2020 CI401
Introduction to programming

Week 1.03
Input, more loops and choices

Dr Roger Evans
Module leader

20th October 2020

Review of week 1.02

- For loops
  - Arrays (lists)
  - Variables (names for values)
- If statements
  - Test expressions
  - Else clauses
  - Else if constructions
- Core principles of programming:
  - Sequence – multiple instructions executed in order
  - Selection – making choices between one set of instructions and another
  - Iteration – repeating a set of instructions several times
- Estate agent bot

Input and Output

Communicating with your program

Input and output

- A program is not much use if it cannot talk to the outside world
- Standard computer terminology for this is input and output, often abbreviated to IO
  - Input – information provided to the program from ‘the outside world’
  - Output – information the program provides to the ‘the outside world’
- We might draw a simple picture of IO like this:

Human IO

- A simple human version of IO is talking – if you hear someone talking to you it is input (to you), and if you talk it is output (by you)
- Of course we have other ways of getting input, such as seeing. In fact the human senses (sight, hearing, touch, taste, smell) are all different kinds of input to a human
- And as well as talking, we can change our appearance (eg smile), move etc, as well as higher level output like writing, drawing or making music
- And in conversations, input and output go in both directions between participants – output by you is input for me and vice versa.

IO in programs

- Computer programs are similar.
- In the simplest form they have one input ‘sense’ and one output, each of which can convey data (such as numbers or Strings)
- And a program communicates with human users, or other programs, or the internet, or specialised hardware (such as a robot).

- Actually they don’t communicate directly with humans – they communicate with devices, such as a keyboard, mouse or screen, but we often think of them as communicating ‘with us’

Input and output in Java

- When a Java program runs, it has one input device and one output device provided to it (by ‘the system’) for basic communication.
- These are called System.in and System.out

- We have already used System.out to print out messages on the screen
- Today we will also use System.in to get input from the user (keyboard)
- And we will use a Java library called Scanner to do so

Scanner

A Scanner example

Scanner is a Java library that makes it easy for a program to get input from the user.

Here is a little program called SayHello which uses Scanner to ask the user for their name, and then says Hello to the user.

SayHello - 2019-CI401-week1.03-lab

Class Edit Tools Options

SayHello

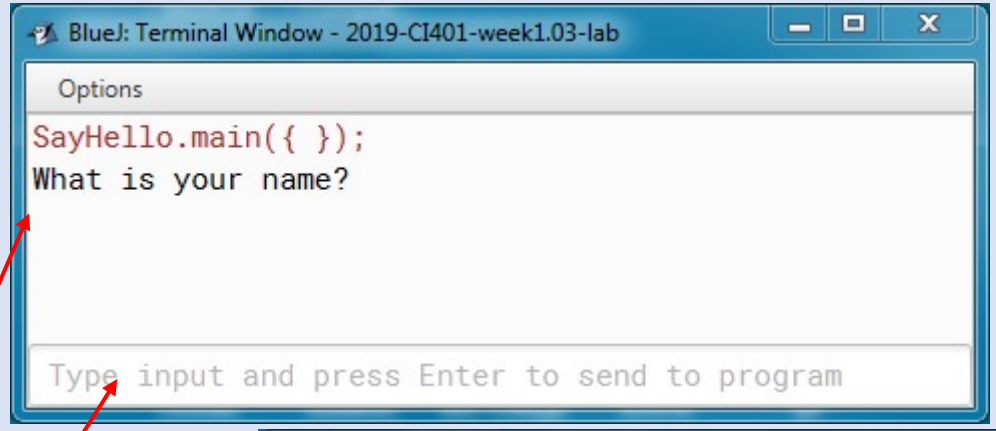
Compile Undo Cut Copy Paste Find... Close Source Code

