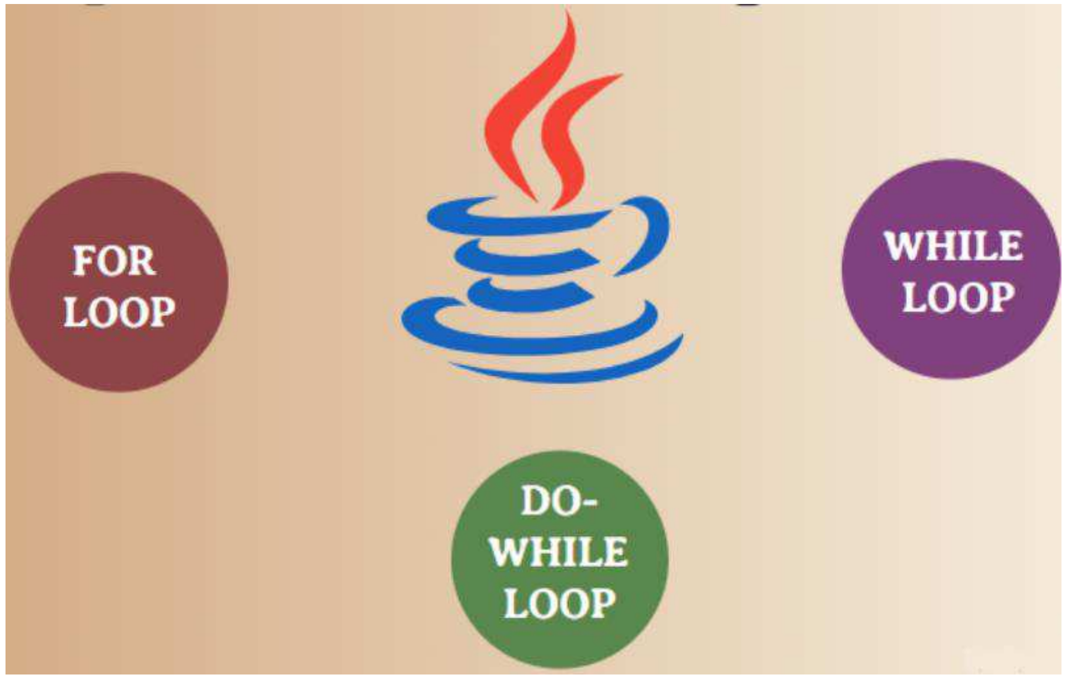
Session 03: Repetition Looping



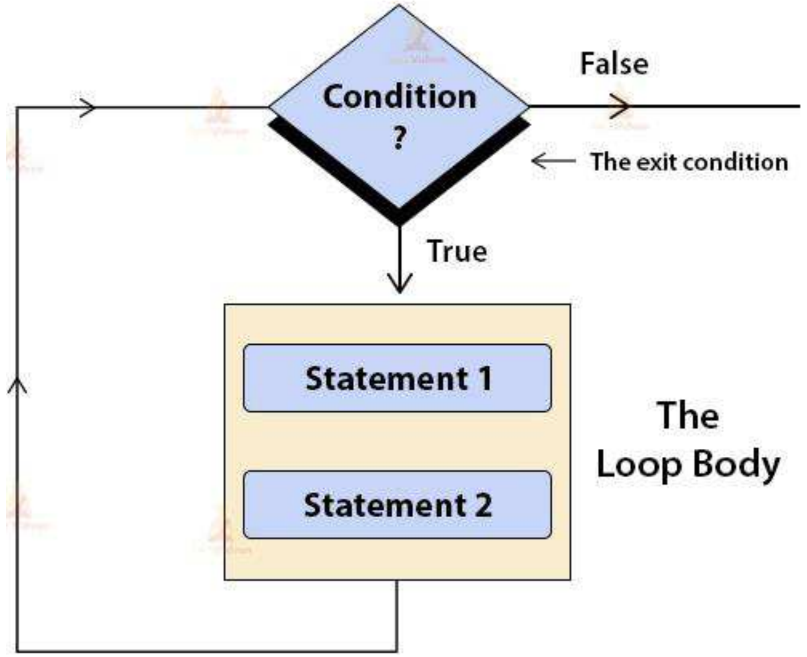
Objectives

Overview (1)

