COMP122 Week 3

UNIVERSITY OF LIVERPOOL

Dr. Patrick Totzke
totzke@liverpool.ac.uk

https://liverpool.instructure.com/courses/59716

Flow Control

if-else example

What will the following code snippet do?

```
1 if (myValue > 0)
2   if (myValue % 2 == 0)
3     System.out.println("positive and even");
4 else
5   System.out.println("not positive");
```

Test Case	Expected Result	Actual Result
2	Prints “positive and even”	“positive and even”
1	Prints nothing	“not positive”
-1	Prints “not positive”	nothing
-2	Prints “not positive”	nothing

if-else example

We can (and often, must) use brackets to enforce what we intend!

```
1 if (myValue > 0) {
2   if (myValue % 2 == 0)
3      System.out.println("positive and even");
4   }
5 else
6   System.out.println(" Not positive.");
```

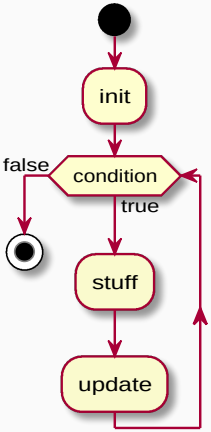
To work out which `if` an `else` relates to, the Java compiler uses the closest `if` without an `else`.

Note it is useful to use indentation to make it clear what is going on, but indentation **does not** enforce the association of an `else` with a particular `if`.

If in doubt, use curly brackets around groups of statements to enforce your intention.

For Loops

...are used to repeat a block of code (for a fixed number of times).



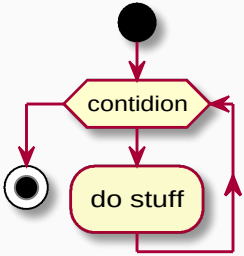

```
for (initialization;
     condition;
     update)  {
       // stuff to repeat
}
```

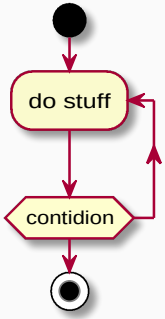
To print the first 10 integers

```
for (int i = 1; i <= 10; i++){
    System.out.println(i);
}
```

While Loops

```
1 while ( condition ) {
2   // stuff to repeat
3 }
```






```
1 do{
2   // stuff to repeat
3 } while ( condition );
```

Methods

So what exactly is...

```
System.out.println("Hello!");
```

? ...is a statement in which we call a method named `System.out.println` with a single `String` parameter `"Hello!"`.

Here's another one:

```
double twoSquared = Math.pow(2.0,2.0);
```

It calls a method named `Math.pow` that has two parameters of type `double` and returns a `double`.

Methods

. . . are named code blocks. They

- are defined inside a class definition.
- can have arguments and a return value.
- correspond to “functions” or “procedures” in other languages.

Defining a method looks like this

```
1  modifiers returnType methodName (parameters){
2    //  method body code here
3  }
```

- `modifiers` determine how the method can be accessed.
- `returnType` is the type of the returned value.
- `methodName` an identifier
- `parameters` a (comma-separated) list of arguments, each given as a type and an identifier. Order matters!
- `//method body` the code block that defines the method's behaviour.

Examples

```
1  public class MaximumDemo {
2
3    public static int maximum(int a, int b) {
4      if (a >= b)
5        return a;
6      else
7        return b;
8    }
9    public static void main(String[] args){
10     int x = 27;
11     int y = 13;
12     System.out.print("The maximum of " + x + " and " + y
13                      + " is " + maximum(x,y));
14   } // end of main method
15 } // end of class
```

A method is uniquely identified by its signature!

The modifiers, spelling of the identifier, types and orderings of the parameters together form the signature of the method.

The following five methods are all different.

```
1 public static int max(int a, int b) {}
2 public static int maX(int a, int b) {}
3 public static int max(int a, double b) {}
4 public static int max(double b, int a) {}
5 public int max(int a, int b) {}
```

```
6 public static int max(int b, int a) {}  // same as 1
```

```
7 public static void main(String[] args) {}  // special
```

```
8 public static void main(int n) {}  // NOT special
```

Example: A random walk

Drunkard's Walk

A person, having had several drinks at one of their local pubs, starts to walk home. At each step, he is equally likely to walk one street to/away from their home. How long (on average) does it take to reach his home if he starts n streets away from home?

A first step

Firstly, just consider doing this experiment once. (We want to repeat it many times, as we're looking for an average, but how do we simulate this "random walk"?)

We consider their current position (which, say, starts at the integer 5) and update their current position by adding or subtracting one each time step, which could occur with equal chance.

We utilize some Java `Math` library functions to help us out.

Generating a (random) value of +1/-1

`Math.random()` gives a (decimal) number in the range $0 \leq x < 1$.

So `2*Math.random()` gives a number in the range $0 \leq x < 2$.

`Math.floor(x)` gives the "floor" of x, i.e. the largest integer that is less than or equal to x. (Examples: `Math.floor(1.336)= 1`, `Math.floor(3.2)= 3`, `Math.floor(-2.5)= -3`, etc.)