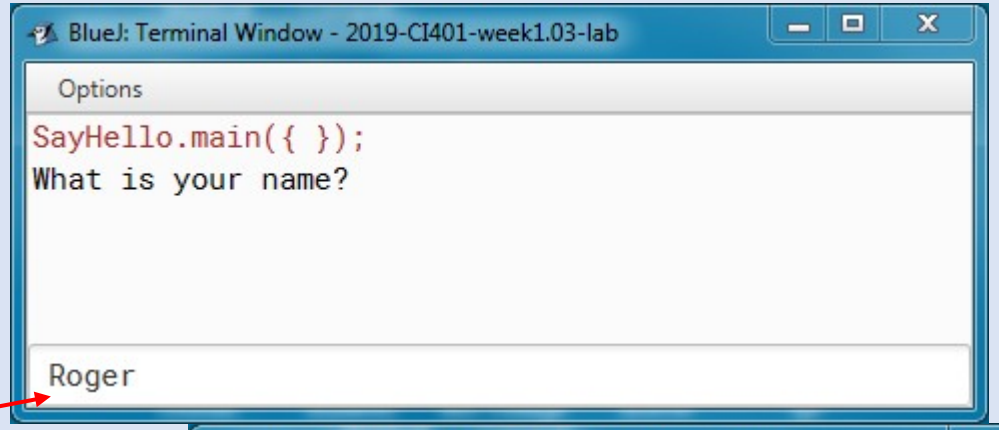
```java
// Tell Java you want to use the Scanner library
import java.util.Scanner;

public class SayHello
{
    public static void main(String[] args) {
        // create a new Scanner object which uses System.in for its data,
        // and call it myInput
        Scanner myInput = new Scanner(System.in);

        System.out.print("What is your name? ");
        String name = myInput.next();
        System.out.println("Hello " + name + " - pleased to meet you");

        // remember to 'close' the Scanner (so the system knows you have finished
        // with it)
        myInput.close();
    }
}
```

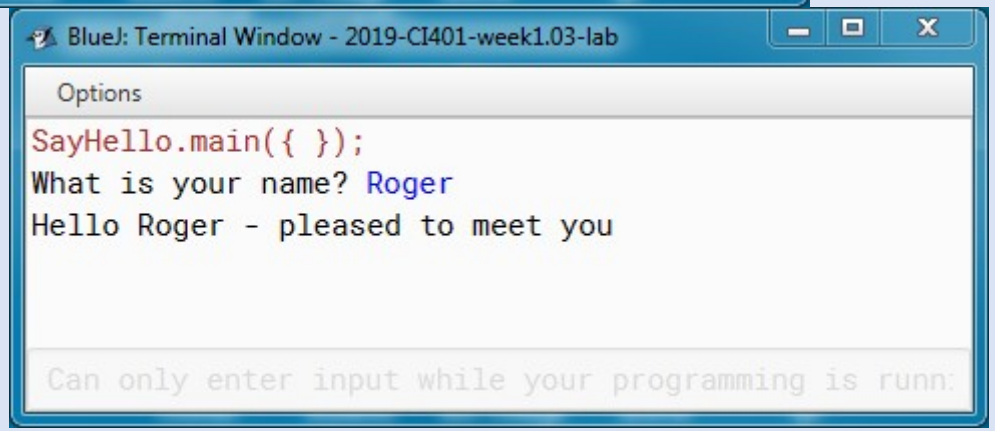
saved

Running SayHello

- First it prints a message (we call this a prompt) to ask me to type something
- It uses Java's output window, but notice the line at the bottom where I can type input as well
- I type 'Roger' and when I press <enter> it gets sent to the program
- The program responds by greeting me!







How it works

We need to do several things to use Scanner:

1. We tell Java we want to use the Scanner library
2. We create a new variable called myInput which holds a new Scanner object which uses System.in to get its data
3. We write code which uses myInput.next(); whenever it wants something from the user
4. We close the Scanner object at the end

SayHello - 2019-CI401-week1.03-lab

Class Edit Tools Options

SayHello

Compile Undo Cut Copy Paste Find... Close Source Code

```java
// Tell Java you want to use the Scanner library
import java.util.Scanner;

public class SayHello
{
    public static void main(String[] args) {
        // create a new Scanner object which uses System.in for its data,
        // and call it myInput
        Scanner myInput = new Scanner(System.in);

        System.out.print("What is your name? ");
        String name = myInput.next();
        System.out.println("Hello " + name + " - pleased to meet you");

        // remember to 'close' the Scanner (so the system knows you have finished
        // with it)
        myInput.close();
    }
}
```

saved

Using Scanner

- Once you have a Scanner object, like myInput, you can use it to ask the user for several different kinds of thing.
- The example uses myInput.next() which gets the next thing you type until you type a space or <enter>
- Another common need is to input a number, which you can do with myInput.nextInt() – this returns an integer (whole number), which you have to store in a variable of type int eg int age = myInput.nextInt();
- You sometimes want to get a whole line (including spaces), and you can use myInput.nextLine() for this.

