COMP122 Week 2

The compiler is your friend!

Dr. Patrick Totzke
totzke@liverpool.ac.uk

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

Today's lecture

- Common programming errors
- Static vs dynamic type systems
- Literals
- Type Casting
- Constants

Common Programming Errors

What does this program do?

```
/*
  * Author: Patrick Totzke
  * The Welcome class implements an application */

public class Welcome {
     // ---------------METHODS--------------
     /*  Main Method */
     public static void mian(String[] args) {
         System.out.println("Welcome to COMP122")
         System.out.Println("This is going to be easy!");
}
$> javac Welcome.java
Welcome.java:4: error: class, interface, or enum expected
  * that prints out a welcome message.
  ^
Welcome.java:11: error: ';' expected
   System.out.println("Welcome to COMP122")
                                           ^
Welcome.java:12: error: unclosed string literal
   System.out.Println("This is going to be easy!);
                      ^
Welcome.java:12: error: ';' expected
   System.out.Println("This is going to be easy!);
                                                     ^
Welcome.java:14: error: reached end of file while parsing
}
 ^
Welcome.java:15: error: reached end of file while parsing
6 errors
```

After fixing the reported errors

```
/*
  * Author: Patrick Totzke
  * The Welcome class implements an application
  * that prints out a welcome message.
*/

public class Welcome {
     // ---------------METHODS-------------
     /*  Main Method */
     public static void mian(String[] args) {
         System.out.println("Welcome to COMP122");
         System.out.Println("This is going to be easy!");
    }
}
```

```
$> javac Welcome.java
Welcome.java:12: error: cannot find symbol
         System.out.Println("This is going to be easy!");
                   ^
  symbol:   method Println(String)
  location: variable out of type PrintStream
1 error
$
```

How about now?

```java
/*
  * Author: Patrick Totzke
  * The Welcome class implements an application
  * that prints out a welcome message.
*/

public class Welcome {
     // ---------------METHODS-------------
     /*  Main Method */
    public static void mian(String[] args) {
        System.out.println("Welcome to COMP122");
        System.out.println("This is going to be easy!");
    }
}
```

Yes!

```
$> javac Welcome.java
$>
```

However, we get run-time errors:

```
$> java Welcome
Error: Main method not found in class Welcome, please define the main method
    as:
   public static void main(String[] args)
or a JavaFX application class must extend
javafx.application.Application
```

Welcome Program (Correct)

```java
/*
  * Author: Patrick Totzke
  * The Welcome class implements an application
  * that prints out a welcome message.
*/
public class Welcome {
     // ---------------METHODS-------------
     /*  Main Method */
     public static void main(String[] args) {
         System.out.println("Welcome to COMP122");
         System.out.println("This is going to be easy!");
     }
 }
```

Welcome Program (a little nicer)

```java
/**
 * An application that prints out a welcome message.
 * @author Patrick Totzke <totzke@liverpool.ac.uk>
 */
public class Welcome {
	public static void main(String[] args) {
		System.out.println("Welcome to COMP122");
		System.out.println("This is going to be easy!");
	}
}
```

Data Types and Variables

Declarations, initialisations, assignments

```java
double currentWeight;
int studentsLearningJava;
int maximumValue;
studentsLearningJava = 78;
maximumValue = 120;
maximumValue = studentsLearningJava;
```

Recall: HelloWorld v2 (from the Variables video)

```java
public class HelloWorld2 {
    public static void main(String[] args) {

        // declare and initialize a string
        String output = "Hello ";

        // change it
        output = output + "World!";

        // use it
        System.out.println(output);
    }
}
```

Why Types?

Better Question:

Why use a strongly typed language?

Why use a strongly typed language?

For Your Safety!

Compared to dynamically typed languages (python), a strong type system

- prevents accidental (or malicious) memory access
- captures preventable runtime errors at compile time

```
int x = 1;
String y = "two";
...
x = y;  // compiler complains, not your users!
```

Let's check out `bad.py`

Quick Quiz

Which type does the following literal have?

- `"A"` ...is a `String`
- `’A’` ...is a `char`
- `0.5` ...is a `double`
- 56 ...is an `int`
- 2147483647 ...is an `int`
- 2147483648 ...is `long` but *not* `int`

```
jshell> int a = 2147483648;
|   Error:
|   integer number too large
|   int a = 2147483648;
|           ^
```

On the range of integers see `https://stackoverflow.com/questions/14020097/the-range-of-int-in-java`

Type Casting

is when you assign a value of one primitive data type to a variable of another type.

1. Widening Casting (automatically)

   `byte` → `short` → `char` → `int` → `long` → `float` → `double`

```java
int myInt = 9;
double myDouble = myInt;
System.out.println(myInt);      // Outputs 9
System.out.println(myDouble);   // Outputs 9.0
```

2. Narrowing Casting (manually)

   `double` → `float` → `long` → `int` → `char` → `short` → `byte`

```java
double myDouble = 9.78;
int myInt = (int) myDouble;
System.out.println(myDouble);   // Outputs 9.78
System.out.println(myInt);      // Outputs 9
```

Type Casting

is when you assign a value of one primitive data type to a variable of another type.