What are Loops in Java Programming

- While programming, sometimes, there occurs a situation when we **need to execute a block of code several numbers of times**
- **Loops in programming** allow a **set of instructions to be executed repeatedly** until a certain condition is **True**

```
1
2
3
4
5
```



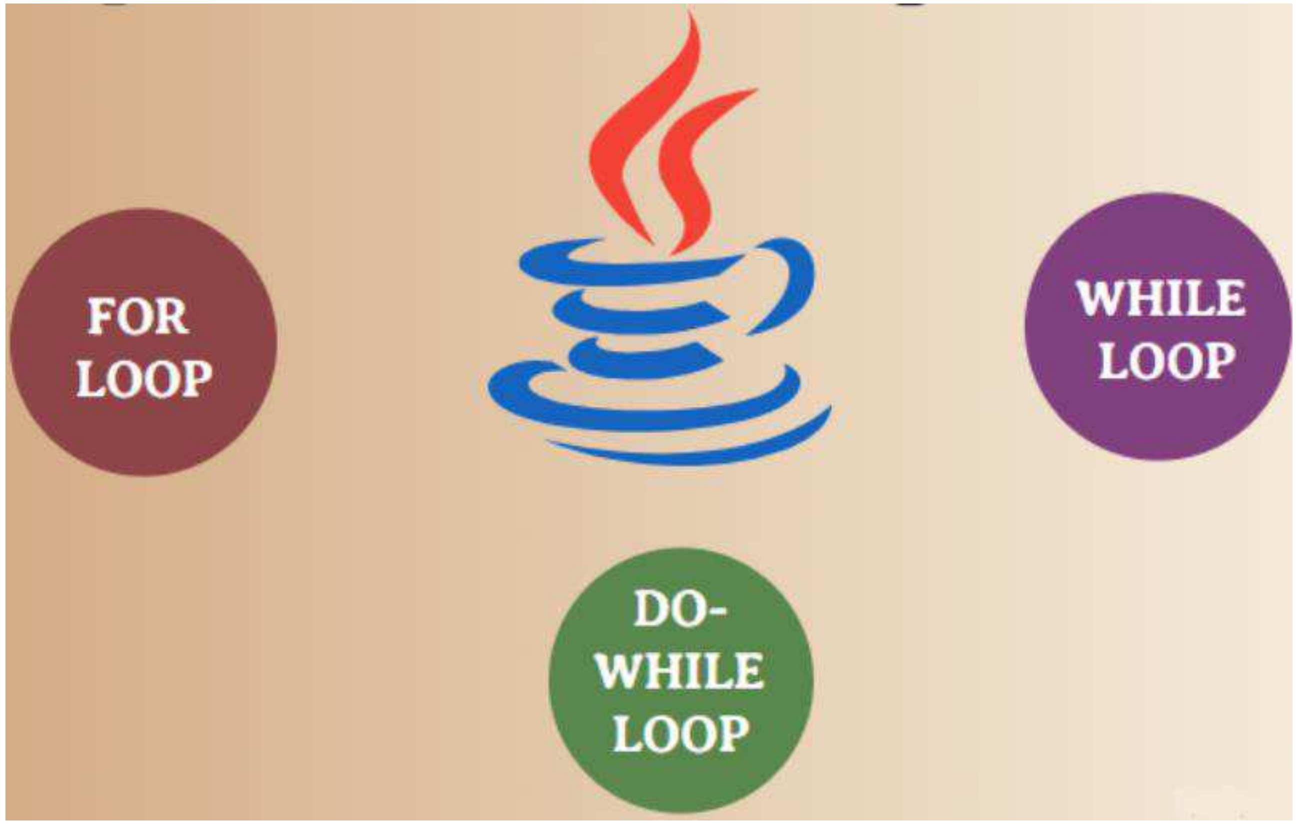
Overview (2)