Using Scanner yourself

- SayHello is in this week's lab
- There is also a class called ScannerTemplate (shown here)
- ScannerTemplate does nothing, but has the basic code you need to use a Scanner object
- You can use bits of it to make new programs
- In this week's labs there are two copies of ScannerTemplate, called QueenBot and HolidayPlanner, for you to fill in with code (see Lab1 and Lab3)

ScannerTemplate - 2019-CI401-week1.03-lab

Class Edit Tools Options

ScannerTemplate

Compile Undo Cut Copy Paste Find... Close Source Code

```java
// Tell Java you want to use the Scanner library
import java.util.Scanner;

// This class provides the 'template code for using Scanner. It doesn't
// do anything, but you can copy it and add your own code in the middle
// which uses myInput to get user input

public class ScannerTemplate
{
    public static void main(String[] args) {
        // create a new Scanner object which uses System.in for its data,
        // and call it myInput
        Scanner myInput = new Scanner(System.in);

        // Put your own code here. This code can use scanner to get information
        // from the user. For example:
        //     String home = myInput.next();
        //     int myAge = myInput.nextInt();

        // remember to 'close' the Scanner (so the system knwos you have finished with
        myInput.close();
    }
}
```

Class compiled - no syntax errors

saved

More loops – the ‘while’ loop

While loops

- We saw last week how we can make a program run the same piece of code several times on each item of a list, using a for loop.
- Java has several other ways of making loops, and we are now going to look at the while loop
- A while loop does not have a list of things to run the loop over. Instead it has a test (like an if statement) which it checks each time round the loop.
- As long as the test is true, the loop will run again. If the test is false the loop stops.
- Of course, something needs to change during the loop, to make the test change, otherwise it will loop forever!

While loops in Java

- A Java while loop looks like this:

```
while ( test ) {
    body
}
```

- You can read it like this: ‘while the test is true, run the body again’

- Notice the general structure is just like the simplest form of if statement, only with the keyword while instead of if

While loop example

There are some new things here:

1. At the top we have a variable, but it contains a number not a String. We tell Java it's a number variable with the word int (short for integer – which is maths-speak for 'whole number').
2. The test checks whether x is greater than 0. This is a new kind of test, again from maths. Since we just set it to be 2, this will be true, so the loop body will run.
3. The body print Hello, and then runs the line x = x – 1; . Look carefully at this – it is NOT an equation. It say 'set the (new) value of x to be the (old) value of x minus one' (ie 'subtract 1 from x')
4. We go back to the while line and do the test again. x is now 1, so still greater than 0, so we run the body again.
5. We go back to the top, but now x is 0, so the loop stops, having printed Hello twice.

```
int x = 2;
while ( x > 0 ) {
    System.out.println("Hello");
    x = x - 1;
}
```

Scanner and while loops

Let's use a while loop to extend the SayHello example.

- Here is HelloEverybody, a program which says hello to lots of people
- Instead of asking for just one name, it uses a while loop to keep asking for names until the name given is the word "end".
- Notice the last line of the loop body reads in the next name. This is essential to make sure something about the loop test changes.
- Notice also the ! in the test – this means not – in other words we want the opposite of the test (nextPerson is NOT EQUAL TO "end")

HelloEverybody - 2019-CI401-week1.03-lab

Class Edit Tools Options

HelloEverybody

Compile Undo Cut Copy Paste Find... Close Source Code

```java
// Tell Java you want to use the Scanner library
import java.util.Scanner;

public class HelloEverybody
{
    public static void main(String[] args) {
        // create a new Scanner object which uses System.in for its data,
        // and call it myInput
        Scanner myInput = new Scanner(System.in);
        String nextPerson = "";

        System.out.print("What are your names? ");
        nextPerson = myInput.next();

        while (! nextPerson.equals("end")) {
            System.out.println("Hello " + nextPerson + " - pleased to meet you");
            nextPerson = myInput.next();
        }

        // remember to 'close' the Scanner (so the system knows you have finished with
        myInput.close();
    }
}
```

Class compiled - no syntax errors

saved

More choices – the ‘switch’ statement

Another way of making choices

- Last week we talked about different ways of writing an ‘if’ statement
- The most complex one was when we have several different conditions, to test, one after another, and looked like this:

```
if ( test 1 ) {
    instructions to run if test 1 is true
} else if ( test 2 ) {
    instructions to run if test 2 is true
} else if ( test 3 ) {
    instructions to run if test3 is true
} else {
    instructions to run if all the test are false
}
```

