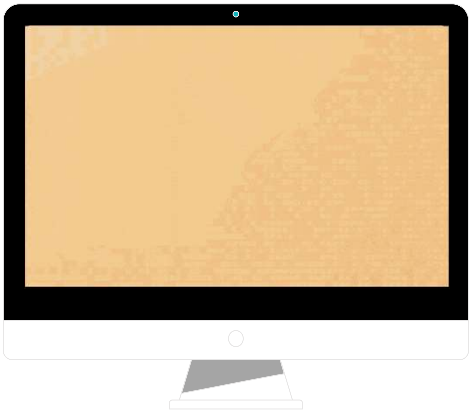
Recall: Intro to Java

- What is the difference between primitive and non-primitive data types and how can we tell the difference when they are declared?
- What is the shortcut to bring up the autocomplete suggestions?
- What do the ++ and -- operations on a numerical value do?



1

09. Java Fundamentals

What is control flow?

How can we create while, do-while, for and enhanced for loops?

How can we accept user input?

How do we cast?

What are classes and class methods? How do we pass to and return from a method?

01
Front-End
Development

02
Back-End
Development

03
BCS
Certification

2

Task Icons

Trainer-led lecture

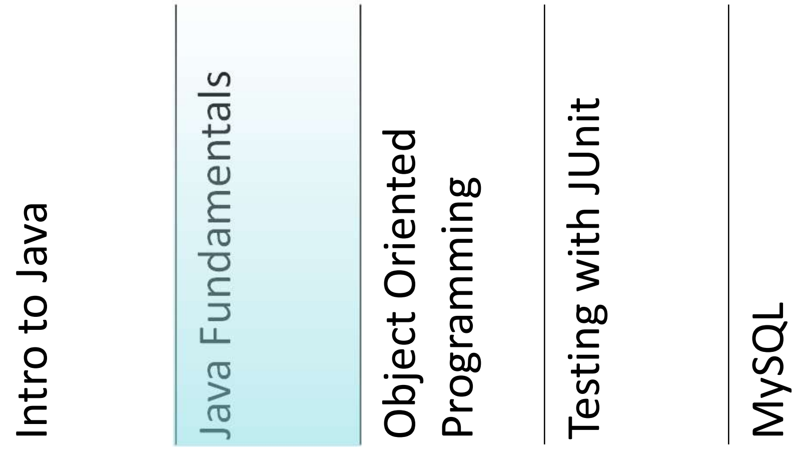
Code-along

Links to Certification

Independent task

Reflection

Learner Journey
Unit 2: Back-End Development



Intro to Java

Java Fundamentals

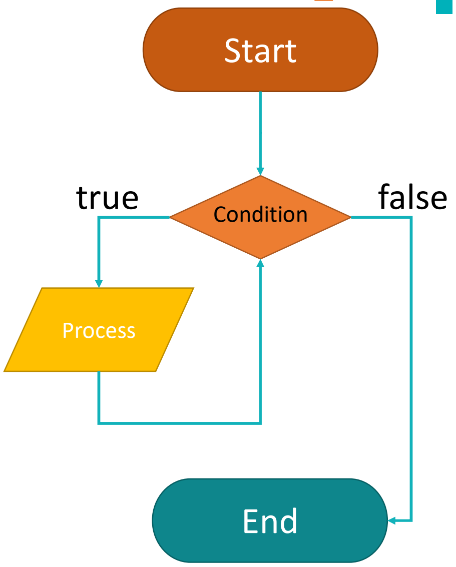
Object Oriented Programming

Testing with JUnit

MySQL

Loops

- Loops are a way of iterating the same piece of code many times.
- There are many reasons for this; Perhaps you need to go through a list of usernames and compare them to one a user is trying to register with.
- They allow us to reduce file size and keeps code legible when we need to execute the same thing many times.

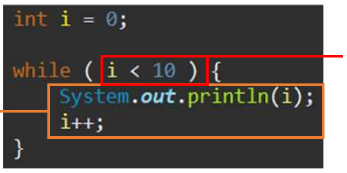


5

While Loop

- Lets start with a ‘while loop’. This loop will execute the block of code as long as an expression is true.

