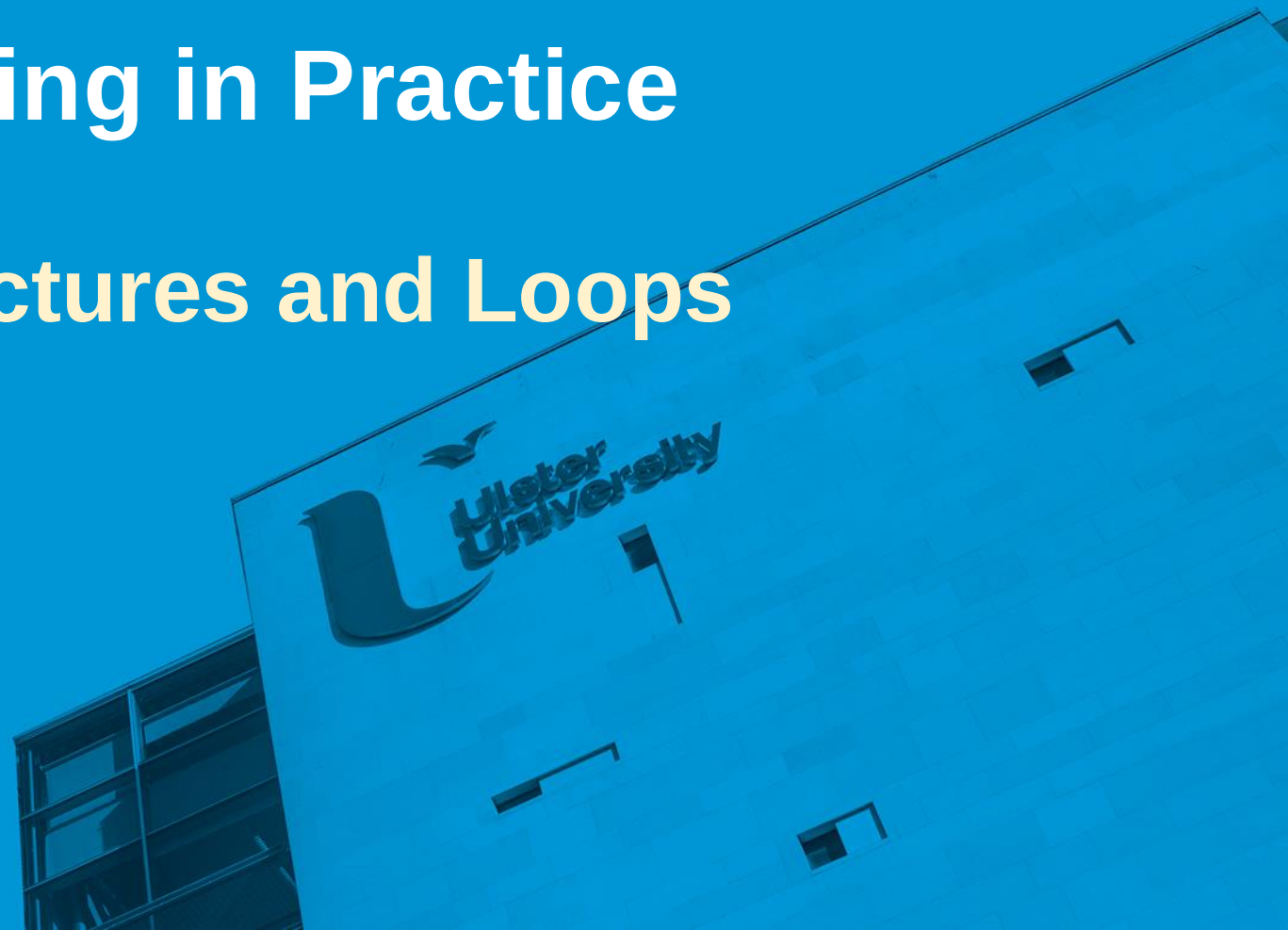
Ulster University

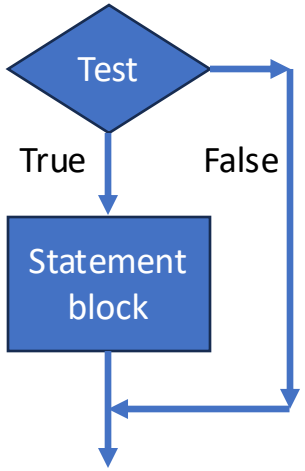
COM410 Programming in Practice

A2.2 Conditional Structures and Loops

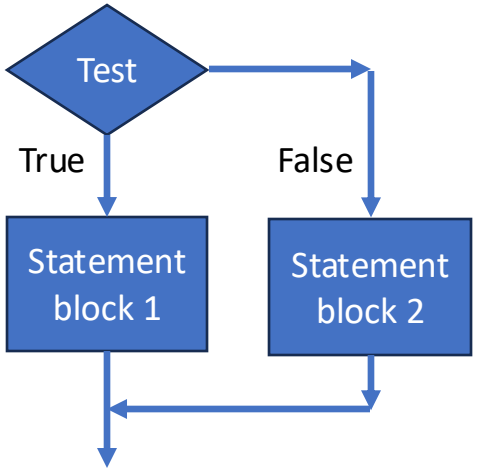


Conditional Structures

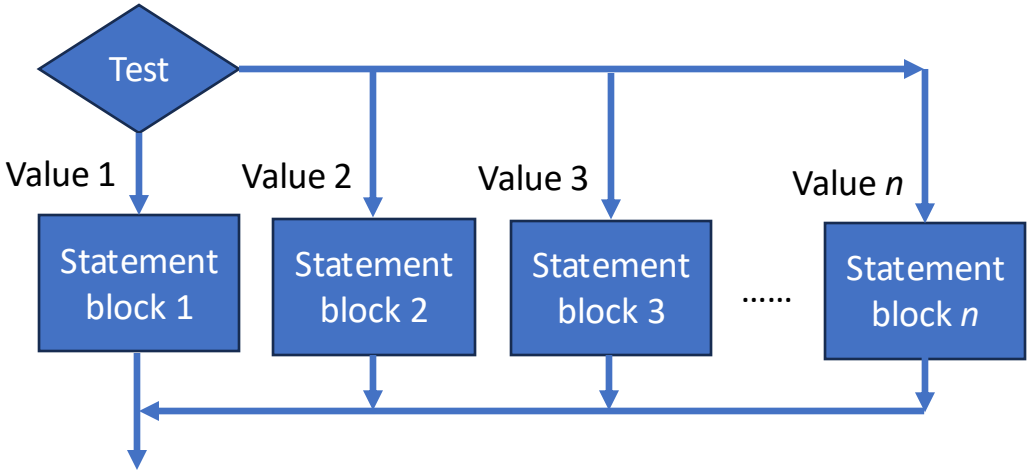
- Enabling a decision of whether a piece of code is executed, or the selection between alternative pieces of code to execute
    - 3 alternative conditional structures are available



`if` statement



`if`…**`else`** statement



`switch` statement

`if` Statement

- A statement block is optionally run depending on the result of a logic condition

```java
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        System.out.print("Please enter a number > ");
        int value = keyboard.nextInt();

        if (value % 2 == 0) {
            System.out.printf("%d is an even number.\n", value);
        }
    }
}
```

```java
if (condition) {

        // statements if true

}
```

```
Please enter a number > 54
54 is an even number.

Process finished with exit code 0
```

```
Please enter a number > 17

Process finished with exit code 0
```