1. Widening Casting (automatically)

   byte → short → char → int → long → float → double

```java
int myInt = 9;
double myDouble = myInt;
```

2. Narrowing Casting (manually)

   double → float → long → int → char → short → byte

```java
double myDouble = 9.78;
int myInt = (int) myDouble;
System.out.println(myDouble);   // Outputs 9.78
System.out.println(myInt);      // Outputs 9
```

https://www.w3schools.com/java/java_type_casting.asp

Constants

Constant Declarations

Constants $\sim$ variables whose values don't change. Declare via

```
final <type> <CONSTANT_NAME> = <value>;
```

- Here `final` is a keyword to denoting that the following is a constant.
- <`type`> and <`value`> are as previously.
- <`CONSTANT_NAME`> is a legal identifier.

By convention identifiers denoting constants are in all caps, e.g.:

```
final double ASSIGNMENT_WEIGHT = 0.2;
final int SURVEY_TOTAL = 763;
```

Q: Why bother with constants?

Why Use Constants?

- Efficiency: Compiler optimizations will result in smaller and binary code and smaller memory footprint.
- Readability: e.g.

```
result = ASSIGNMENT_WEIGHT * assignment
       + LABS_WEIGHT * lab
       + EXAM_WEIGHT * exam;
```

  instead of

```
result = 0.75 * assignment + 0.10 * lab + 0.15 * exam;
```

- Ease of maintenance: Imagine a set of financial programs that include VAT calculations. This was 15% in 2009, 17.5% in 2010, and 20% in 2011. If VAT is defined as a constant, it only has to be changed *once*, and the changes propogate throughout the program. `final double VAT = 20.0;`

Schedule for Thursday

1. Q&A
2. Control Flow:
    - Boolean expressions
    - if-then-else
    - switch
3. Syntax baggage from C

Your Questions?

Q5

Q: When is the deadline for labs?

A: Fridays 5pm;

Q6

Q: Do I have to bring a PC to the lectures?

A: No, but you are welcome to.

Q7

Q: Are we allowed to use the scanner object instead of using the class that you made for the user inputs?

A: Sure.

Q8

Q: Can you help me set up Java on my Macbook?

A: *I* really can't, but...

- check out the "Setting up your programming environment" page on canvas
- bring it to the labs!

Q9

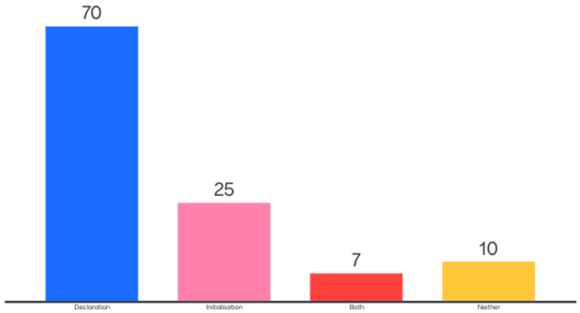
- Where to type code?
- Sir, I don't know I should use Txt or Eclipse to program.
- Which IDE (text editor) to use?
- Can I use VS Code?
- Is apache net beans a good IDE?

A: Use what works best for you...
but make sure your submissions work with the CodeGrade automarker

Q10

Why was double pi a declaration?

```
double pi;
```



Q11

Q: Does the code have to be compiled everytime you make changes to it

A: Yes

Q12

Q: Could you convert the PDF lecture slides into hand out format please

A: sure

Q13

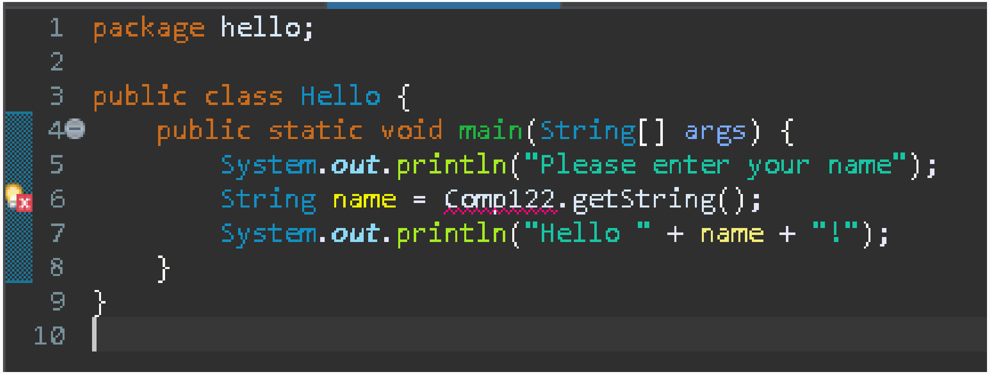
Q: Can you tell the people at the back to stop talking?

