Laboratory 3 – Week 4

Conditional Statements in Java

### 3.1 Introduction

The laboratory session covers conditional statements 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].**

### 3.2 Relational operators

Create a new project and Java class called CS1702\_Lab3. Add the following code:

**static** **public** **void** main(String args[])

{

**int** a = 1000,b = -22;

**if** (a < b)

{

System.*out*.println("a is less than b");

}

**else**

{

System.*out*.println("a is NOT less than b");

}

}

Run the program. It should display the message “*a is NOT less than b*”. Test the program on a number of different values of a and b such that both parts of the if statement is run. Now add a similar set of program code that tests if a is greater than b.

Using the above code “snippet” as a basis, write a set of Java if statements that determines if the following statements are true, note that you must declare the variables first and choose the correct type.

Let x = 100, y = 204, z = -23.1, a = true, b = false, c = -204

1. x < y
2. x > z and a = b
3. 2c > y
4. x = b
5. c ≠ y or c = y
6. z ≠ y and c = a
7. y ≥ y and a+3 ≠ 2

### 3.3 Designing if Statements

Write the Java code that will solve the following problems:

1. Given three numbers, displays them in the correct order (ascending)
2. Given three names (e.g. name1, name2 and name3 as string variables), display them in the correct alphabetical order

For example, for the first part of this exercise, if:

a = 10, b = -10 and c = 3

Then the program might output:

The correct order is ‘b’<=’c’<=’a’

Note that your code should work on any three numbers or names, not just one specific example.

The following suggestions might help in writing these small programs:

1. You could assume that the three numbers will be stored as three variables, named for example, a, b and c.
2. When writing these programs, you should try and design your program first, i.e. it might be useful to try and write the flow chart for the program.
3. Test any design (or program) you write with some test data.
4. The second program (of section 3.3) might be solvable by using (copying and modifying) the code from the first one (of section 3.3).

How would you solve this problem yourself? Could you base your design on the way that you solve it? Once you have got your program running, have a think about the following:

1. How easy would it be to change your program so that it reverses the order?
2. What modifications would be needed to make your program work for 10 numbers? What about 1000? What are the implications of this?

### 3.4 Compound Conditions

Write the following programs:

1. Displays if a whole number (>0) is divisible by 2 and 3
2. Displays if a whole number (>0) is divisible by 7 or 9
3. Displays if a whole number (>0) is divisible by 2 and 3 but not 5

Assume that the number is stored as a variable. What mathematical operator could you use to solve these exercises? If an integer *x* is divisible by an integer *y*, what does that mean? Try your programs on a number of examples.

If I wanted you to rerun your programs on the numbers between 1 and 1000, what issues would you encounter?

### 3.5 Using switch Statements

Consider the following code “snippet” that uses a switch statement:

String month = "September";

**int** daysinmonth = 0;

**switch**(month)

{

**case** "January":

daysinmonth = 31;

**break**;

**case** "February":

daysinmonth = 28;

**break**;

**case** "March":

daysinmonth = 31;

**break**;

**case** "April":

daysinmonth = 30;

**break**;

**case** "May":

daysinmonth = 31;

**break**;

**case** "June":

daysinmonth = 30;

**break**;

**case** "July":

daysinmonth = 31;

**break**;

**case** "August":

daysinmonth = 31;

**break**;

**case** "September":

daysinmonth = 30;

**break**;

**case** "October":

daysinmonth = 31;

**break**;

**case** "November":

daysinmonth = 30;

**break**;

**case** "December":

daysinmonth = 31;

**break**;

**default**:

daysinmonth = -1;

**break**;

}

System.***out***.println("Days in month " + month + " = " + daysinmonth);

}

This program should display the days in a month for a given variable month. Copy and run the code. Change the month variable to a number of different valid and invalid values.

The code is slightly cumbersome and bloated, with lots of repetition. Also note that I have not included the braces (brackets “{}”), this was to save space. Make sure you put them in!

Consider the following modified version:

**switch** (month)

{

**case** "January": **case** "March": **case** "May": **case** "July": **case** "August": **case** "October": **case** "December":

daysinmonth = 31;

**break**;

**case** "February":

daysinmonth = 28;

**break**;

**case** "April": **case** "June": **case** "September": **case** "November":

daysinmonth = 30;

**break**;

**default**:

daysinmonth = -1;

**break**;

}

System.***out***.println("Days in month " + month + " = " + daysinmonth);

Check that the program runs in the same manner as the previous version. Notice how we can group similar cases together on the same line, thus reducing space and aiding readability.

Write a similar pair of programs to the two above that use a switch statement to determine the number of legs a creature has (stored in a String in a similar manner to the month variable above). Remember to print the name of the creature as in the first switch statement.

The animals the program should cater for are as follows:

### Alsatian Dog, Sphynx Cat, Cobra, Baboon, Centipede, Ocelot, Cod, Human, Tarantula, Black Scorpion, Paul Allen the German Octopus, Minke Whale, Klingon and Potto

Remember to have a default option in your program.

### 3.6 The ?: Notation

Finally rewrite all of your answers to section 3.2 using the ?: notation. Which style do you think is better and why?

### 3.7 Using Conditional Statements with your Finch

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 a new method in your program.

Finch myfinch = **new** Finch();

**if** (myfinch.isFinchUpsideDown() == **true**)

{

System.*out*.println("I am on my back!");

}

**else**

{

System.*out*.println("I am NOT on my back!");

}

myfinch.quit();

Position your Finch so that it is level and the correct way up. Run the program and observe the results. Put your Finch on its back and run the program. You should hopefully get a different result.

This small program is responding to a sensor within the Finch and uses the call to the predefined method myfinch.isFinchUpsideDown() to determine if the Finch is on its back or not.

The *JavaDocs* help pages for the Finch control code can be found at:

<http://www.finchrobot.com/javadoc/index.html>

Follow this link and click in the left hand pane and select *Finch*. Find the documentation for the isFinchUpsideDown method and read what it does.

Read up on the following methods and then modify and test the program above to use each of these methods.

myfinch.isBeakDown();

myfinch.isBeakUp();

myfinch.isFinchLevel();

myfinch.isLeftWingDown();

myfinch.isRightWingDown();

myfinch.isObstacleLeftSide();

myfinch.isObstacleRightSide();

myfinch.isShaken();

myfinch.isTapped();

Are there any problems in conducting the tests for the last two methods?