Types of Loops

01 while loop

02 do-while loop

03 for loop



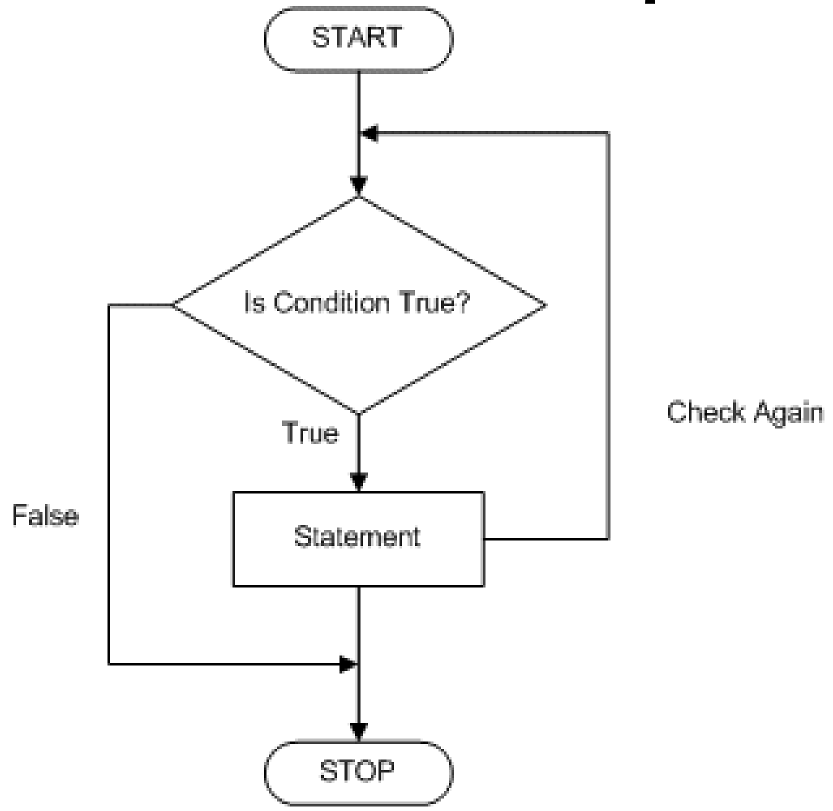
2

Types of Loops

while Loop (1)

- while loops are used for situations when a loop has to be executed as long as certain condition is True
- The **number of times** a loop is to be executed is **not pre-determined**, but **depends on the condition**
- Syntax

```
while (condition) {
    // action statements
}
```



while Loop (2)

Example

```
public class WhileDemo {
 public static void main (String[] args) {
   int num = 5, sum = 0;

   while (num >= 1) {
     sum += num;   // sum = sum + num;
     num--;
    }

   System.out.println("The sum is : " +sum);
 }
}
```

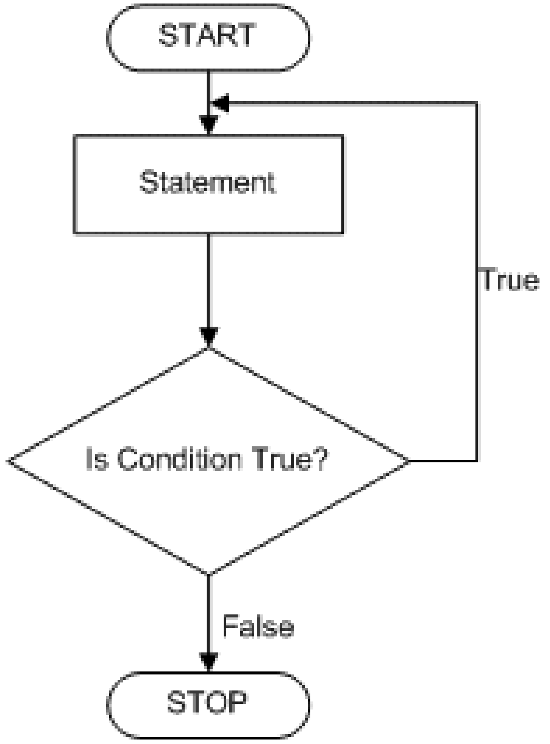
Output:

```
The sum is: 15
```

do - while Loop (1)

- The do-while loop executes certain statements till the specified condition is True
- These loops are similar to the while loops, except that a do-while **loop executes at least once**, even if the specified condition is False
- Syntax

```
do {
   // action statements
} while (condition);
```



do - while Loop (2)

