

## Lecture – 5 (Control Statement Part - 2)

### Java While Loop

The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

#### Syntax:

```
while(condition){  
    //code to be executed  
}
```

#### Example:

```
public class WhileExample {  
    public static void main(String[] args) {  
        int i=1;  
        while(i<=10){  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

#### Output:

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

#### Example:

```
public class WhileExample2 {
```

```
public static void main(String[] args) {  
    while(true){  
        System.out.println("infinitive while loop");  
    }  
}  
}
```

Output:

```
infinitive while loop  
infinitive while loop  
infinitive while loop  
infinitive while loop  
infinitive while loop
```

## Java do-while Loop

The Java do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

The Java do-while loop is executed at least once because condition is checked after loop body.

### Syntax:

```
do{  
    //code to be executed  
}while(condition);
```

### Example:

```
public class DoWhileExample {  
    public static void main(String[] args) {  
        int i=1;  
        do{  
            System.out.print(i+" ");  
            i++;  
        }while(i<=10);  
    }  
}
```

Output:

1 2 3 4 5 6 7 8 9 10

## Java Infinitive do-while Loop

If you pass **true** in the do-while loop, it will be infinitive do-while loop.

**Syntax:**

```
do{  
    //code to be executed  
}while(true);
```

**Example:**

```
public class DoWhileExample2 {  
    public static void main(String[] args) {  
        do{  
            System.out.println("infinitive do while loop");  
        }while(true);  
    }  
}
```

Output:

```
infinitive do while loop  
infinitive do while loop  
infinitive do while loop
```

## Java For Loop

The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

**Syntax:**

```
for(initialization;condition;incr/decr){  
    //code to be executed
```

```
}
```

**Example:**

```
public class ForExample {  
    public static void main(String[] args) {  
        int i;  
        for(i=1;i<=10;i++){  
            System.out.print(i+" ");  
        }  
    }  
}
```

Output:

1 2 3 4 5 6 7 8 9 10

## Java Infinitive For Loop

If you use two semicolons ;; in the for loop, it will be infinitive for loop.

**Syntax:**

```
for(;;){  
    //code to be executed  
}
```

**Example:**

```
public class ForExample {  
    public static void main(String[] args) {  
        for(;;){  
            System.out.println("infinitive loop");  
        }  
    }  
}
```

Output:

infinite loop  
infinite loop  
infinite loop  
infinite loop  
infinite loop

## Null Statement

If you use semicolon ; after for loop, it will be null statement. It will give you unexpected result.

### Example:

```
public class ForExample {  
    int i;  
    public static void main(String[] args) {  
        for(i=1;i<=10;i++) ;  
        {  
            System.out.print(i);  
        }  
    }  
}
```

## Java Break Statement

The Java break is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

### Java Break Statement with Loop

#### Example:

```
public class BreakExample {  
    public static void main(String[] args) {  
        int i;  
        for(i=1;i<=10;i++){  
            if(i==5){  
                break;  
            }  
            System.out.print(i+" ");  
        }  
    }  
}
```

```
}
```

Output:

1 2 3 4

## Java Continue Statement

The Java continue statement is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

### Java Continue Statement Example

**Example:**

```
public class ContinueExample {  
    public static void main(String[] args) {  
        int i;  
        for( i=1;i<=10;i++){  
            if(i==5){  
                continue;  
            }  
            System.out.print(i+" ");  
        }  
    }  
}
```

Output:

1 2 3 4 6 7 8 9 10