**Lab\_B: Java Arithmetic Expressions**

The purpose of this lab is to reintroduce you to the use of arithmetic expressions but in Java. It covers material not specifically looked at in any depth in the lectorals.

**Preliminaries**

In the lectorals we have introduced the notion of primitive types. These can be grouped as:

* Numeric types
  + Integer types
  + Real types
* Character types
* Boolean types

In this lab we are only interested in the numeric types.

In the lectoral you have seen that there are several types of integer variables. Each type defines a range of legal values. For the majority of this module we will only make use of the type **int**. In this lab we will use int. Similarly although there are two real types we will only make use of type **double.**

**Java supports the following arithmetic operators:**

|  |  |
| --- | --- |
| **+** | **addition** |
| **-** | **subtraction** |
| **\*** | **multiplication** |
| **/** | **division** |
| **%** | **remainder** |

In the course of your general education you will have met most of these. The one you may not have

come across is %. This calculates the remainder after an integer division. The following examples

illustrate how % works:

10 % 5 gives 0

10 % 3 gives 1

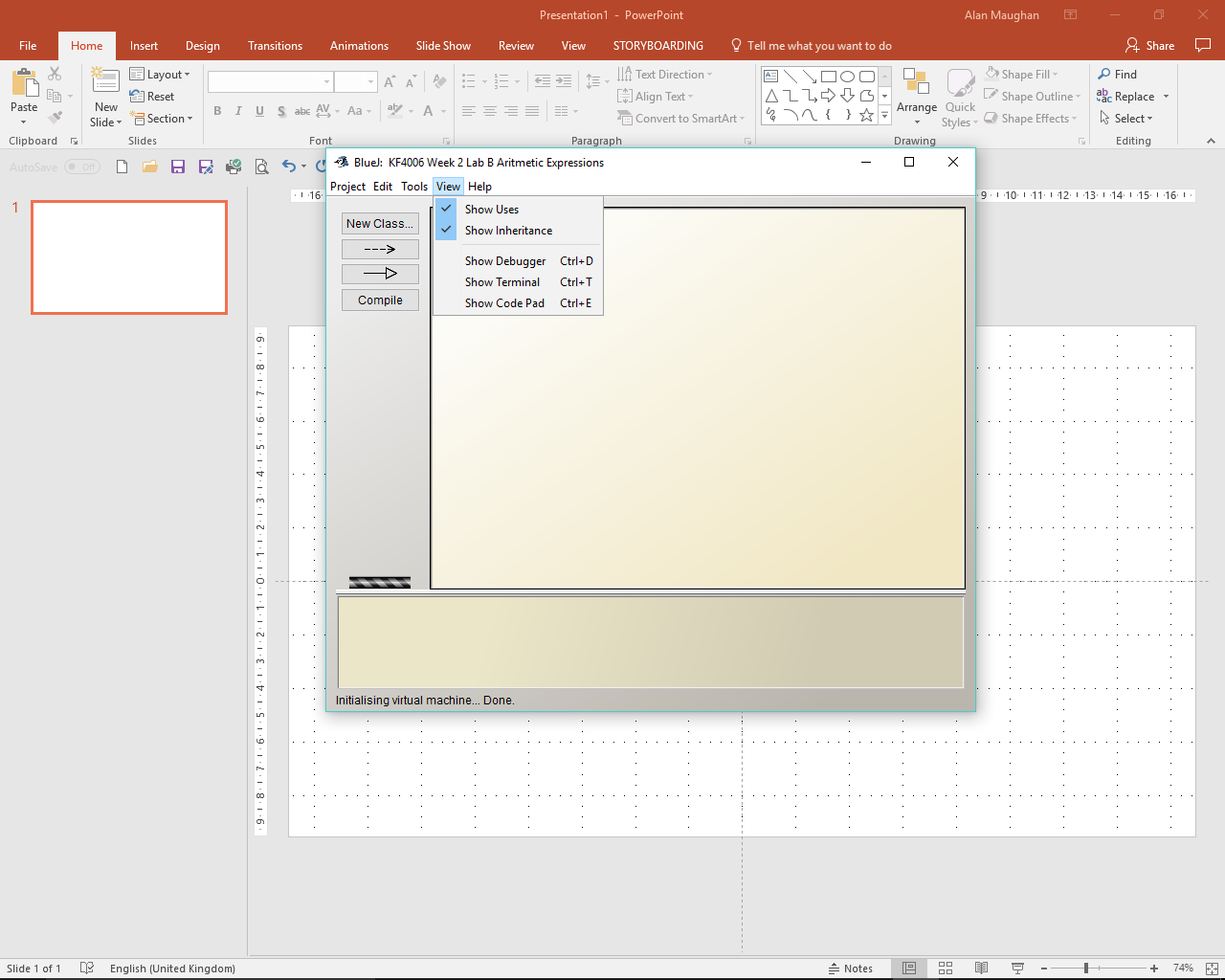
10 % 6 gives 4

10 % 11 gives 10

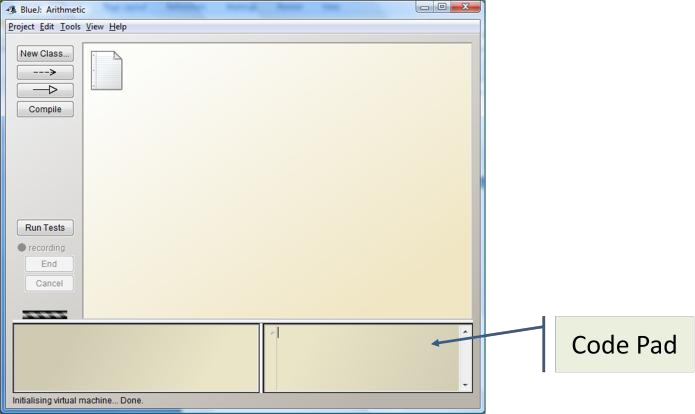
When using programming languages we need to understand how the arithmetic operations work.

**Task 1**

Create a new project, call it Arithmetic. From the menu bar select view and then Show Code Pad:



***Figure 1***

Your Blue J window will now appear as:

***Figure 2***

The code pad can be used to investigate details of Java semantics and to illustrate and experiment

with Java syntax. You can find put more about using the Code Pad in chapter 6 of the BlueJ Tutorial (Help -> BlueJ Tutorial).

**Task 2**

In order to try out the Code Pad type 4 / 2 in the Code Pad and press return. This should result in 2

(int) appearing. Now try 3 / 2. The answer is 1. For integer division Java only gives the integer part of

the result.

**Task 3**

You will now investigate integer arithmetic in Java. Evaluate each of the expressions in Table 1,

writing down your answer in the appropriate column. Then evaluate each using the Code Pad. Check

to ensure your original answer was correct.

|  |  |  |
| --- | --- | --- |
| Expression | Your Answer | BlueJ Answer |
| 5/2 | 2 | 2 |
| 3 \* 5 + 2 | 17 | 17 |
| 9 / 3 + 1 | 4 | 4 |