Example

```
public class DoWhileDemo {
  public static void main (String[] args) {
    int count = 1, sum = 0;

    do {
        sum += count;
        count++;
     } while (count <= 10);

     System.out.println("The sum is: " + sum);
   }
}
```

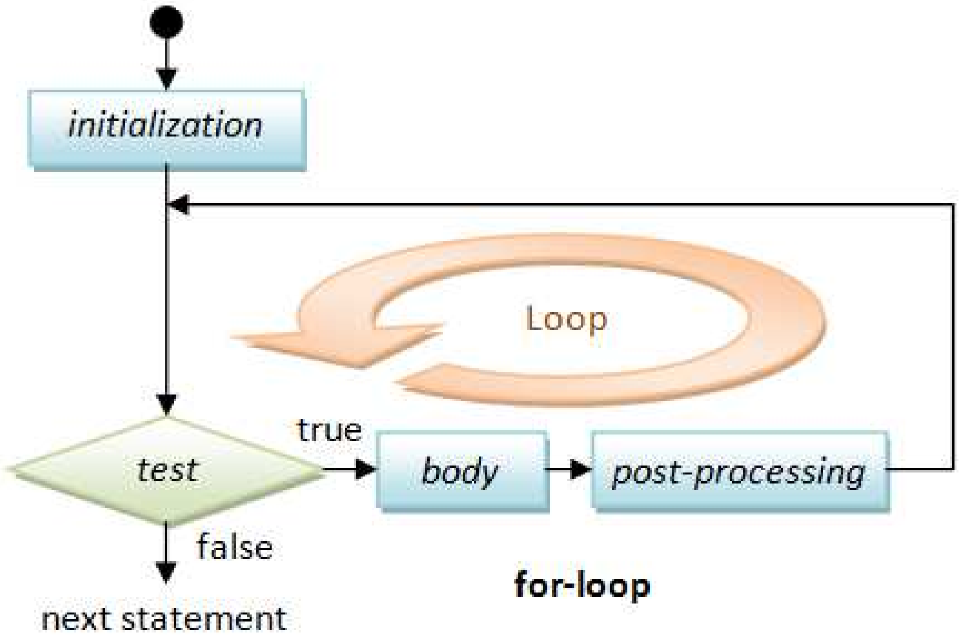
Output:

```
The sum is: 55
```

for Loop (1)

- All loops have some common features: a **counter variable** that is initialized before the loop begins, **a condition that tests** the counter variable and a statement that **modifies the value** of the counter variable
- The for loop provides a compact format for incorporating these features
- Syntax

```
for (initialization; condition; step) {
  // statement
}
```



for Loop (2)

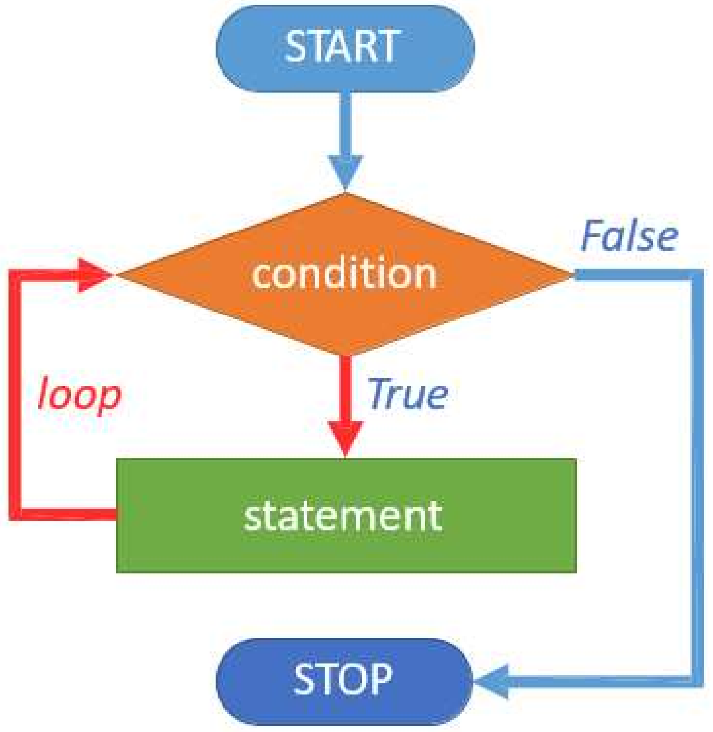
Example

```
public class ForDemo {
  public static void main (String[] args) {
    int count = 1, sum = 0;

   for (count = 1; count <= 10; count += 2) {
      sum += count;
   }

     System.out.println("The sum is : " + sum);
  }
}
```

