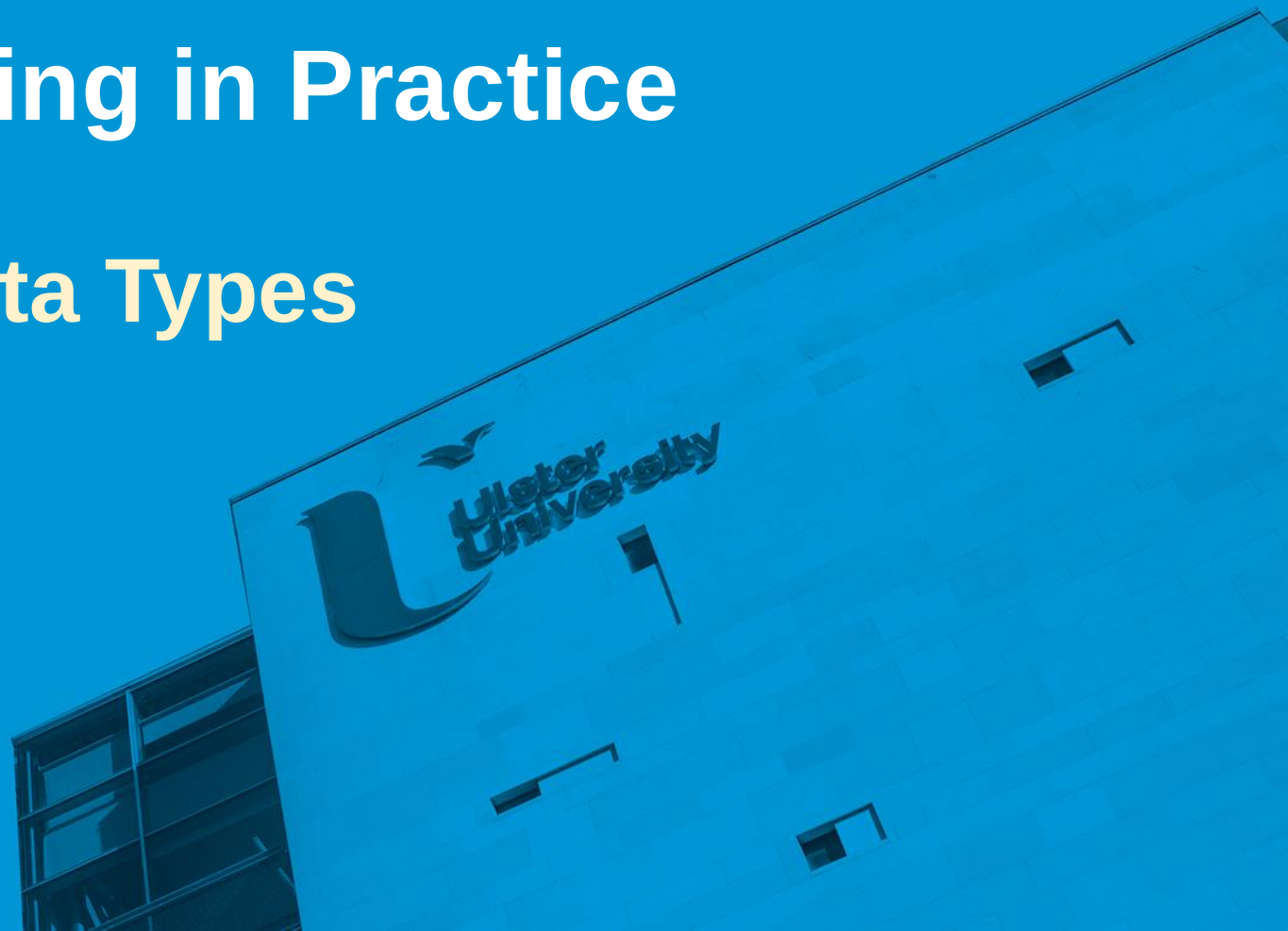
Ulster University

COM410 Programming in Practice

A2.1 Variables and Data Types



Java Data Types

- In languages that you have met earlier on the course, you were not (necessarily) required to allocate a specific type to a variable.
- Java is a **strongly-typed** language
  - A variable must be declared before it can be used
  - A variable cannot be declared without specifying the range of values it can hold
  - Once declared, the type of a variable cannot be changed

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

    int age = 25;
    double pi = 3.14;
    double accuracy = 99.5, percentage = 0.5;
    boolean javaIsFun = true, isJavaFun;

}
```

- Can declare multiple variables of the same type in a single statement
- Initial values are optional
- Note **camelCase** format for variable names (by convention)

Java Data Types

- Java provides 8 primitive variable types

Primitive type	Description
`byte`	8-bit integer
`short`	16-bit integer
`int`	32-bit integer
`long`	64-bit integer
`float`	32-bit floating point
`double`	64-bit floating point
`boolean`	`true` or `false`
`char`	Single character value

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

    byte myVariable1 = 100;
    short myVariable2 = 2000;
    int myVariable3 = 32766;
    long vaVariable4 = 655360;
    float myVariable5 = 3.14f;
    double myVariable6 = 1234.567;
    boolean myVariable7 = true;
    char myVariable8 = 'c';

}
```