`if...else` Statement

- One statement block from a choice is run depending on the result of a logic condition

```java
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {

        Scanner keyboard = new Scanner(System.in);
        System.out.print("Please enter a number > ");
        int value = keyboard.nextInt();

        if (value % 2 == 0) {
            System.out.printf("%d is an even number.\n", value);
        } else {
            System.out.printf("%d is an odd number.\n", value);
        }
    }
}
```

```
if (condition) {
        // statements if true
} else {
        // statements if false
}
```

```
Please enter a number > 54
54 is an even number.
```

```
Please enter a number > 17
17 is an odd number.
```

`switch` Statement

- One statement block from a (larger) choice is run depending on the value of some variable

```
public class Demo {

    public static void main(String[] args) {
        int season = 10;

        switch (season) {
            case 0: System.out.println("Season is Spring");
                    break;
            case 1: System.out.println("Season is Summer");
                    break;
            case 2: System.out.println("Season is Autumn");
                    break;
            case 3: System.out.println("Season is Winter");
                    break;
            default: System.out.println("Invalid season value");
        }
    }
}
```

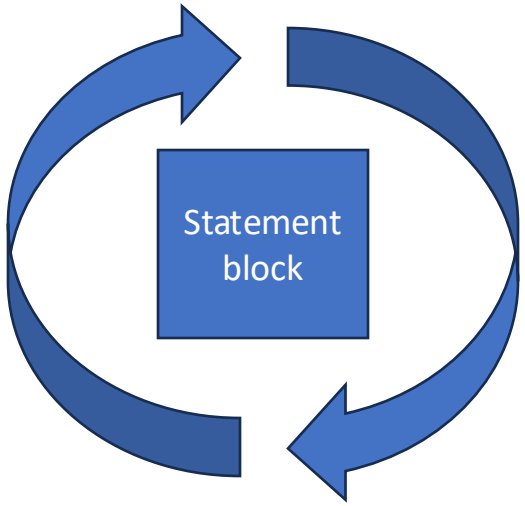
```
switch (expression) {
  case x:  // code for x
           break;
  case y:  // code for y
           break;
  default: // default code
}
```

```
Invalid season value
```

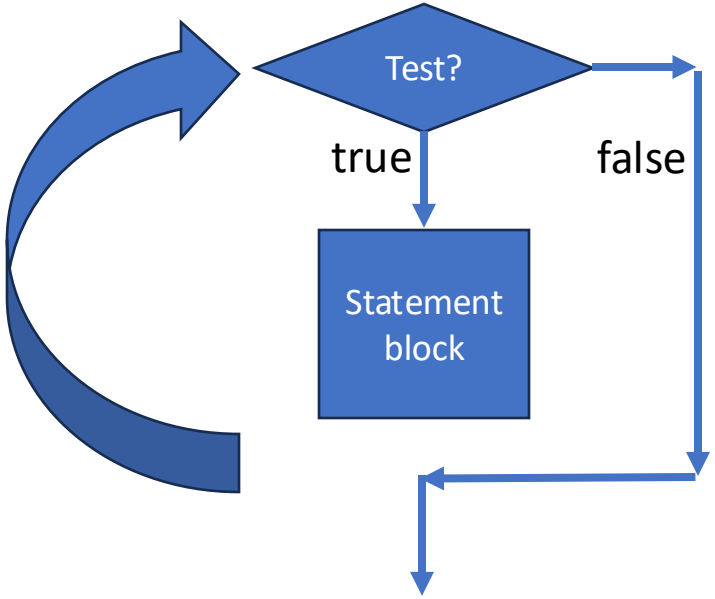
- **`break`** is needed to prevent further execution
- **`default`** is optional

