# Assessed Lab 2

# Conditional Statements in Java

## ASSESSMENT INFORMATION

**This worksheet *is* one of the six assessed lab sheets.**

**Let me know *in advance* when you’d like to be assessed.**

**Do not forget to have it ‘signed off’ after you have been assessed.**

## 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.

## 2 Relational operators

Create a new project and Java class called CS1002\_Lab2. 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");

}

}

### EXERCISE 1

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.

### EXERCISE 2

Using the above code “snippet” as a basis, write a set of Java if statements that check if the following statements are true. The code should print either “True” or “False”.

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 Designing if Statements

### EXERCISE 3

Write the Java code that will solve the following problem:

1. Given three numbers, displays them in the correct order (ascending)

For example, 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, 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 Compound Conditions

### EXERCISE 4

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?

## 5 Using switch Statements

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

String month = "";

**int** daysinmonth = 0;

**int** month\_num = 1;

**switch** (month\_num)

{

**case** 1:

month = "January";

daysinmonth = 31;

**break**;

**case** 2:

month = "February";

daysinmonth = 29; //Given that it is 2012...

**break**;

**case** 3:

month = "March";

daysinmonth = 31;

**break**;

**case** 4:

month = "April";

daysinmonth = 30;

**break**;

**case** 5:

month = "May";

daysinmonth = 31;

**break**;

**case** 6:

month = "June";

daysinmonth = 30;

**break**;

**case** 7:

month = "July";

daysinmonth = 31;

**break**;

**case** 8:

month = "August";

daysinmonth = 31;

**break**;

**case** 9:

month = "September";

daysinmonth = 30;

**break**;

**case** 10:

month = "October";

daysinmonth = 31;

**break**;

**case** 11:

month = "November";

daysinmonth = 30;

**break**;

**case** 12:

month = "December";

daysinmonth = 31;

**break**;

**default**:

daysinmonth = -1;

month = "(unknown)";

**break**;

}

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

}

### EXERCISE 5

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

The code is slightly cumbersome and bloated, with lots of repetition. Consider the following modified version:

**int** month\_num = 1;

**int** daysinmonth = 0;

**switch** (month\_num)

{

**case** 1: **case** 3: **case** 5: **case** 7: **case** 8: **case** 10: **case** 12:

daysinmonth = 31;

**break**;

**case** 2:

daysinmonth = 29; //Given that it is 2012...

**break**;

**case** 4: **case** 6: **case** 9: **case** 11:

daysinmonth = 30;

**break**;

**default**:

daysinmonth = -1;

**break**;

}

System.*out*.println("Days in month #" + month\_num + " = " + 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.

### EXERCISE 6

Write a similar **pair** of programs to the two above that use a switch statement to determine the number of legs a creature has.

1. In the first program, the case number is each animal. So for example, case **1** could be “llama”, case **2** is the “goat” etc.

The print message should look like:

"a(n)" + animal + " has " + leg\_num + " legs"

For instance,

a(n)llama has 4 legs

1. In the second program, group the animals together so that the case number is the leg number. So case 4 should be the Alsatian dog, cat and ocelot, case 2 Human, Klingon, Baboon and Potto etc.

This time the print message should look like:

"a(n)" + animals + " have " + leg\_num + " legs"

For instance,

a(n)llama, goat and ocelot have 4 legs

The animals the program should cater for are as follows:

### 1:llama, 2:goat, 3:Cobra, 4:Baboon, 5:Centipede, 6:Ocelot, 7:Cod, 8:Human, 9:Tarantula, 10:Black Scorpion, 11:Paul Allen the German Octopus, 12:Minke Whale, 13:Klingon and 14:Lela

Remember to have a default option in your program.

## 6 The ?: Notation

### EXERCISE 7

Finally rewrite one of your answers to exercise 2 in section 2 using the ?: operator. Which style do you think is better and why?

### EXERCISE 8

Use Scanner class to ask the user to enter a value, and then print out if the entered value is even or odd