Code to be executed



```
int i = 0;

while ( i < 10 ) {
    System.out.println(i);
    i++;
}
```

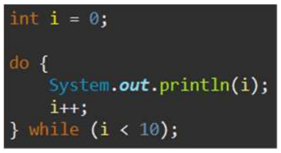
Expression that must be true to execute code block

- This code will iterate 10 times, increasing ‘i’ on every iteration and printing its value. What will be the last printed value of ‘i’?

6

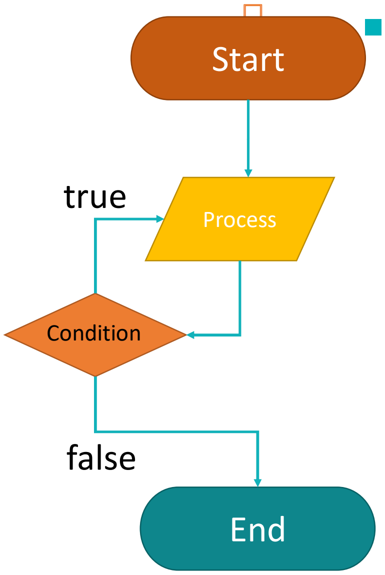
Do-While Loops

- Just like the first while loop, this loop will iterate 10 times and print the value of ‘i’ to the console.



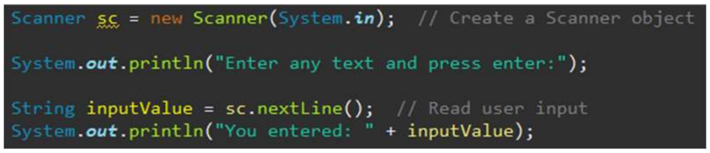
```
int i = 0;

do {
    System.out.println(i);
    i++;
} while (i < 10);
```

- The difference here is that the code block will always execute at least once before the condition is checked. This means that even if ‘i’ was set to 10 or more, it would still print the value of ‘i’.



User Input: The Scanner Class

- The Scanner class is used to create an object that allows the user to input information into the program.

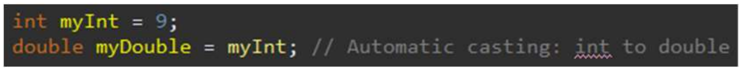


```
Scanner sc = new Scanner(System.in);  // Create a Scanner object

System.out.println("Enter any text and press enter:");

String inputValue = sc.nextLine();  // Read user input
System.out.println("You entered: " + inputValue);
```

- The ‘next()’ or ‘nextLine()’ methods are used to get that information, which must be stored into a String variable.

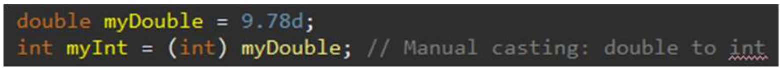
Casting

- Changing a value into another data type must be done with casting.
- We can do this by simply allowing Java to handle it with automatic casting.



```
int myInt = 9;
double myDouble = myInt; // Automatic casting: int to double
```

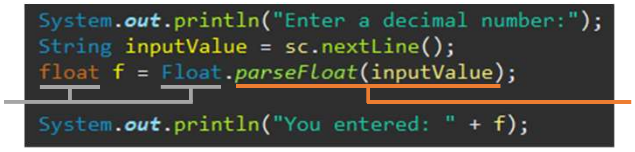
- Sometimes we need to be more explicit than that and tell Java which data type to cast to.



```
double myDouble = 9.78d;
int myInt = (int) myDouble; // Manual casting: double to int
```

Casting: Scanner Input

- With the example of the Scanner's next() and nextLine() methods, these values must be stored as a String.
- When we want to get numerical information from the user, we must cast the variable into another data type using the data-types class definition's method:



```
System.out.println("Enter a decimal number:");
String inputValue = sc.nextLine();
float f = Float.parseFloat(inputValue);
System.out.println("You entered: " + f);
```

Data type/ class name

Method name and value being cast

Task 1: Create Some Loops

- Create a loop that iterates 20 times and prints the sentence “The value of i is ” and the value of ‘i’:
  - The value of i is 0
  - The value of i is 1

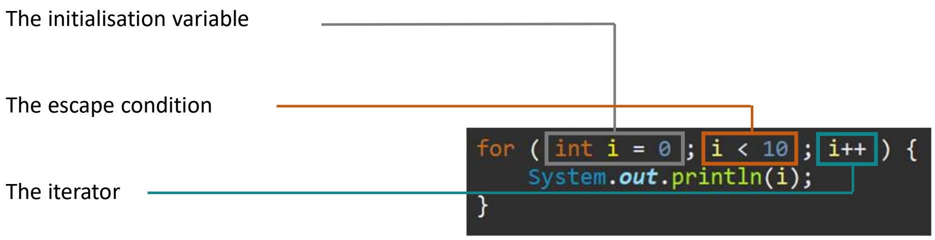
- Create a loop with the same output as the first loop, but asks the user for a number and iterate that number of times.