Loop Structures

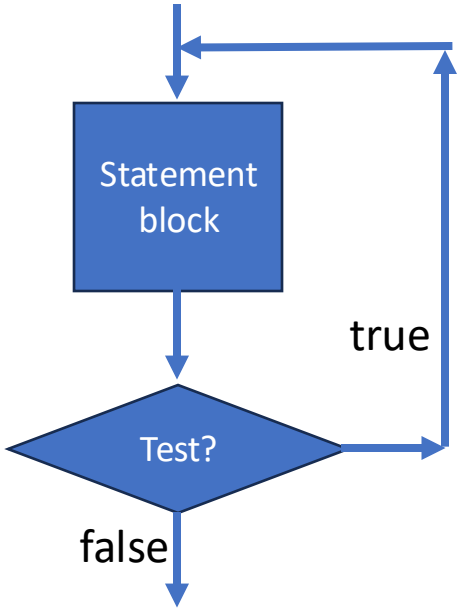
- Enabling repetition of code until a certain limit is reached or for as long as a certain condition is true
    - 3 alternative loop structures are available



`For` loop



`While` loop



`Do…While` loop

`for` Loop

- Repeat a code block for a fixed number of times

```
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        double marks = 0;

        for (int i = 1; i <= 6; i++) {
            System.out.printf("Enter mark for module %d > ", i);
            marks = marks + keyboard.nextInt();
        }
        System.out.printf("Average mark is %.1f%%", marks / 6);
    }
}
```

Three parts to the **`for`** statement

1. Create and initialise loop counter
2. Test to continue in loop
3. Update counter at end of loop

```
Enter mark for module 1 > 67
Enter mark for module 2 > 53
Enter mark for module 3 > 69
Enter mark for module 4 > 42
Enter mark for module 5 > 70
Enter mark for module 6 > 63
Average mark is 60.7%
```

`while` Loop

- Enabling repetition of code while a condition is met

```
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int mark, total = 0;

        System.out.print("Enter score (or a negative number to exit) > ");
        mark = keyboard.nextInt();
        while (mark > 0) {
            total = total + mark;
            System.out.print("Enter score (or a negative number to exit) > ");
            mark = keyboard.nextInt();
        }
        System.out.printf("Total score is %d", total);
    }
}
```

```
while (condition) {

    // code to be repeated
}
```

- Sometimes needs two data entry points
- Use if the loop code may not be needed

```
Enter score (or a negative number to exit) > 180
Enter score (or a negative number to exit) > 180
Enter score (or a negative number to exit) > 141
Enter score (or a negative number to exit) > -1
Total score is 501
```

do...while Loop

- Enabling repetition of code while a condition is met

```java
import java.util.Scanner;

public class Demo {

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        int mark = 0, total = 0;

        do {
            total = total + mark;
            System.out.print("Enter score (or a negative number to exit) > ");
            mark = keyboard.nextInt();
        } while (mark > 0);

        System.out.printf("Total score is %d", total);
    }
}
```

```java
do {

    // code to be repeated

} while (condition)
```

- Use if the loop code must not be avoided

```
Enter score (or a negative number to exit) > 180
Enter score (or a negative number to exit) > 180
Enter score (or a negative number to exit) > 141
Enter score (or a negative number to exit) > -1
Total score is 501
```

Arrays and Loops

- **for-each** is a special form of the `for` loop used to iterate over an array
  - The loop variable takes on the value of each array element in turn

```
public static void main(String[] args) {
    String[] fruits = { "Apples", "Strawberries", "Grapes", "Cherries" };

    for (String fruit : fruits) {
        System.out.println(fruit);
    }
}
```

```
Apples
Strawberries
Grapes
Cherries
```

```
for (variable : array name) {

   // code to be repeated where variable
   // is each array element in turn

}
```

Scenario

- In your `TownChallenge` project, add additional code to *TownChallenge.java* to achieve the following:
    i. Modify the code for generation of results so that it uses a loop structure so that the main body of the loop only generates a single result. The loop should run as many times as is required to generate the complete set of fixtures.
    ii. Keep track of the number of home wins, away wins and draws in the set of fixtures.
    iii. Add a line to the output after each set of results in the form (e.g.)
    `Homes 2, Draws 1, Aways 1`
    according to the results generated.