EstateAgent

- Quite often, all these conditions test the same variable for different values.
- The EstateAgent class (in this week's lab) is just example 4 from last week.
- In it we tested room several times to see if it was a room we wanted to make a comment on
- Java provides a neater way of doing, called the switch statement

```
public static void main(String[] args)
{

    String lastRoom = "";
    String lastButOneRoom = "";
    for (String room: args)
    {   // using 'print' instead of 'println' to stop each
        // part printing on a different line
        System.out.print("Here we are ");
        //see whether we have just come back here
        if (room.equals(lastButOneRoom)) {
            System.out.print("back ");
        }
        System.out.print("in the "+room + "  ");
        if (! room.equals(lastButOneRoom)) {
            // comment about individual rooms.
            if (room.equals("kitchen")) {
                System.out.print("Can you smell the coffee? ");
            } else if (room.equals("bedroom")) {
                System.out.print("Plenty of room for all your clothes.");
            } else if (room.equals("lounge")) {
                System.out.print("Notice the original fireplace.");
            }
        }
        System.out.println();
        lastButOneRoom = lastRoom;
        lastRoom = room;
    }
}
}
```

The Java switch statement

```
switch ( expression ) {
  case value1:
    instructions for label1;
    break;
  case value2:
    instructions for label2;
    break;
  default:
    instructions for no matching case;
}
```

The parts of a switch statement

- A switch statement is a bit like an if statement, except that the part between the round brackets is not a test (something true or false), but instead an expression which returns a value (usually a number or a String)
- And the body of the switch statement is a set of cases
- A case is labelled with one (or more) possible values, and has a block of instructions to run if the expression's value is that particular label. Usually the last instruction is the special instruction break;
- You can also have a default case, which is like an else statement – what to do if the value of the expression is not any of the case labels

Week 1.02 Example 4 again

```
if (! room.equals(lastButOneRoom)) {
    // comment about individual rooms.
    if (room.equals("kitchen")) {
        System.out.print("Can you smell the coffee? ");
    } else if (room.equals("bedroom")) {
        System.out.print("Plenty of room for all your clothes.");
    } else if (room.equals("lounge")) {
        System.out.print("Notice the original fireplace.");
    }
}
```

EstateAgent with a switch statement

```
switch ( room ) {
   case "kitchen":
   System.out.print("Can you smell the coffee? ");
   break;
   case "bedroom" :                    System.out.print("Plenty of room for
all your clothes. ");
   break;
   case "lounge" :                     System.out.print("Notice the original
fireplace. ");
   break;
   default:
   break;
}
```

The ‘break’ statement

- You might wonder what those break statements were for
- They tell Java that you have finished the switch statement completely and want to go on to the next thing
- (NB: you can use a break statement in a loop body too, to jump right out of the loop early)
- Without it, Java just carries on with the next case of code, as well as the one just completed.
- This may seem strange, but it is sometimes useful.

Month lengths

This switch statement takes a month name and tells us how many days there are in the month.

- Notice that some months have no code body at all, which means they just fall down into the case below them.
- So the effect is that one block of code has multiple labels.
- Notice also the default case – for all the months not named

```
switch ( month ) {
  case "april":
  case "june":
  case "september":
  case "november":
  System.out.println("30 days");
  break;
  case "february":
  System.out.println("28 days");
  break;
  default:
  System.out.println("31 days");
  break;
}
```

Star ratings

Here's an example where all the cases do have code, but they still fall into one another (because there are no break statements)

So this code prints out as many stars as the number in the variable 'stars' tells it to.

```
switch ( stars ) {
   case "5": System.out.print("*");
   case "4": System.out.print("*");
   case "3": System.out.print("*");
   case "2": System.out.print("*");
   case "1": System.out.print("*");
   default:
      System.out.println("");
      break;
}
```

Week 1.03 Labs

Lab exercises – BlueJ

- Create a folder for this week's work on your S: drive eg at S:\CI401\week1.03
- Download BlueJ project 2020-CI401-week1.03-lab.zip from myStudies into this folder
- Open BlueJ on your computer and create a new project from the zip file in your new folder
- BlueJ will show you a folder full of Example files and Lab exercises

Lab exercises – coding

- Open each of the example files (SayHello, HelloEverybody, EstateAgent and StarRating) by double clicking them and look at the code. Try to understand what each one does (look at the lecture slides too), and then compile and run it to see if you were right
- Open each of the lab (Lab1, Lab2 and Lab3) files and follow the instructions in them to create QueenBot and HolidayPlanner.
- Remember to save your work to your S: drive before finishing.
- If you want to access your labs at home as well, remember to copy the week1.03 folder to your O: drive

Lab exercises – we are here to help!

- **If you get stuck, ask for help!**
- **Even if you don’t get stuck, talk to us!**