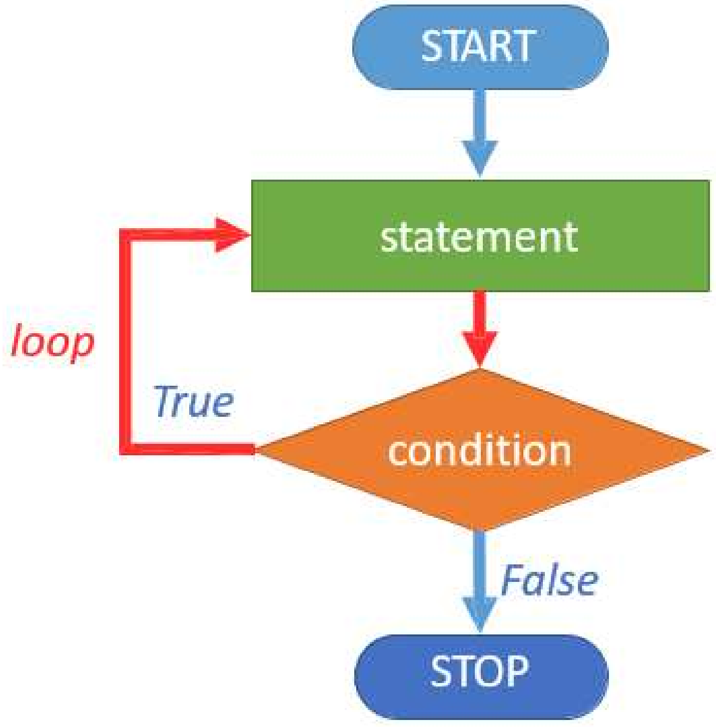
Output:

```
The sum is: 25
```

while vs do - while Loop



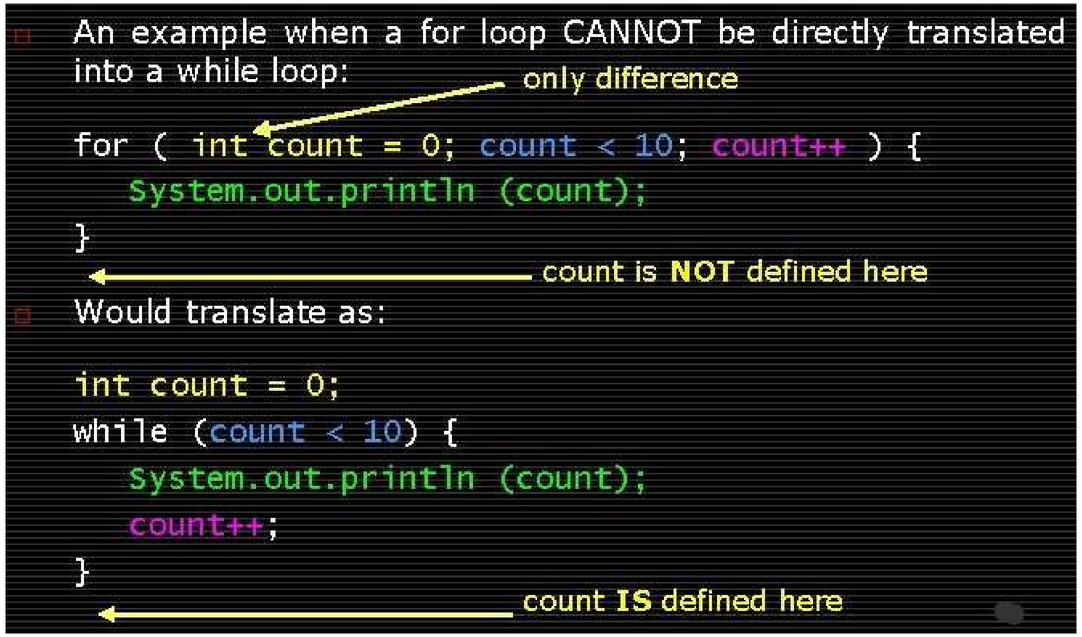
while



do ... while ...

