

# While Loop

Sungchul Lee

# Learning Object

- Repetition Statements
  - While Loop
    - ❑ Flow
    - ❑ Infinite loop – break
    - ❑ Continue
  - Do-While Loop
    - ❑ Flow
- Pre-test vs Post-test Loops



Conveyor belt – repeat same process

# Repetition Statements

- In a program, repetition statements control a block of code to be executed for many times
- Fixed number of times
  - ❑ **“for”** loop
  - ❑ **“for each”** loop
- Until a certain condition is met
  - ❑ **“while”** loop
  - ❑ **“do-while”** loop

# While Loop

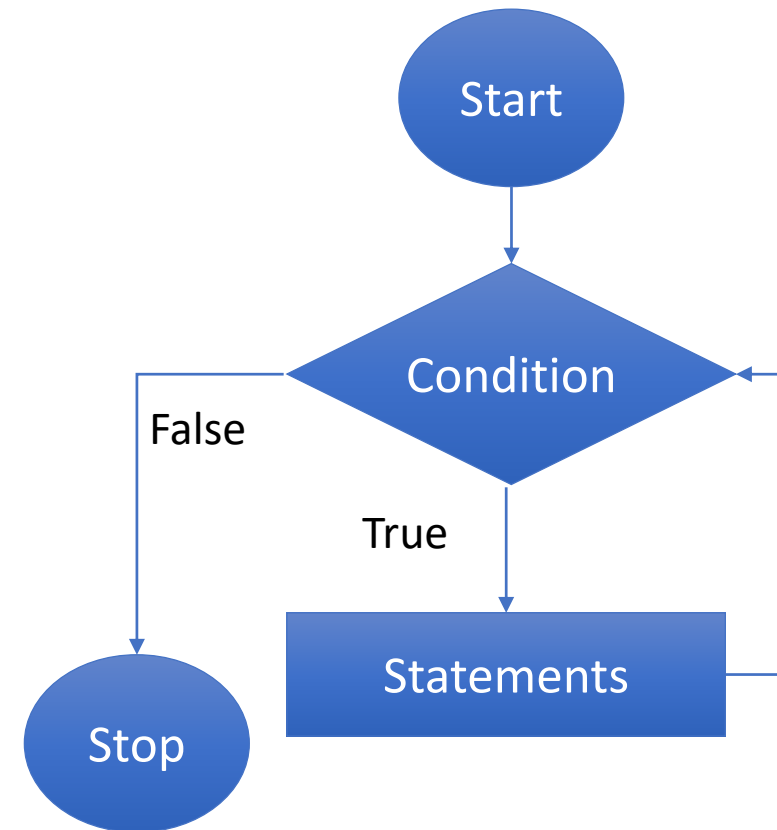
- The syntax for the while loop is as follows:

```
while ( loop-condition ) {  
    statement;  
}
```

- Note: if the condition is **true**, the loop body is executed; if the condition is false, the entire loop terminates.

# Flow of While Loop

- Evaluate Boolean expression
  - ❑ If true, execute statement
  - ❑ Re-evaluate Boolean expression
  - ❑ If false, terminate loop



# While loop - Example

Loop  
Condition

```
int sum = 0, i = 1;  
while (i < 10) { //loop-continuation-condition  
    sum = sum + i;  
    i ++;  
}  
System.out.println("sum is " + sum); //sum is 45
```

➤ **Note:** The **loop-continuation-condition** must always appear inside the **parentheses**. The **braces** enclosing the loop body can be omitted only if the loop body contains one or no statement.

# Infinite While Loop

- The syntax for the while loop is as follows:

```
while (true) {  
    statement;  
}
```

- While a condition is true , a while loop repeats

- Example:

```
while (true) {  
    System.out.println("To exit while loop: Ctrl-C");  
}
```

# Exit While Loop using Break Statement

## ➤ Syntax

```
while (true) {  
    statement;  
    if(condition){ break;}  
}
```

## ➤ Force to go out in a certain condition

## ➤ Example:

```
int count = 10;  
Boolean flag = false;  
while (true) {  
    while (true) {  
        count ++;  
        if(count > 20) {  
            flag = true;  
            break; // out inner while loop  
        }  
    }  
    if(flag) {break;} // out outer while loop  
}
```



# Continue Statement

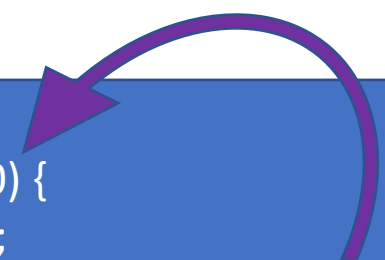
## ➤ Syntax

```
while (true) {  
    statement;  
    if(condition){ continue;}  
}
```