Stretch Challenge: Create a loop that will print the number of times the loop has been executed, and only exit when the user inputs the word “stop”.

Simple For Loops

- A basic ‘for loop’ will allow us to create the conditions in how our loop operates within the loop declaration.

- Within the for loop we have three distinct sections:



The initialisation variable

The escape condition

The iterator

```
for ( int i = 0 ; i < 10 ; i++ ) {
    System.out.println(i);
}
```

Simple For Loops

- With this structure we have an incredible amount of control within our loops. We don’t need to stick to this strict format, either.
- For our escape condition, we can use anything we want here – **it doesn’t have to be the variable iterator you created**! As long as the expression results to a Boolean, we can use anything we need.
- For our iterator, we can use things like:
  - i--
  - i*=2
  - Or even a call to another method!

13

Arrays and ArrayLists

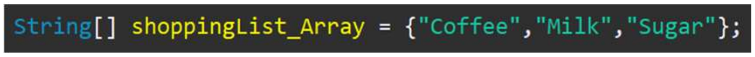
- Both Arrays and ArrayLists are collections of data inside a single object variable.
- The two types are closely linked but have differing properties:
  - Arrays are fixed in size – Arrays can store any data type
  - ArrayList are variable in length – ArrayList can only store non-primitive data

- Based on the above information, we need to make our decisions carefully.

14

Arrays

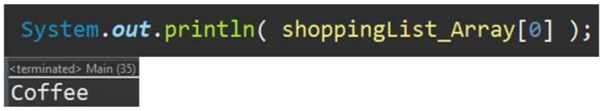
- Arrays are declared by using the non-primitive data type, accompanied by square brackets, before the name of the variable.
- The other side of the assignment operator, we create a collection of data, separated by a comma, encased in curly braces:



```
String[] shoppingList_Array = {"Coffee","Milk","Sugar"};
```

Arrays

- We then access the data within that array by referencing the variable name and stating **which index** of the array we wish to inspect.
- Beware, that the indexing **starts at zero**:

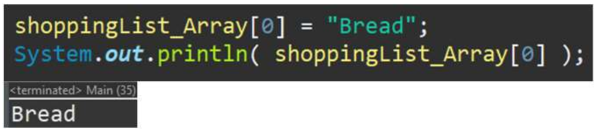


```
System.out.println( shoppingList_Array[0] );
```

```
<terminated> Main (35)
Coffee
```

Arrays

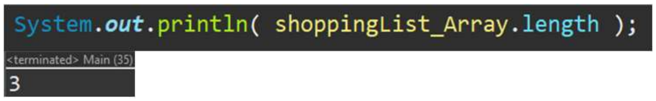
- Although we cannot change the size of the array, we **can** change the contents of the data inside of the array at a given index:



```
shoppingList_Array[0] = "Bread";
System.out.println( shoppingList_Array[0] );
```

```
<terminated> Main (35)
Bread
```

- We can even see the size of the array using the .length value:



```
System.out.println( shoppingList_Array.length );
```

```
<terminated> Main (35)
3
```

Task 2: Shopping List

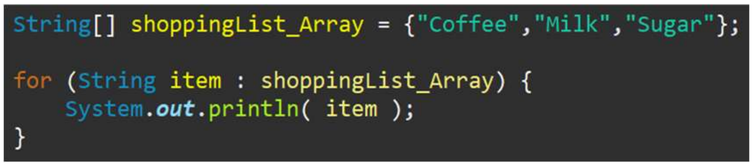
- Create a data collection of items you might need when shopping.
- This collection of shopping should be stored inside of an **Array**.
- Iterate over the array and print each item to the console by using a '**for loop**' and using the .length property of the shopping list array

Stretch Challenge: Take a look at the .foreach() method – can you replicate the above using a **stream**? You may need to do some research on this one!

Enhanced For Loop

- The ‘enhanced for loop’ allows us to use a new structure within our loops that will iterate the exact amount of times for the size/length of the data collection we are inspecting.