Java Strings

- A character string is not a primitive type in Java, but is an object that supports a range of methods providing various manipulations such as
  - Return the length
  - Case conversion
  - Finding and extracting substrings

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

    String myModule = "COM410 Programming in Practice";

    System.out.println("Length is " + myModule.length() + " characters");
    System.out.println("As uppercase : " + myModule.toUpperCase() );
    System.out.println("As lowercase : " + myModule.toLowerCase() );
    System.out.println("\"in\" first appears at position " + myModule.indexOf("in"));
    System.out.println("\"in\" next appears at position " + myModule.indexOf( str: "in",  fromIndex: 16));
    System.out.println("Substring starting at position 7 is " + myModule.substring(7));
    System.out.println("Substring from position 7 to 18 is " + myModule.substring(7, 18));

}
```

Java Strings

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
C	O	M	4	1	0		P	r	o	g	r	a	m	m	i	n	g		i	n		P	r	a	c	t	i	c	e

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

    String myModule = "COM410 Programming in Practice";

    System.out.println("Length is " + myModule.length() + " characters");
    System.out.println("As uppercase : " + myModule.toUpperCase() );
    System.out.println("As lowercase : " + myModule.toLowerCase() );
    System.out.println("\"in\" first appears at position " + myModule.indexOf("in"));
    System.out.println("\"in\" next appears at position " + myModule.indexOf( str: "in",  fromIndex: 16));
    System.out.println("Substring starting at position 7 is " + myModule.substring(7));
    System.out.println("Substring from position 7 to 18 is " + myModule.substring(7, 18));

}
```

```
Length is 30 characters
As uppercase : COM410 PROGRAMMING IN PRACTICE
As lowercase : com410 programming in practice
"in" first appears at position 15
"in" next appears at position 19
Substring starting at position 7 is Programming in Practice
Substring from position 7 to 18 is Programming
```

Java Strings

- Java strings can be concatenated by the **+** operator
    - But we need to be careful when concatenating strings and numbers

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

    String myModuleCode = "COM410",  myModuleName = "Programming in Practice";
    System.out.println(myModuleCode + " " + myModuleName);

    String number1 = "10", number2 = "20";
    System.out.println(number1 + number2);

    int number3 = 30;
    System.out.println(number1 + number3);

}
```

```
COM410 Programming in Practice
1020
1030
```

Java Arrays

- Java Arrays allow us to store multiple values (of the same type) in a single named variable

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

    int[] myNumbers = { 10, 20, 30, 40, 50 };
    int[] yourNumbers = new int[10];
    System.out.println(myNumbers[2]);

    yourNumbers[2] = 100;
    System.out.println(yourNumbers[2]);
    System.out.println(yourNumbers[3]);

    String[] myStrings = new String[3];
    System.out.println(myStrings[1]);
}
```

- Defined and manipulated using the `[]` notation
- We can reserve array space even if we don't know the values yet
- Be careful when accessing uninitialized array elements

```
30
100
0
null
```

myNumbers

0	1	2	3	4
10	20	30	40	50

yourNumbers

0	1	2	3	4	5	6	7	8	9
0	0	100	0	0	0	0	0	0	0

Multidimensional Arrays

- Useful for representing data as a table
  - Each element of the array is itself an array
  - Up to 255 dimensions are possible (but would be very difficult to visualize)

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

    int[][] myNumbers = { { 10, 20 }, { 30, 40 }, { 50, 60 } };
    int[][] yourNumbers = new int[10][4];
    System.out.println(myNumbers[1][0]);

    yourNumbers[7][2] = 100;
    System.out.println(yourNumbers[7][2]);
    System.out.println(yourNumbers[8][2]);
}
```

	0	1
0	10	20
1	30	40
2	50	60

myNumbers

```
30
100
0
```

Random Values

- Uses the **`Random`** object from the **`java.util`** library

```
import java.util.Random;

public class Demo {

    public static void main(String[] args) {

        Random random = new Random();
        int randomDigit1 = random.nextInt( bound: 10);
        int randomDigit2 = random.nextInt( bound: 10);
        int randomDigit3 = random.nextInt( bound: 10);
        System.out.printf("My randomly selected digits are %d %d %d \n",
                          randomDigit1, randomDigit2, randomDigit3);
    }
}
```

```
My randomly selected digits are 2 7 4
```

- **`nextInt()`** method returns a random integer less than the value specified as a parameter
- Corresponding methods for **`long`**, **`double`**, **`byte`**, **`float`**, **`boolean`**, etc.

Scenario

- Create a new Java project called `TownChallenge` containing a class file *TownChallenge.java* with the following functionality in the `main()` method.
    i. Define a `String` array called `towns` to hold the names of 8 towns of your choice.
    ii. Generate and display a set of sporting results such that `towns[0]` plays `towns[1]`, `towns[2]` plays `towns[3]`, and so on. The number of points won by each town should be a random integer in the range 0-9. For example, the display of results might be in the form…

```
Banbridge 7 Ballymena 4
Belfast 3 Newry 3
Coleraine 9 Enniskillen 0
Lurgan 5 Portadown 5
```