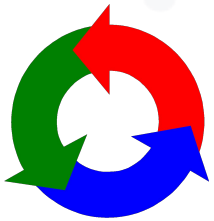
Flow Controls

Classifications

Sequential

Selection

Iteration



Transfer

Executes once* without any conditional checks

Selection statements executes only once based on condition

Looping statement executes repeatedly several numbers of time till exit condition is met

Sequential statements

- Statements can be grouped using braces {} to form a compound statement known as block statements. Types are:
  - Static block
  - Instance block

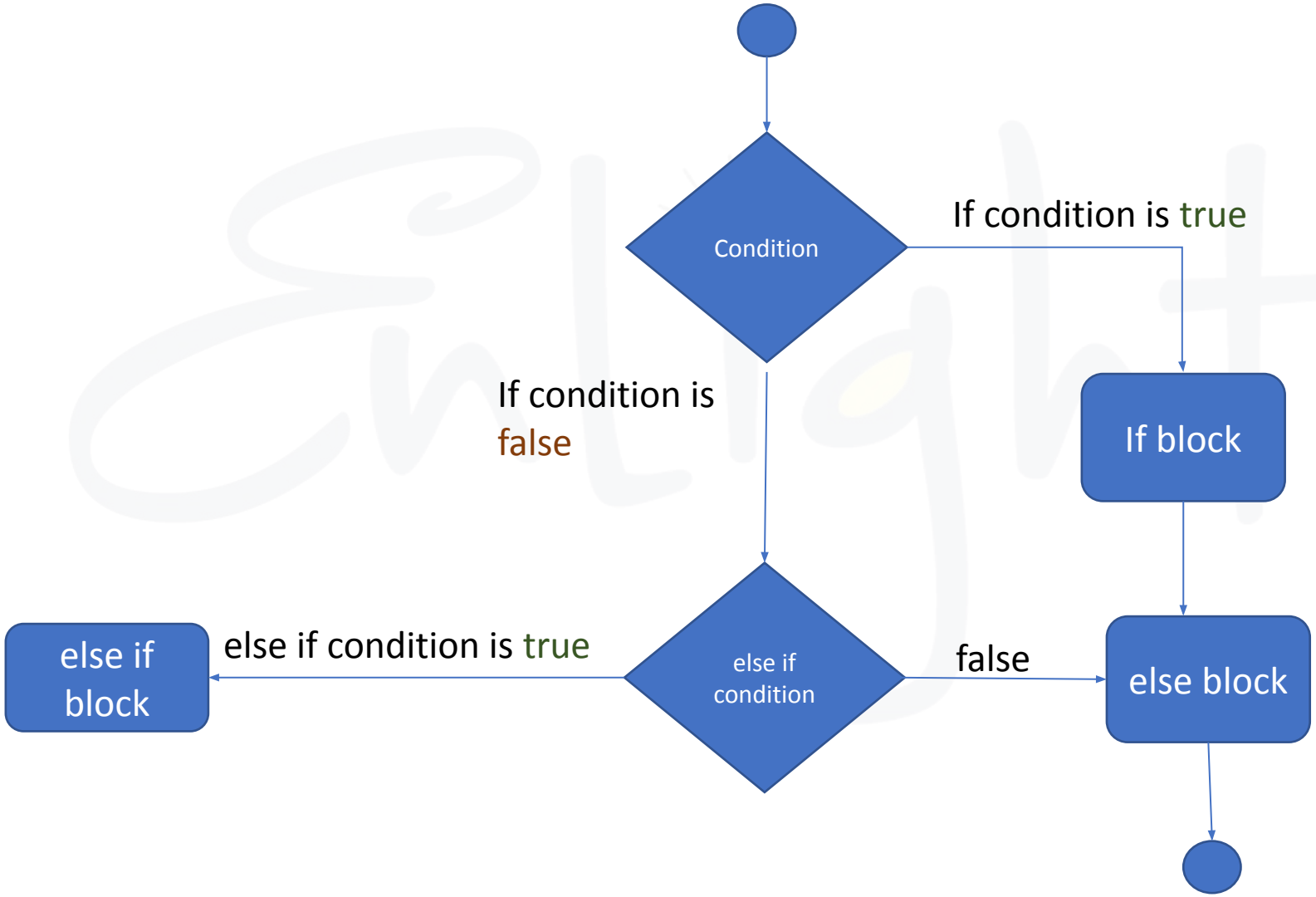
```
class BlockStatementExamples {

    /**
     * Static block which will be executed only once
     * during class loading
     */
    static {

        //code statements
        System.out.println("In the static block");
    }

    /**
     * Instance block which will be executed each
     * time a new instance is created using "new" operator
     */
    {
        //code statements
        System.out.println("In the instance block");
    }

   public static void main(String arg[]) {
       BlockStatementExamples exm = new BlockStatementExamples();
       BlockStatementExamples exm2 = new BlockStatementExamples();
       BlockStatementExamples exm3 = new BlockStatementExamples();
   }
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=Cp1252
In the static block
In the instance block
In the instance block
In the instance block
```