- It works by declaring a temporary variable to store each item in and therefore needs to be declared the same data type as the values within the collection:



```
String[] shoppingList_Array = {"Coffee","Milk","Sugar"};

for (String item : shoppingList_Array) {
    System.out.println( item );
}
```

Thoughts?

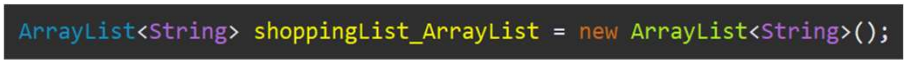
- What do you think of the enhanced for-loop?

- What benefits can you think of when using it?

- Where can we **not** use it?

ArrayLists

- ArrayLists are slightly trickier to set up but can be used with primitive data types, and so have their place in Java.
- We need to utilise a lot of new keywords you've not seen before – by the end of next week's session, these terms will be overly familiar to you.
- For now, we will make a simple ArrayList of type String.

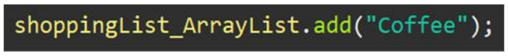


```
ArrayList<String> shoppingList_ArrayList = new ArrayList<String>();
```

21

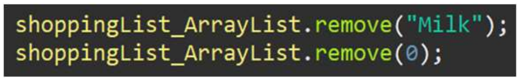
ArrayLists

- We can add and remove data from an ArrayList fairly simply.
- We use the .add() method to add an item to the collection:



```
shoppingList_ArrayList.add("Coffee");
```

- And we use .remove() method to remove an item from the collection – we can even specify either the index or the value specifically:



```
shoppingList_ArrayList.remove("Milk");
shoppingList_ArrayList.remove(0);
```

22

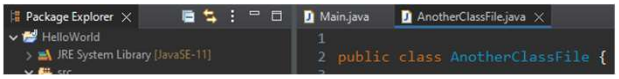
Task 3: Create an ArrayList of Places to visit

- Create a data collection of places you might want to visit in your lifetime.
- This collection of places should be stored inside of an **ArrayList**.
- Iterate over the array and print each item to the console by using a '**enhanced for loop**'.

Stretch Challenge: Create a 2D array of countries, and in each country, an array of places you would like to visit. Can you iterate over the entire 2D array? You will need to use a loop within a loop

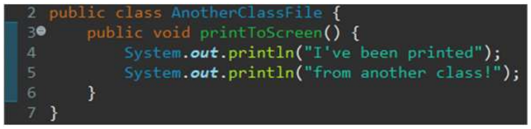
Classes

- At the top of your coding environment, you will see the file name 'Main.java'. This is in fact the name of the 'class' we are writing in.
- I am going to create a new class called 'AnotherClassFile.java',.
- This time, however, **I will not** tick the check box for 'public static void main(String[] args'



```

public class AnotherClassFile {
```

- Within this new class file, I will write a simple method called 'printToScreen':

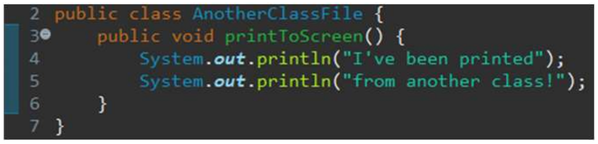


```
public class AnotherClassFile {
    public void printToScreen() {
        System.out.println("I've been printed");
        System.out.println("from another class!");
    }
}
```

- This method can now be accessed from another class within the same project.

Class Methods

- Within this new class file, I will write a simple method called ‘printToScreen’:

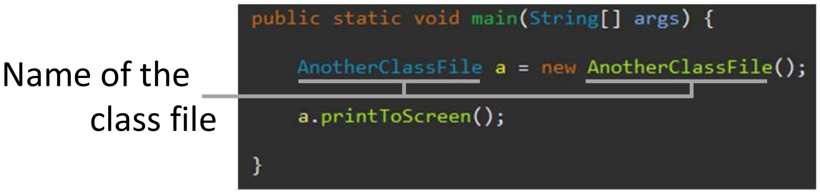


```
public class AnotherClassFile {
    public void printToScreen() {
        System.out.println("I've been printed");
        System.out.println("from another class!");
    }
}
```

- This method can now be accessed from another class within the same project.