| 10 / 2 \* 3 | 15 | 15 |
| 3 \* 10 / 10 | 3 | 3 |
| 5 + 10 / 10 | 6 | 6 |
| 4 \* 10 % 8 | 0 | 0 |
| 3 + 10 - 5 | 8 | 8 |
| 2 - 9 + 5 | -2 | -2 |

***Table 1***

**Task 4**

Java has an order of precedence for evaluation arithmetic expressions: “left to right”, multiplication (\*), division ( /) and remainder ( %) before addition (+) and subtraction ( \_) . This means that we need to think carefully about what we actually want when we write out an arithmetic expression. The use of parenthesis, i.e. brackets can help remove ambiguities in

arithmetic expressions.

Evaluate each of the following expressions. Note when the use of brackets makes a difference. Make

sure you understand why.

|  |  |  |
| --- | --- | --- |
| Expression | BlueJ Answer | Answer Different to Table 1? |
| 3 \* (5 + 2) | 21 | y |
| 9 /( 3 + 1) | 2 | y |
| 10 / (2 \* 3) | 1 | y |
| (3 \* 10) / 10 | 3 | n |
| (5 + 10) / 10 | 1 | y |
| 4 \* (10 % 8) | 8 | y |
| 3 + (10 - 5) | 8 | n |
| 2 - (9 + 5) | -12 | y |

***Table 2***

**Task 5**

Real arithmetic is more intuitive. Java will produce the “correct” answer. (Note: arithmetic on any

computer is only approximate and answers are sometime slightly out.)

Try the following in Code Pad:

3.0 / 2.0

1.0 / 3.0

**Task 6**

Java permits mixed arithmetic, that is a mixture of int and double. Try entering 3 / 2.0 in the Code

Pad.

We need to be careful using mixed arithmetic because it is very easy to make mistakes. Try entering

3 / 2 \* 4.0. You might have been hoping for the answer to be 6.0 but it is 4.0. Why?

**Task 7**

Type the following into Code Pad, take care with the **;**

int i = 10;

int j = 20;

double d = 0.5;

Evaluate the following expressions. In each case predict what the result would be.

|  |  |  |
| --- | --- | --- |
| Expression | Your Answer | BlueJ Answer |
| i + 1 \* 2 | 12 | 12 |
| 2 \* (i / j) | 1 | 0 |
| i % 3 | 1 | 1 |
| j / 2 \* i | 100 | 100 |
| d \* (i / j) | 0.25 | 0.0 |
| i +1 % 3 | 10.33r | 11 |
| 2 \* d | 1 | 1.0 |
| (d \* i) / j | 0.25 | 0.25 |
| “” + i + 2 | (I = 10) + 2 = 12, therefore “” = 90. 90 + 10 + 2 = 102 | 102 |

**Task 8**

Sometimes we may need to convert an int value to a double. In order to achieve this we must *cast*.

The syntax for this is

(type) expression

So if we want a double result when dividing i by j in the previous task we would write

(double)i/j

Try this in the Code Pad. = 0.5

Then try (double)(i/j). What is the answer? Why?

0.0

Experiment with casting to ensure you understand it.