Selection Statements

- Selection statements are also known as **decision making statement**
- Based on condition decision will be made whether to execute an block or not
- Types of selection Statements:
  - if
  - if-else
  - switch

Selection Statements: if, else if & else




```
if(true) {
    //if block execution
} else if (true) {
    //else if block execution
}else {
    //else block execution
}
```

Selection Statement: Switch statement

- Switch statement allows the value of a variable or expression to control the flow of a program execution via a multi-way branch
- Creates multiple branches in a simpler way than using the combination of if and else if statements
- Each branch can be ended with the break keyword else all case block after the matched case statement will be executed
- The value for a case must be same type as the variable in a switch
- switch statement work with byte, short, char and int primitive data type and also works with enumerated types and string

Selection Statements: Switch statement

```
int i = 5;
switch(i)
{
 case 2:
     System.out.println("Case 2 block executed");
 case 3:
     System.out.println("Case 3 block executed");
 case 5:
     System.out.println("Case 5 block executed");
 case 6:
     System.out.println("Case 6 block executed");
     break;
 case 7:
     System.out.println("Case 7 block executed");
 default:
     System.out.println("Default block executed");
}
```

```
<terminated> StaticInnerClass [Jav
Picked up JAVA_TOOL_OPT
Case 5 block executed
Case 6 block executed
```

Selection Statements: Summary…

How do you decide which selection statement use under what scenarios?

Iteration statements

- Iteration statements also known as looping statements execute one or more statement repeatedly a several number of times till exit condition is met
- Types of Iteration statements:
  - While loop
  - Do while loop
  - For loop (for each loop)

Iteration Statements: while & do while loop

while loop

```
int i = 0;
while(i < 5) {
    System.out.println("Executing while loop block, count: "+ i);
    i++;
}
```

```
<terminated> StaticInnerClass [Java Application] /Librar
Picked up JAVA_TOOL_OPTIONS: -Dfile.e
Executing while loop block, count: 0
Executing while loop block, count: 1
Executing while loop block, count: 2
Executing while loop block, count: 3
Executing while loop block, count: 4
```

do while loop

```
int i = 0;
do {
    System.out.println("Executing do while loop block, count: "+ i);
    i++;
}
while(i <= 0);
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.enc
Executing do while loop block, count: 0
```

Iteration Statements: for loop

Mostly used for counter-controlled loops, that is, when the number of iterations is known beforehand

for (<initialization>; <loop condition>; <increment expression>)

    <loop body>

```
for(int i = 0; i < 5; i++) {
    System.out.println("Executing for loop block, count " + i);
}
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile
Executing for loop block, count 0
Executing for loop block, count 1
Executing for loop block, count 2
Executing for loop block, count 3
Executing for loop block, count 4
```

Iteration Statements: for each loop

- From Java 1.5, for easy iteration of arrays/collections elements for each loop was introduced
- Also reduces the possibility of general programming errors in normal for loop
- Syntax:

```
for(<data type of elements in array/collection> variable name: <collection/array to iterate>)
{ ... }
```

Iteration Statements: Summary

When to use for, for-each, while and do-while statements?

