# Using the Java Eclipse IDE

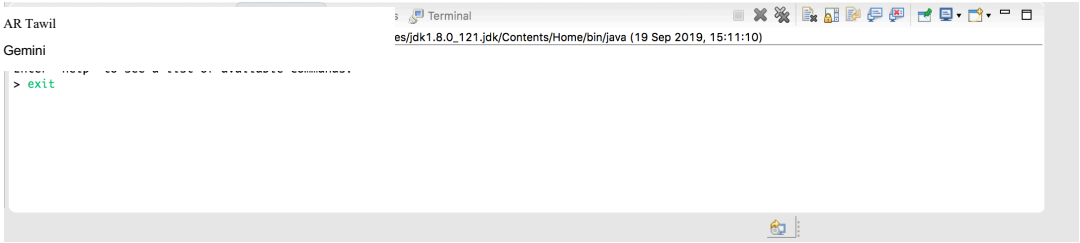
Open Eclipse and then carry out the following tasks:

**a)** Create a NEW WORKSPACE and make sure it's created in you Z: Drive

b) Create a NEW PROJECT (**File > New > Java Project**) - call it **Lab1** (make sure *Empty Project* is selected)

c) ADD A NEW CLASS to your project (**File > New > Class**) - call it **ZodiacApp** (remember to tick the modifier **'public'** and **'generate main method'** boxes).

d) Write, compile and run a program that will display your name and star sign. For example:




To compile and run your file select (**RUN > RUN AS > JAVA APPLICATION**).

If you have errors they will be listed below the program. Fix and re-compile.

Experiment with the **print** and **println** commands.

e) Your work should be automatically saved.

f) QUIT Eclipse by selecting **(ECLIPSE > QUIT ECLIPSE)**

g) Now re-open ECLIPSE and retrieve your work by opening your workspace to ensure that your project is saved correctly.

# Evaluating Java Expressions

Create a project (*evaluate-java-expressions*) and execute each of the Java expressions below. Make a note of any expressions with unexpected results. Pay close attention to avoid syntax errors: for example, "a" and 'a' are not the same in Java.

Add a new class *EvaluateExpressions*

Use System.out.println to print the value of your expression evaluation

e.g.:

```
System.out.println("Arithmetic operations");
System.out.println(1+1);
```

1. Arithmetic operations:
   - 1 + 1
   - 5 - 2 * 3
   - (5 - 2) * 3
   - 4.5 + 6.7
   - 3 - 2.1
   - 6 / 2
   - 7 / 2
   - 7.0 / 2.0
   - 8 % 2
   - 9 % 2

2. Comparison operations:
   - 1 + 1 == 2
   - 1 + 1 != 3
   - 1 < 3
   - 1 > 3
   - 3 <= 3
   - 3 >= 1

3. Logical operations:
   - true && false
   - true || false
   - !false

4. String operations:
    - "Hello, " + "world!"
    - "Catch " + 22
    - "A piece of string".length()
    - "ABCDEFG".charAt(3)
    - "MMXVIII".toLowerCase()
    - "Yellow Submarine".startsWith("Yellow")

5. Type conversions:
    - (double) 5
    - (int) 5.3
    - (int) 'a'
    - (char) 120
    - String.valueOf(1234)
    - Integer.parseInt("5678")
    - Double.parseDouble("3.14159")

# Types

Notice that when you evaluate an expression in the Code Pad, you see the resulting value and also its type. For example, `1 + 1` is `2` and its type is `int`, whereas `"1" + "1"` is `"11"` and its type is `String`.

Some of Java's most useful built-in types are `int`, `double`, `String`, `char` and `boolean`. For each expression below, write down its type **before** running it in the Code Pad to check your answers. Make a note of any expressions with unexpected result types.

- `1 + 1.5`
- `"a" + "b"`
- `"1" + 1.5`
- `'a' + 'b'`
- `'a' + 1`
- `false || !false`
- `"Hello".length()`

Java is a **statically typed** language, so before you run any code, the compiler works out the result type for each expression and checks for type errors. A type error occurs when you try to do something with the wrong type of value — try the following in the Code Pad:

- `1 && 2`
- `"a" - "b"`
- `true + false`
- `(1 + 2).length()`

These would all fail in Python too, but Python actually does `1 + 2` to get `3` before realising there is a type error. In contrast, Java can tell in advance that this will be a type error, so it doesn't run at all.

# Variables

The statement int x; declares a variable named x which holds a value of type int. In Java, you have to declare a variable before using it, and you have to say what type of value it will hold. (Note that you have to write a semicolon, ; at the end of a statement.)

```
int x;
x = 5;
```

For convenience, you can declare a variable and give it an initial value in the same statement:

```
int y = 8;
```

The int is needed to declare the variable, but this only happens once per variable. After that, you use the variable by writing just its name, like x instead of int x — Java already knows that x is of type int, so you don't need to tell it again. (It's a syntax error if you do.)

```
x + y

int x + int y
        Error: ';' expected
```

Just like in Python, you can assign a new value to x:

```
x = 11;

x += 23;
```

Unlike Python, you can't give x a new value of a different type:

```
x = 6.28;
        Error: incompatible types: possible lossy conversion from double to int
x = "Hello";
        Error: incompatible types: java.lang.String cannot be converted to int
```

Try declaring some more variables:

```
double pi = 3.14159;
char letter = 's';
String name = "Ada Lovelace";
```

**Work out the correct types and declare these variables:**

- A variable named `age` with the value `19`.
- A variable named `grade` with the value `'A'`.
- A variable named `gigaWatts` with the value `1.21`.
- A variable named `isBlue` with the value `true`.
- A variable named `phoneNumber` with the value `"555-1234"`.