Could you tell the girls at the back of the lecture room to be quiet please

“I really, really wish that you had told those at the back who spoke through the lectures and over you to be quiet or leave.”

Q14

Please help me debug. Here is my code so far.



```java
package hello;

public class Hello {
    public static void main(String[] args) {
        System.out.println("Please enter your name");
        String name = Comp122.getString();
        System.out.println("Hello " + name + "!");
    }
}

```

Q15

Q: Is there any way to verify whether or not our submission of a lab has actually gone through?

A: If you see the automarker run, it is ”submitted”. Feel free to overwrite.

Your Feedback – Week 1 – some negatives

“I’m really struggling to understand. I only started coding in September in Comp101 and I have never used Java before. I have really struggled to even set up the programming environment on my pc. I went and downloaded the thing from the link on canvas and ran the wizard but now I can’t work out how to open it. ”

“I’m now trying to use the web-based programming environment but I honestly just don’t understand what I’m meant to do. ”

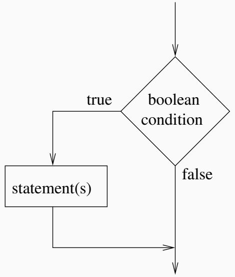
“I think I really could have done with a super basic explanation going through how to set up java on your computer, and then how to begin working on the labs. It seemed like even in the lecture you were struggling to get it set up. ”

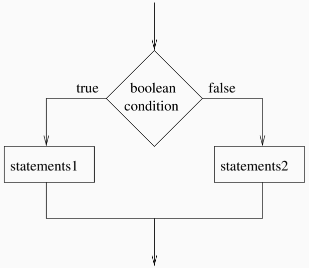
Control Flow

Java's syntax for if-else Statements

```
if ( condition) {
      statements;
}
```

```
if ( condition) {
      statements1;
} else {
      statements2;
}
```





Conditions

What goes in the condition part? Any expression that evaluates to either true or false!

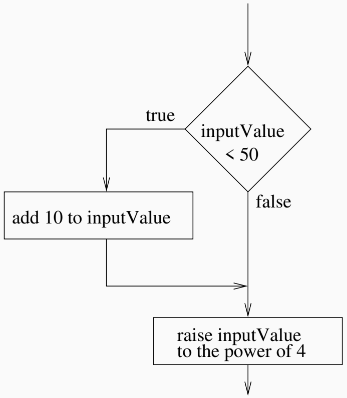
Examples.

1. A comparison between two variables ("if A is greater than B then something happens otherwise something else happens")
2. The check that some expression is not larger than a certain threshold ("if the employee earns more than £100,000 tax them at 75%").
3. Check that input is within the desired range (e.g. disallowing a negative value for a circle radius).

In Java such conditions usually involve `boolean` variables and so-called *relational operators* such as `&&` (and), `||` (or).

Power 4 Example: Flow of Control

Write a Java program that takes a single integer as input. If the input is less than 50, add 10 and return the result raised to the power of 4, otherwise simply return the input raised to the power of 4.



$$f(x) = \begin{cases} (x+10)^4 & \text{if } x < 50 \\ x^4 & \text{otherwise} \end{cases}$$

```java
public class Power4 {

  public static void main(String[] args) {
    int input = 101;

    if (input < 50)
      input = input + 10;

    int result = (int)Math.pow(input, 4.0);
    System.out.println(result);
  }
}
```

The “switch” statement

The `switch` statement is used for making a choice from (usually) several possible outcomes.

You need a variable to compare with a collection of “cases”. You can include a “default” case at the end, that will be executed if none of the cases match.

A “switch” example

Here is an example of a `switch` statement.

```
int x = input.nextInt();

switch (x)  {
    case 0:
    case 1:
    case 2:  System.out.println("x is small");
             break;
    case 3: case 4:  System.out.println("x is less small");
                      break;
    case 5:  System.out.println("x is big");
             break;
    default:  System.out.println("x is strange.");
              break;
}
```

More general syntax

Typically, the "control variable" is an `int` or `char`. The `switch` statement also works with enumerated types (not discussed here), `String`, and certain other special classes.

```
    switch (variable)  {
           case ___ :  //  code
                       break;
           case ___ : //  code
                       break;
           . . .
           case ___ :  //  code
                       break;
           default:  //  code;
                     break;
    }
```

The `break` reserved word ceases execution of the `switch` block. The `default` case is not required.

Just so that you know...

Java inherits some dirty syntax hacks from C/C++

```
int a = 0;
int b = a +=5;

int c = 12 > 10 ? ++a : b++;
```

You can nest these (add brackets as needed) but it gets ugly fast, so don't!

Summary of Week 2

We looked at...

- Common programming errors
- Java's type system
- Constants
- if-then-else and switch statements
- some syntax baggage from C

Next Week

- more control flow
- methods