for vs while Loop

FOR	WHILE
The 'for' loop used only when we already **knew the number of iterations**	The 'while' loop used only when the **number of iteration are not exactly known**



An example when a for loop CANNOT be directly translated into a while loop:

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

Would translate as:

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

3

Jumping

break Statements

- The break statement has two forms: **labeled** and **unlabeled**
- Use **unlabeled break** to terminate a **switch, for, while**, or **do-while loop**. Use **labeled** break to terminates an outer statement ~ **goto**
- Example

```
public class BreakDemo {
 public static void main(String[] args) {
   for (int count = 1; count <= 100; count++) {
     if (count == 6) {
       break;
     }
      System.out.println("The value of num is: " + count);
   }
  }
}
```

Output:

```
The value of num is: 1
The value of num is: 2
The value of num is: 3
The value of num is: 4
The value of num is: 5
```

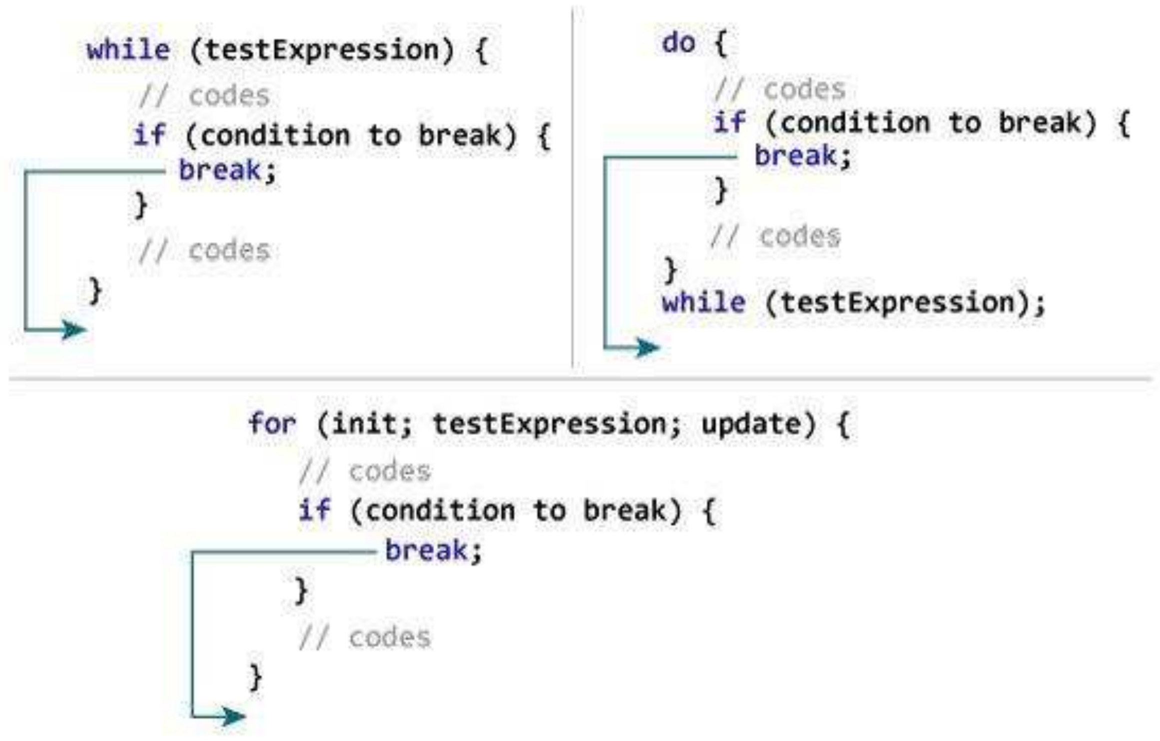
continue Statements

- The continue statement **skips the current iteration** of a for, while, or do-while loop
- Example

```
public class ContinueDemo {
 public static void main(String[] args) {
  byte sum = 0;
   for (int i = 1; i <= 10; i++) {
     if (i %2 == 0) {
       continue;
     }
      sum += count;
    }
    System.out.println("The sum is: " + sum);
  }
}
```

```
Output:
The sum is: 25
```

