# CSCU9A2 Spring 2017
# Programming Practical 1A: 23/24 January
# Recap Java and using BlueJ

**Remember to register your practical attendance at the start of each session:**

- **Double click on the My Computer icon, then on Groups on Wide (the V: drive), then on CSCU9A2, and double click on the Register icon (it should be a camera).**
- **You will be presented with a screen inviting you to register for the practical. Click to register.**
- **You will receive some response indicating a successful registration or failure.**

**Please report any registration problems to the tutors. Note that the program will not accept registrations in other labs, nor outside the scheduled practical hours. The registration application is also used for recording your checkpoints, so you might like to leave it open until later. Alternatively, you could click on the "Exit" button to quit the registration program.**

**You can also use it to view your attendance and checkpoint records at any time by selecting View from the menu bar.**

**If you get stuck or need help at any time during the practical, *ask a demonstrator.***

**Remember that these are teaching sessions, and during the practicals you should not be surfing the Web, nor texting nor chatting online with your friends.**

***Most worksheets contain a checkpoint. When you reach a checkpoint you must show your work to a demonstrator in order to get credit for your work. The demonstrator will check what you have done, may ask some questions, and will give feedback. The checkpoints contribute 20% to your final module grade.***

**There will normally be some additional, more challenging exercises after the checkpoint that you should attempt to get maximum benefit from the practical work.**

**You should carry out practical work during the scheduled practical sessions, *and continue in your own time if you need to*. However, you must attend a scheduled practical class to have your checkpoints recorded. The demonstrators will not record checkpoints outside of scheduled practical times.**

***Each checkpoint must be completed by the end of the second week following the worksheet (or third week for the checkpoint(s) immediately preceding the mid-semester break).***

## Further Documentation

You will normally find it useful to refer to your notes from the CSCU9A1 and CSCU9A2 programming lectures, and also to Java for Everyone. The worksheets will occasionally point you to extra on-line information – consulting the documentation should become a habit!

## This worksheet:

This worksheet leads you through some Java exercises based on the worksheets from CSCU9A1, to give you the opportunity to recall your Java and how to use BlueJ. There is a helpful "Introduction to BlueJ" reminder document on the module practicals web page, and in the CSCU9A2 Groups on Wide folder.

## InputLoop

- Open **My Computer**, and then open the **CSCU9A2 Groups on Wide** folder, and then the **Java** folder. This is where Java programs and fragments will be placed *for you to **copy** for practical work* – you *cannot* use them where they are. Open **InputLoop.txt** with Notepad.
- We recommend that you keep your folders and files organized very carefully. Suggestion: Create a new folder called **CSCU9A2** in your home folder, and keep each individual BlueJ project in a separate folder inside **CSCU9A2**.
- Launch BlueJ and create a new BlueJ project called **InputLoop** ***in its own new folder in your CSCU9A2 folder***, create a new class and paste into it the contents of the file **InputLoop.txt** that you opened above (*select and copy* in Notepad, and then *paste* in BlueJ).
- Try compiling the project. There are many syntax errors in the Java code, so you will need to go around the compile/correct cycle until the errors are gone. *Be patient!*
- The **InputLoop** program shows how we can use a loop for input validation, displaying an error message if a user types in the wrong kind of input and inviting them to enter another input. Try it out to make sure that it works properly.
- Add a declaration of a new variable to act as a counter, and appropriate statements to count each *invalid (bad)* input. When a valid integer is finally entered, display the number of invalid inputs that had been entered, as well as the final valid integer. Try it out to make sure that it works properly.

**Remember: Your code must be well formatted and clearly commented.**

## InputMethod

- Save a copy of your **InputLoop** *project folder* as **InputMethod** (use BlueJ's **Project** menu, **Save As...**) You are going to work on **InputMethod** for a while.
- Introduce a new method into your program called `readInteger` with this method header:

  ```
  private static int readInteger()
  ```

  It receives no parameters, and should *return* the next available integer typed by the user, inviting them repeatedly to input an integer until they do so. You can obtain the Java for this method's body by copying from the body of the `main` method, with minor edits. Note: The method should *not display* the valid integer once it has been received and does not need to count invalid inputs – it should simply return the integer.
- Now *replace* the statements in the body of the `main` method with statements using a `for` loop to input ten integers (using a single statement in the loop body that calls `readInteger)`, to add them together as they are input, and to display their average after the input process has finished.
- Try out your **InputMethod** program to make sure that it works properly.

**Remember: Your code must be well formatted and clearly commented.**

**Checkpoint [Recap]:**

**Show a demonstrator your final code for the InputMethod program, and demonstrate it working correctly. Answer any questions they ask you.**

**For most benefit, continue to the work after the checkpoint, on the next page.**

## **InputArray** – there is no checkpoint for this, but it is valuable practice!

- Save a copy of your **InputMethod** project as **InputArray**. You are now going to work on **InputArray**.
- Introduce a new method into your program called `readArray` with this method header:

  ```
  private static void readArray(int[] a)
  ```

  It receives a single parameter, an array of `int`s, and should then input ten integers from the user, assigning them to consecutive elements of the received array parameter. The method returns no result – the information to be returned has been assigned to the array elements. You can obtain the Java for this method's body by copying from the body of the `main` method, with minor edits.
- Now *replace* the statements in the body of the `main` method with the declaration and creation of an array of ten `int`s, a call of `readArray` to fill the declared array with data, and finally an enhanced `for` loop to add all the array elements together and display their average.

  Hint: You declare and set up an array in Java like this:

  ```
  int[] myList = new int[size];
  ```
- Try out your **InputArray** program to make sure that it works properly.
- **Finally make sure that your program is formatted neatly, and clearly commented.**

That's all for this worksheet.

SBJ 22 January 2017