`Math.floor(2*Math.random())` will give the integer 0 or 1 (with equal probability).

`Math.pow(x, y)` will give the value x^y, i.e. x raised to the power y.

So `Math.pow( (-1), Math.floor(2*Math.random()))` will give the value $+1$ or -1 with equal probability.

Once around the block...

```
1  public class DrunkOne {
2
3     public static void main(String[] args) {
4        final int START = 5;
5        final int END = 0;
6        int numberOfSteps = 0;
7
8        //Take a step to the left or to the right
9        for (int position = START;
10                position > END;
11                position += (int) Math.pow((-1),
12                                           Math.floor(2*Math.random()))){
13           System.out.print(position + " ");
14           numberOfSteps++;
15       }
16       System.out.println("\nNumber of steps taken: " + numberOfSteps);
17    }
18 }
```

Once around the block...

Running the program will give you the sequence of steps taken (i.e. integers, one after another that differ by one), but doesn't show the final integer (0) representing reaching home.

```
$ java DrunkOne
5 4 5 4 5 6 5 4 3 2 1

Number of steps taken: 11
```

Other paths of the drunkard...

Running this program again will give different output, as the steps are generated randomly.

```
$ java DrunkOne
5 6 7 8 9 10 9 10 9 8 9 10 11 12 11 10 11 10 11 10
9 8 7 8 9 10 9 8 7 6 5 6 7 6 5 6 5 6 7 8 7 6 7 6 5 4 5
6 5 6 5 6 5 4 3 4 5 4 5 6 7 6 7 6 5 4 5 4 3 2 3 4 3 2 3
4 5 4 3 2 3 4 5 6 5 4 5 4 5 4 3 2 3 4 3 2 3 4 5 6 5 4 5 6
5 4 3 2 1 2 1 2 1

Number of steps taken: 113
```

```
$ java DrunkOne
5 6 5 6 5 4 3 4 5 6 5 6 7 6 5 6 5 4 3 4 3 2 1

Number of steps taken: 23
```

Some can be quite long (in the thousands of steps, or even more).

Multiple repetitions (I)

In order to get an average, we'll repeat this many times.

We'll wrap our program in another `for` loop, and average the total number of steps.

```
1  public class DrunkardWalk {
2
3     public static void main(String[] args) {
4        final int START = 5;
5        final int END = 0;
6        final int NUMBER_OF_TRIALS = 2000;
7
8        long totalSteps = 0;
9
10       for (int i = 1; i <= NUMBER_OF_TRIALS; i++){
11             int numberOfSteps = 0;
12
13             //Take a step to the left or to the right
14             for (int loopParameter = START;
```

Multiple repetitions (II)

```
15                      loopParameter > END;
16                      loopParameter +=
17                           (int) Math.pow( (-1),
18                                  Math.floor(2*Math.random()) ){
19                  numberOfSteps++;
20              }
21              totalSteps = totalSteps + numberOfSteps;
22          }
23
24          double averageSteps = 1.0*totalSteps/NUMBER_OF_TRIALS;
25          System.out.print("Average over " + NUMBER_OF_TRIALS + " trials: ");
26          System.out.println(averageSteps);
27      }
28  }
```

An average number of steps

We're averaging over 2000 trials. By declaring this number as a constant, it's easy to increase/decrease this number by changing one line of code.

```
$ java DrunkardWalk
Average over 2000 trials: 82885.269
$
```

Schedule for Thursday

1. Q&A
2. Some errors flagged by CodeGrade
3. Assignment 1 brief

Your Questions?

Q16

Q: I have completed the assignments due in today, but have had some trouble with submitting them into the self marking software - shall I email them to you separately?

A: No!

I cannot accept submissions by email. Speak to a TA or post your problem on Canvas to get help.

Q17

Q: Is it still 5% lost per day?

A: Only for the three *assignments* (up to max 5 days).
For the lab exercises you loose 100% after 5pm on Friday.

Q18

Q: Can I use Java 17?

A: Yes

(This was actually about the lab computers having an old oracle java installed)

Q19

```
Hi.
Last week I did both of the labs but forgot to submit the second one. How much
     does this affect my grade for the module?
```

A: about 0.5%. ($\rightsquigarrow$ FAQ 3.2)

Q20

Q: Will I get full credit for a "decent attempt" like in other modules?

A: Yes and No ($\rightsquigarrow$ FAQ 3.2)

Q21

Q: I accidentally submitted the wrong file for a lab. Can I delete my submission and re submit my other file?

A: yes, just submit again

Q22

Hi!

I was just wondering since the labs were worth under 1% per one lab is it okay to not submit anything at all. I have been trying to do the labs for 20+ hours and researching etc but still have no idea how to do them. I understand the tasks and know how to do them maths wise, but not in java. I don't even know where to start so it isn't anything I can ask in the discussions either.