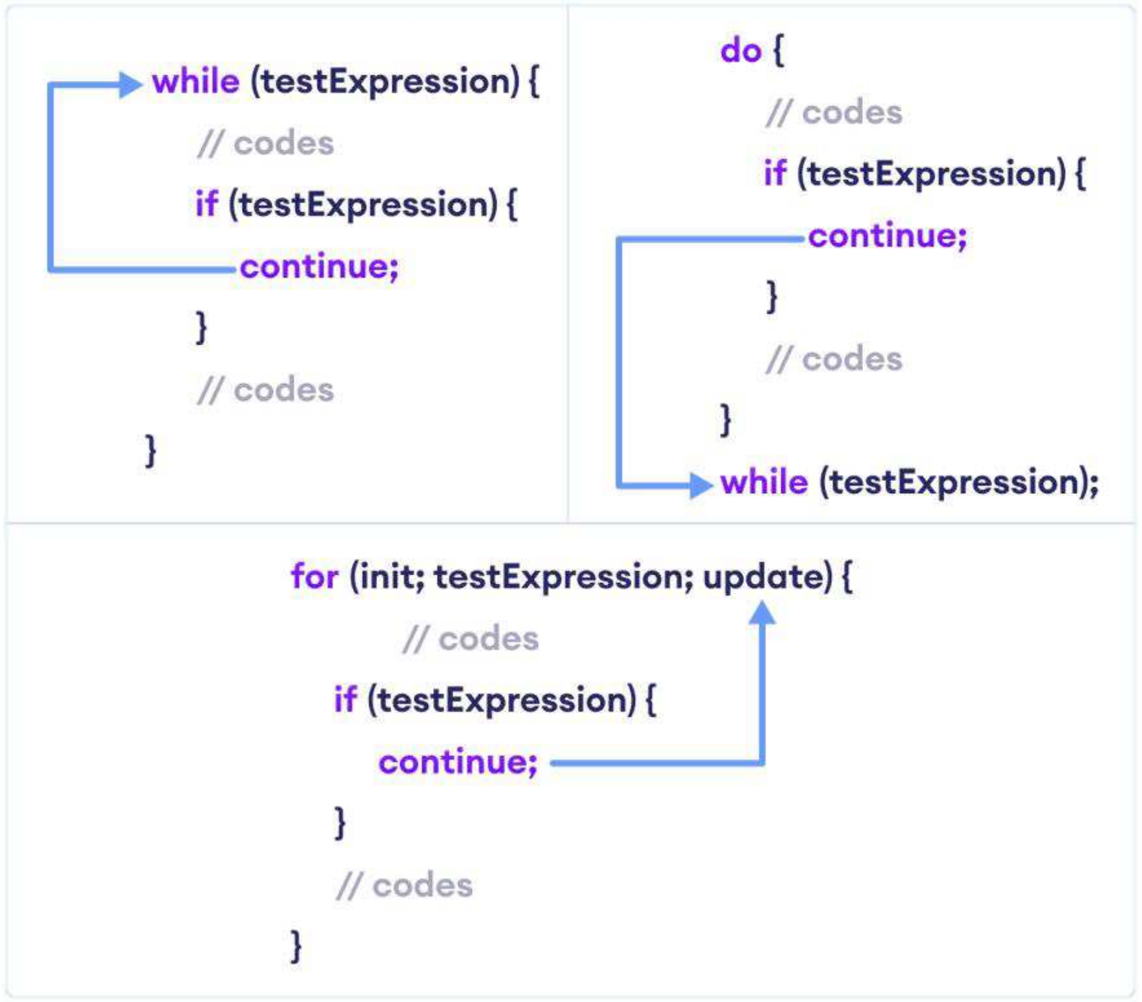
break vs continue Statements



```
while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