## ➤ Go to start of while loop

## ➤ Example:

When a is even number



```
int a = 0;  
while (a < 10) {  
    a++;  
    if (a % 2 == 0) { continue; }  
    System.out.println(a); //1,3,5,7,9  
}
```

# do-while Loop

➤ First, execute the loop body and then check condition

➤ Syntax

```
do {  
    statement;  
} while (loop-condition) ;
```

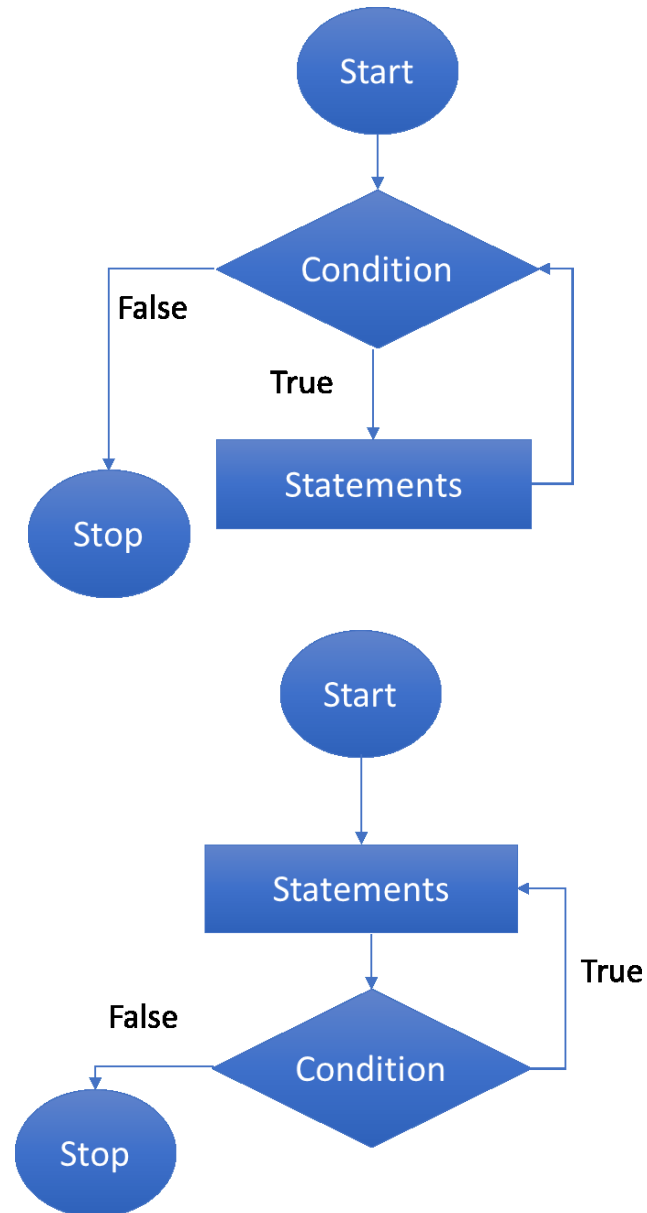
➤ **Note:** the semicolon!

➤ Example:

```
int sum = 0, i = 1;  
do{  
    sum = sum + i;  
    i ++;  
} while (i < 10); //loop-continuation-condition  
System.out.println("sum is " + sum); //sum is 45
```

# Pre-test vs. Post-test Loops

- The **while** loop and **for** loop are called pre-test loop because the continuation condition is checked before the loop body is executed.
- The **do-while** loop is called a post-test loop because the condition is checked after the loop body is executed.