Transfer Statements

- Types of Transfer statements:
- break
- continue
- return

Break

- Break statement forces immediate termination of a loop, bypassing the conditional expression and any remaining code in the body of the loop.
- When used inside a set of nested loops, will only break out of the innermost loop.

Transfer Statements: Break

```
for(int i = 0; i < 5; i++) {
    System.out.println("Executing outer for loop block, count " + i);
    for(int j = 0; j < 5; j++) {
        System.out.println("Executing Inner for loop block, count " + j);
        //Condition to break from inside loop
        if(j == 2) {
            System.out.println("Breaking out of inner loop");
            break;
        }
    }
    //Condition to break from outside loop
    if(!(i < 2)) {
        System.out.println("Breaking out of outer loop");
        break;
    }
}
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.encodi
Executing outer for loop block, count 0
Executing Inner for loop block, count 0
Executing Inner for loop block, count 1
Executing Inner for loop block, count 2
Breaking out of inner loop
Executing outer for loop block, count 1
Executing Inner for loop block, count 0
Executing Inner for loop block, count 1
Executing Inner for loop block, count 2
Breaking out of inner loop
Executing outer for loop block, count 2
Executing Inner for loop block, count 0
Executing Inner for loop block, count 1
Executing Inner for loop block, count 2
Breaking out of inner loop
Breaking out of outer loop
```

Transfer Statements: Continue

```java
for (int j = 0; j < 5; j++) {
    System.out.println("Executing Inner for loop block, count " + j);
    // Condition to break from inside loop
    if (j % 2 != 0) {
        System.out
                .println("Rest of this loop logic skipped, continue with next iteration");
        continue;
    }
    System.out.println("Printing even number " + j);
}
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=Cp1252
Executing Inner for loop block, count 0
Printing even number 0
Executing Inner for loop block, count 1
Rest of this loop logic skipped, continue with next iteration
Executing Inner for loop block, count 2
Printing even number 2
Executing Inner for loop block, count 3
Rest of this loop logic skipped, continue with next iteration
Executing Inner for loop block, count 4
Printing even number 4
```

Transfer Statements: Return

```
for (int i = 0; i < 7; i++) {
    System.out.println("Executing outer for loop block, count " + i);
    for (int j = 0; j < 5; j++) {
        System.out
                .println("Executing Inner for loop block, count " + j);
        // Condition to break from inside loop
        if (j >= 3) {
            System.out
                    .println("Rest of this method logic will be skipped");
            return;
        }
    }
    System.out.println("Rest of the outside loop execution");
}
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.encodi
Executing outer for loop block, count 0
Executing Inner for loop block, count 0
Executing Inner for loop block, count 1
Executing Inner for loop block, count 2
Executing Inner for loop block, count 3
Rest of this method logic will be skipped
```

Exercise

1

```
if (num = 0)
    System.out.println("first string");
else
    System.out.println("second string");
```

2

```
int b = 1; while (true) { b++; break; }
System.out.println("value of b: " + b);
```

3

```
int aNumber = 5;
if (aNumber == 0)
    System.out.println("first string");
else
    System.out.println("second string");
System.out.println("third string");
```

4

```
switch("third") {
    case "first":
        System.out.println("printing first");
    case "second":
        System.out.println("printing second");
    case "third":
        System.out.println("printing third");
    case "fourth":
        System.out.println("printing fourth");
        break;
    case "fifth":
        System.out.println("printing fifth");
    default:
        System.out.println("printing default");
}
```