Laboratory 4 – Week 5

Loops in Java

### 4.1 Introduction

The laboratory session covers loops in Java. Again we will be implementing some of the examples given during the lecture, followed by some additional examples. We will then look at a few sections of the help pages.

**Note that: this worksheet *is* one of the worksheets from which your laboratory worksheets portfolio of work will be assessed [CodeRunner tests].**

### 4.2 Preliminaries

Create a new project and Java class called CS1702\_Lab4.

### 4.3 Conditions and Loops

In worksheet three, section 3.4, we wrote three programs that decided if a single number had some sort of property, for example one of section 3.4 was to determine if a number (>0) was divisible by two and three. Copy these programs and amend each of them so that you use a for loop to test all of the number between 1 and 100 in a single run of each program.

Does this now change your answer to the final question of section 3.4?

### 4.4 Generating Sequences Using For Loops

Implement the following for loop and consider what it does.

**int** i;

**for**(i=0;i<10;++i)

{

System.*out*.println(i);

}

Modify the loop to produce the following sequences:

1. 4,5,6,7,8,9,10,11
2. 10, 13, 16, 19
3. 1, 4, 7, 10, 13, 16
4. 2, 4, 6, 8, 10, 12
5. 1, 4, 9, 16, 25, 36, 49, 64, 81, 100
6. -10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10
7. -20, -15, -10, -5, 5, 10, 15, 20

**Note**: you *may* have to add some conditional statements as part of the for loop code block.

### 4.5 Generating Patterns Using For Loops

Implement the following for loop and consider what it does.

**int** i;

**for**(i=0;i<10;++i)

{

System.*out*.print("\*");

}

Modify the loop to produce the following sequences:

1. "++++++++++"
2. "---------"
3. "+-+-+-+-+-"
4. "\*+-\*+-\*+-"
5. "\*+-\*+-\*+-\*"
6. "\*\*++\*\*++\*\*++"
7. "\*\*\*+++---\*\*\*+++---"
8. "\*\*\*+++------+++\*\*\*"

**Note**: again you *may* have to add some conditional statements as part of the for loop code block.

### 4.6 While Loops

Implement two of the solutions to section 4.4 using a while loop. This can initially be done by implementing one of the while loop examples from the lecture notes.

Next, implement Euclid's algorithm (*Highest Common Factor* - HCF) by using a while loop.

Test your program on the following:

1. HCF(88,26)=2
2. HCF(54,87)=3
3. HCF(16,84)=4
4. HCF(55,25)=5
5. HCF(42,72)=6
6. HCF(77,28)=7
7. HCF(80,88)=8

Verify that HCF(*a*,*b*) = HCF(*b*,*a*).

This may seem daunting at first; however it is straightforward if you consider two of the flow charts in the lecture notes. Look at the flow chart for Euclid's algorithm and also the flow chart for the while loop. By comparing them, you should be able to construct a solution to this problem.

### 4.7 Nested Loops

Implement the following for loops and consider what they do.

**int** i,j;

**for**(i=1;i<5;++i)

{

**for**(j=1;j<i+1;++j)

{

System.*out*.print(i);

}

}

Modify the loops to produce the following sequences:

1. "122333444455555"
2. "22333444455555666666"
3. "133355555"
4. "1335557777"
5. "555554444333221"
6. "544333222211111"
7. "+\*\*+++\*\*\*\*+++++"
8. "--\*\*\*++++-----\*\*\*\*\*\*+++++++"

Write nested loops that perform the following tasks:

1. Display all of the possible pairs of numbers between 1 and 10 (Hint: there are 100 pairs)
2. Display all possible pairs in the form *x*, *y* where *x* < *y* and 0 < *x*, *y* < 11 (Hint: there are 45 pairs)
3. Display all possible pairs of the numbers 1,2,3,4 paired with 4,5,6,7,8 (Hint: there are 20 pairs)

### 4.8 Using Loops With Your Finch

This part of the worksheet involves making the Finch move, thus you should test your program outside of the laboratory or at home.

At the very top of your program, before the class definition, add the import statement for the Finch control code as in the previous worksheet.

Then add the following lines of code to the bottom of your program:



Within this program there are a few things to note:

public static void main(String args[]) throws Exception: Because we are using the command Thread.Sleep(500) we have to add this addition to the declaration of the main method. We will be dealing with Exception handling next term.

System.currentTimeMillis(): This function/command returns the current time in milliseconds (ms); note that 1000 ms = 1 second.

while(System.currentTimeMillis() - before < 5000) : This part of the program repeats the loop until 5 seconds has past. By previously storing the time in the variable before we can check to see if 5000 milliseconds have passed by subtracting the current time from this recorded value.

Thread.sleep(500): This causes the program to pause for half a second/500 ms.

if (myf.isTapped()) break: This command returns true if the Finch is tapped, otherwise it returns false.

myf.stopWheels(): This stops both of the Finch’s wheels.

This program moves the Finch forward for 5 seconds, whilst checking whether the Finch has been tapped or not. If the Finch is tapped, the Finch stops and the Finch halt. Can you replace the line:

myf.setWheelVelocities(100,100)

for

myf.setWheelVelocities(100,100,5000)?

What happens? Why do you think this is?

Modify the program so that the Finch moves in a straight line until it reaches an obstacle and then stops. You might find the following Finch commands useful:

isObstacleLeftSide, isObstacleRightSide, isObstacle