Class Methods

- Back in my Main class, I can now access the method I just created:



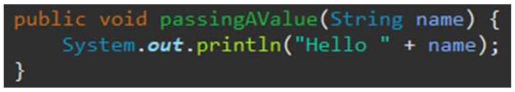
```
public static void main(String[] args) {

    AnotherClassFile a = new AnotherClassFile();

    a.printToScreen();

}
```

Name of the class file

- The code can be a little confusing to look at, but we will cover what is going on in this line shortly.

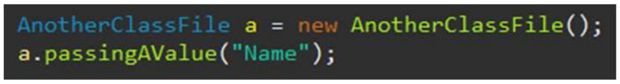
Class Methods – Passing a Value

- We can pass a value to a method by declaring the data types and the name of the data type we are going to use in the parentheses.



```
public void passingAValue(String name) {
    System.out.println("Hello " + name);
}
```

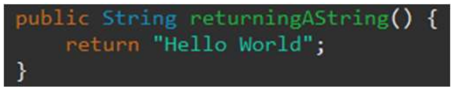
- Which means we need to pass a value, in this case a String, to the method when we call it.



```
AnotherClassFile a = new AnotherClassFile();
a.passingAValue("Name");
```

Class Methods – Getting a Value

- We can also get values back from a class method too, by using the return key word.

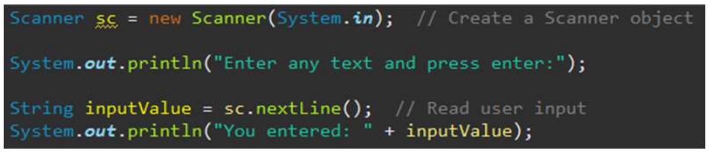


```
public String returningAString() {
    return "Hello World";
}
```

- A key thing to note here is that we no longer use the word ‘void’ in the method declaration, but instead, we use the ‘String’ key word. ‘void’ was previously signifying that there isn’t a returned value. Now, we’re saying a String is being returned.

User Input – The Scanner Class

- The Scanner class is used to create an object that allows the user to input information into the program.



```
Scanner sc = new Scanner(System.in);  // Create a Scanner object

System.out.println("Enter any text and press enter:");

String inputValue = sc.nextLine();  // Read user input
System.out.println("You entered: " + inputValue);
```

- The ‘next()’ or ‘nextLine()’ methods are used to get that information, which must be stored into a String variable.

29

Task 3: Create a calculator with classes

- Create a new class called “Calculator.java” and within this class, create methods for each of the operators (+ - / *) passing in the two numerical values (described below).

- Create a new Calculator object in your Main class. Then, from Main, given the value of 3 variables taken from the user (with Scanner):
  - int firstNumber
  - char operator
  - int secondNumber

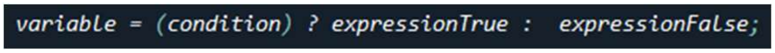
- Print to screen the correct value.

Stretch Challenge: Accept user input for modulo and to-the-power-of too and print the correct result.

30

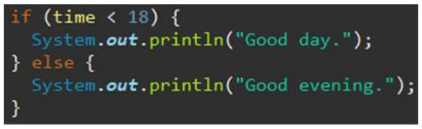
Ternary Operator
A New Kind of If Statement

- Ternary operators are used to create single line if-statements, reducing code size.

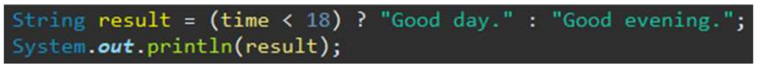


```
variable = (condition) ? expressionTrue :  expressionFalse;
```

- The question mark symbol is used as the separator between the condition and the results, and the colon is used to separate results.



```
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

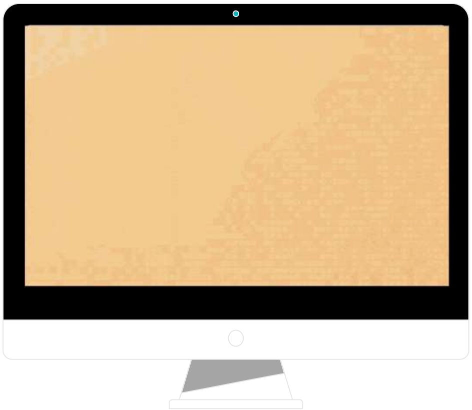


```
String result = (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);
```

31

Review: Question

- What is the purpose of the ‘return’ key word and how does the ‘void’ keyword affect the return statement?



32