```
do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```



```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

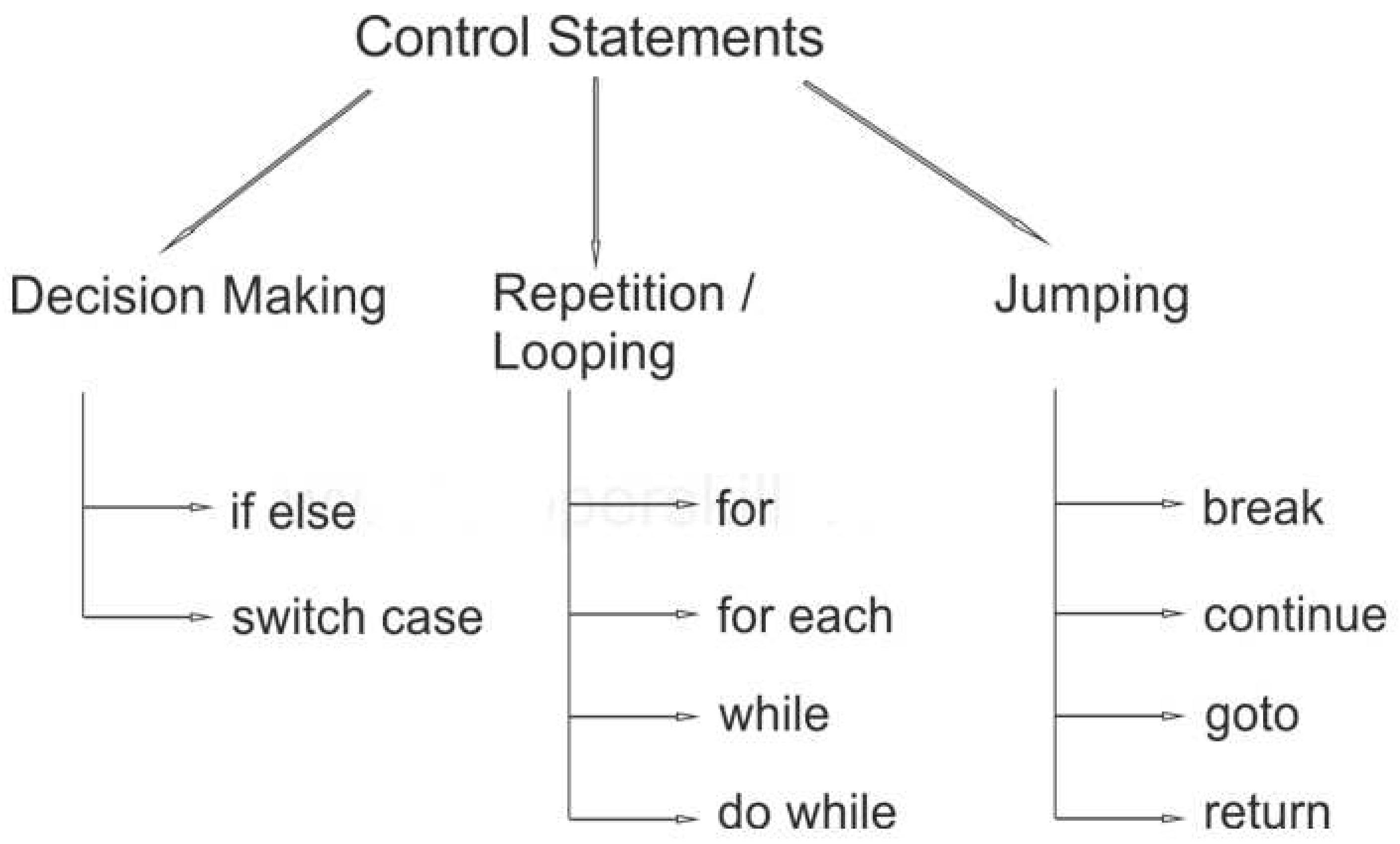
```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

Summary

Flow Control Statements

1. if..else
1. switch-case
2. while
3. do-while
4. for
5. break, continue



Resource Software Solution

Thankyou!