It might help to see the correct code and study from that.
I know some teachers need everything submitted to pass so I was wondering is this the case for Comp122.

Kind regards,

Some Errors Flagged by CodeGrade

No	Summary	Score	Pass
1	**Compile** Run `javac Declarations.java` and check for successful completion.	0 / 1	✖

Output | Errors

Command line

```
javac Declarations.java
```

Exit code

1

Output

```
No output.
```

No	Summary	Score	Pass
1	**Compile** Run `javac Declarations.java` and check for successful completion.	0 / 1	✖

Output | Errors

```
1. Declarations.java:8: error: cannot find symbol
2.         weightDifference = newWeight - curentWeight;
3.                                        ^
4.   symbol:   variable curentWeight
5.   location: class Declarations
6. 1 error
```

```
for(i=1;i<5;i++) {
    System.out.println("Soon may the compiler come,");
    System.out.println("To bring class files and exceptions");
}
```

```
CShanty.java:6: error: cannot find symbol
        for(i=1;i<5;i++) {
            ^
  symbol:   variable i
  location: class CShanty
```

```
1.  public class Largest {
2.      public static void main(String [] args) {
3.  →   System.out.println("Length of Array:");
4.          int n = Comp122.getInt("");
5.          //int n=Integer.parseInt(args[]);
6.          int[] myArray = new int[n];
7.          for (int i=0;i<n;i++) {
8.              myArray[i] = Comp122.getInt("Enter an Integer:");
9.          }
10.
11.         int largestValue = myArray[0];
```

3.1 **test empty sequence** Run `java Largest` and match its output to an expected value.

Output | Difference | Input | Errors

```
1.  Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0
2.  →   at Largest.main(Largest.java:3)
```

```
1.
2. public class Factorial {
3.     public static void main(String[] args) {
4.         System.out.println("Enter an integer:");
5.         int n = Comp122.getInt();
6.         int factorial = 0;
```

3.1 **0** Run `java Factorial` and match its output to an expected value. 0 / 1

Output | Difference | Input

Expected output

```
1. Enter an integer: 1
```

Actual output

```
1. Enter an integer:
2. 0
```

```
1. public class Largest {
2.     public static void main() {
3.         int n = Integer.parseInt(args[0]);
4.         int[] myArray = new int[n];
5.
```

1 **Compile** Run `javac Largest.java` and check for successful completion. 0 / 1 ✖

Output | Errors

```
1. Largest.java:3: error: cannot find symbol
2.         int n = Integer.parseInt(args[0]);
3.                                  ^
4.   symbol:   variable args
5.   location: class Largest
6. 1 error
```

```
1. public class Factorial {
2.     public static void main(String[] args) {
3.         int n = Comp122.getInt("Enter an integer: ");
4.         int i = 1;
5.         int factorial = 1;
```

3.8 **15** Run `java Factorial` and match its output to an expected value. 0 / 1 ✖

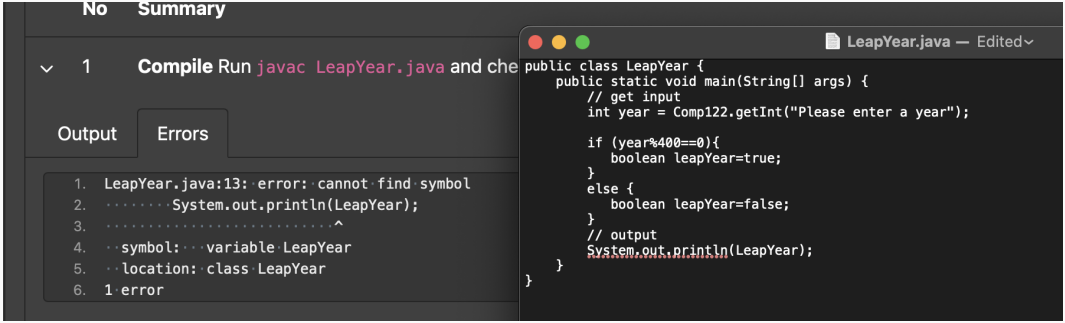
Output | Difference | Input

Expected output

```
1. Enter an integer: 1307674368000
```

Actual output

```
1. Enter an integer: 2004310016
```


No
Summary
1
Compile Run javac LeapYear.java and che
Output
Errors
1. LeapYear.java:13: error: cannot find symbol
2. System.out.println(LeapYear);
3. ^
4. symbol: variable LeapYear
5. location: class LeapYear
6. 1 error
LeapYear.java — Edited
public class LeapYear {
public static void main(String[] args) {
// get input
int year = Comp122.getInt("Please enter a year");
if (year%400==0){
boolean leapYear=true;
}
else {
boolean leapYear=false;
}
// output
System.out.println(LeapYear);
}
}

Assignment 1

Summary of Week 3

We looked at...

- Loops
- Methods
- Drunkards
- CodeGrade submissions
- Assignment 1 overview

Next Week

- Tuesday: Javadoc, Version Control
- Thursday: No new materials
- Friday: Assignment 1 submission!