# Practice

1. Make a new project (Reference: Create Project and Class File)

❑ Project name: While\_Loop

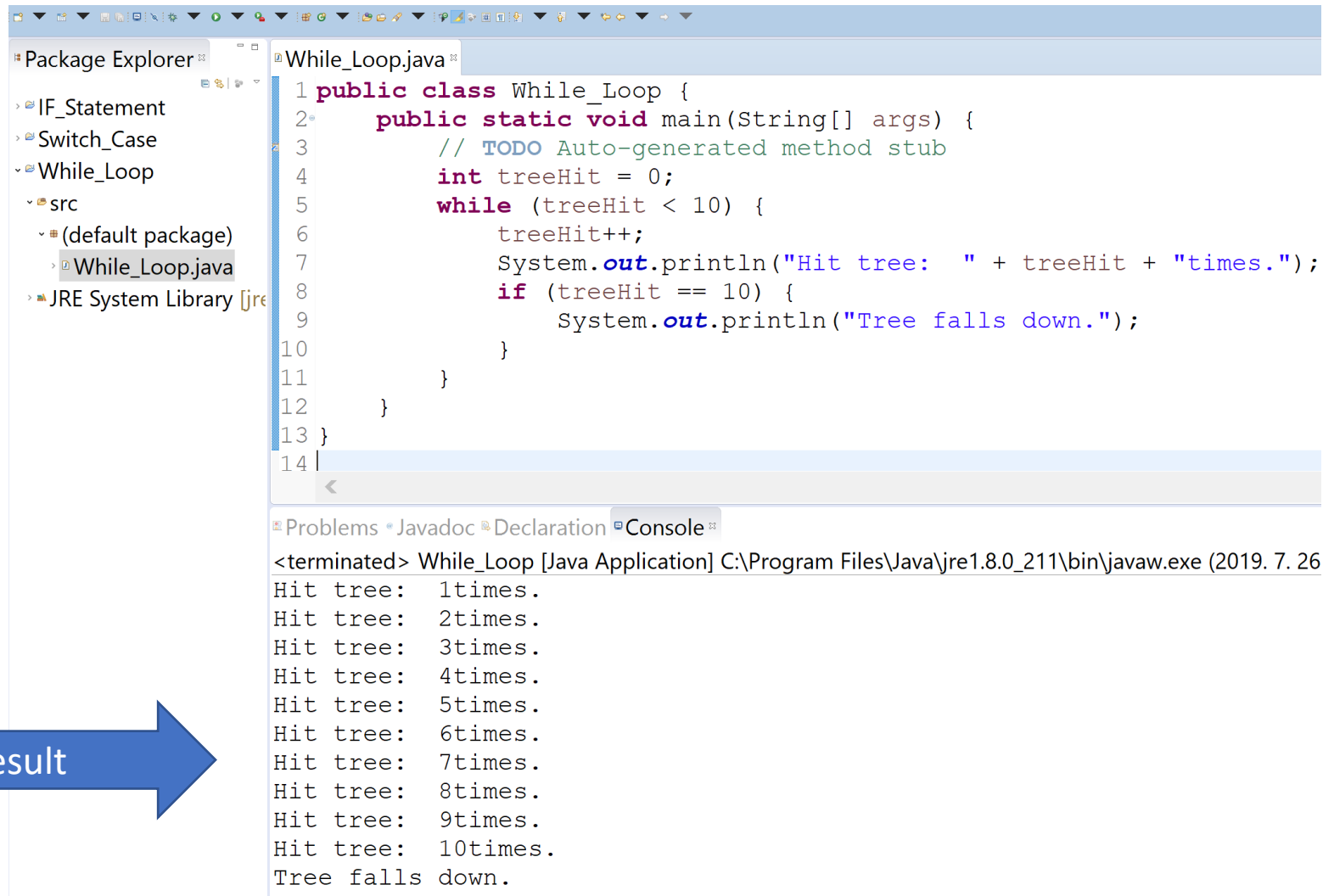
2. Create a new Class File

❑ Class name: While\_Loop

3. Coding:

```
public class While_Loop {  
    public static void main(String[] args) {  
        int treeHit = 0;  
        while (treeHit < 10) {  
            treeHit++;  
            System.out.println("Hit tree: " + treeHit + "times.");  
            if (treeHit == 10) {  
                System.out.println("Tree falls down.");  
            }  
        }  
    }  
}
```

# Practice – Code and Result



The screenshot displays an IDE with two main panels. The left panel, 'Package Explorer', shows a project structure with a package named 'While\_Loop' containing a file 'While\_Loop.java'. The right panel shows the code of 'While\_Loop.java' and its execution output in the 'Console' tab.

```
1 public class While_Loop {  
2     public static void main(String[] args) {  
3         // TODO Auto-generated method stub  
4         int treeHit = 0;  
5         while (treeHit < 10) {  
6             treeHit++;  
7             System.out.println("Hit tree:  " + treeHit + "times.");  
8             if (treeHit == 10) {  
9                 System.out.println("Tree falls down.");  
10            }  
11        }  
12    }  
13 }  
14
```

The console output shows the program's execution:

```
<terminated> While_Loop [Java Application] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (2019. 7. 26)  
Hit tree:  1times.  
Hit tree:  2times.  
Hit tree:  3times.  
Hit tree:  4times.  
Hit tree:  5times.  
Hit tree:  6times.  
Hit tree:  7times.  
Hit tree:  8times.  
Hit tree:  9times.  
Hit tree: 10times.  
Tree falls down.
```

Result

# Summary

## ➤ While Loop

- ❑ Infinite loop – break
- ❑ Continue

## ➤ Do-While Loop

```
While_Loop.java
1 public class While_Loop {
2     public static void main(String[] args) {
3         // TODO Auto-generated method stub
4         int treeHit = 0;
5         while (treeHit < 10) {
6             treeHit++;
7             System.out.println("Hit tree:  " + treeHit + "times.");
8             if (treeHit == 10) {
9                 System.out.println("Tree falls down.");
10            }
11        }
12    }
13 }
```

