }

    System.out.print("The sum of all the entered numbers 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

```java
public static void main(String[] args) { //sentinel while loop
    Scanner input = new Scanner(System.in);

    int sum = 0;
    int counter= 0;
    System.out.print("Please enter the number, -1 ends input: ");
    int n = input.nextInt();


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

    System.out.print("The sum of all the entered numbers is : " + sum);
    System.out.print("The number of entered numbers is : " + counter);

}
```

```
Driver.main({ });
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: -1
The sum of all the entered numbers is : 20
The number of entered numbers is : 4
```

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

```
public static void main(String[] args) { //storing input
Scanner input = new Scanner(System.in);

int sum = 0;
int counter= 0;
int numbers[] = new int[100];
System.out.print("Please enter the number, -1 ends input: ");
int n = input.nextInt();


while (n != -1 && counter < 100) {
     numbers[counter] = n;
     sum+= n;
     counter++;
     System.out.print("Please enter the number, -1 ends input: ");
     n = input.nextInt();
}

System.out.println("The sum of all the entered numbers is : " + sum);
System.out.println("The number of entered numbers is : " + counter);

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

```
Driver.main({ });
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: 5
Please enter the number, -1 ends input: -1
The sum of all the entered numbers is : 20
The number of entered numbers is : 4
0     Number entered: 5
1     Number entered: 5
2     Number entered: 5
3     Number entered: 5
```

Sentinel based loops and menus

- Menus
  - Based on repetitively giving user options
  - Stops when user enters a know ‘exit’ value
  - We will use the sentinel-based loops (with ‘exit’ value as our sentinel) when we implement a menu system

Topics list

- Loops
  - Loop Control Variables
  - Arrays and counter controlled loops
  - Arrays and sentinel based loops
  - Arrays and flag-based loops

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 i = 0; i< numbers.length; i++)
    if (numbers[i] %2 == 1) {   //check if it is odd
       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 numbers 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 flagBasedLoopWithReturn(){
    int numbers[] = {4,6,8,7,10,12};
    boolean oddNumberInArray = false;

    for(int i = 0; i< numbers.length; i++)
    if (numbers[i] %2 == 1) {    //check if it is odd
        oddNumberInArray = true;
    }

    return oddNumberInArray;

}
```

Calling the method and handling the returned boolean

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

Questions?