Lecture 26

Jump Statements
(Break, Continue & Return)

Break :- It is used to terminate the loop or switch statement prematurely. When a break statement encounters, the loop or switch ends immediately and control passes to the next statement following the loop or switch.

e.g. :-
```
for( int i = 1; i<=10; i++)
{       if (i==5)
            break;

}
   System.out.println(i);   // error out of scope
}
```
⇒ output :-

if we modify it :-
```
for(int i=1; i<=10; i++) {
        if(i==5) {
            break;
        }
        System.out.println(i);
}
```
⇒ Output :-
```
1
2
3
4
```

In switch statement we use break in all cases to prevent fall-through.

e.g. -
```
for(int i=1; i<=5; i++){
    System.out.println("Inside for");
switch(i) {
case 3 :
        System.out.println(i);
        break;
}
System.out.println("Outside Switch");
}
System.out.println("Outside for");
```

this break will pass the control out of switch not out of for.

This means break will only exit the innermost loop or switch where it is placed. The outer loop or switch will continue to run as usual.

<u>NOTE</u>:- Break statement is restricted to use within loops & switch. We can use use a break inside an if block by itself without an enclosing loop or switch statement.

e.g:-
```
if (condition)
{
        break; // Compilation error.
}
```

<u>labeled Break</u>:- A labeled break statement allows you to break out of a specific loop when you have nested loops.

e.g:-
```
labelfor: for (int i =1; i<=5; i++) {
        System.out.println(" Inside for");
        switch (i) {
            Case 3 :
                System.out.println(i);
                break labelfor;
        }
        System.out.println("Outside switch");
}
System.out.println("Outside for");
```

This break will pass the control out of for loop enclosing switch

Example 2:-

```
for( int i=1; i<=3; i++) {
    for(int j=1 ; j<=3; j++) {
        if( i==1 && j==1) {
            break;
        }
        System.out.println("i=" +i+ ", j=" +j);
    }
}
```

Output:-

```
i=2, j=1
i=2, j=2
i=2, j=3
i=3, j=1
i=3, j=2
i=3, j=3
```

Quiz Que: Now what if I use labeled break in above example.

```
labelfor: for(int i=1; i<=3; i++) {
              for(j=1; j<=3; j++) {
                  if (i==1 && j==1) {
                      break labelfor;
                  }
                  System.out.println("i=" +i+ ",j=" +j);
              }
          }
```

Output?

NOTE:- labeled break can be used with if statement as well. ⓤ Contin

```
iflabel:  if ( condition)
          {
               break ifLabel;    => No error.
          }
```

e.g.-

```
int num = 5;
outerIf:
if(num >0) {
    System.out.println("Number is positive.");
    if (num %2 ==0) {
        System.out.println("Number is even");
    }else{
        System.out.println("Number is odd.");
        break outerIf;
    }
    System.out.println(" This will only print if the number
                                         is even.");
}
System.out.println(" This is outside the Labeled block.");
```

Output:-

```
Number is positive
Number is odd
This is outside the labeled block.
```

Quiz Time:-

```
for(int i=0; i<5 ; i++)
{   System.out.println(i);
     if (i==3) {
            break;
}    }
```

=> what would be output of this code ?

Continue Statement :-

↳ It skips the current iteration of a loop and moves to the next iteration

- It doesn't terminate the loop, It just bypasses the rest of the code in the current iteration.

Syntax →

```
continue;
```

e.g:-

```
for( int i = 1; i<=10; i++){
        if( i== 3) {
              Continue;
        }
        System.out.println(i);
}
```

- Continue statement is primarily used within loops (for, while, do while)
- It can not be used directly within switch & if. But it works with switch & if only if these are within loop.
- We use continue to filter out specific elements. If we don't want to process some data then we can use continue to skip that.

e.g: - (i)

```
for( int i = 1; i<=10; i++){
        if( i % 2 != 0) {
              Continue;
        }
        System.out.println(i);
}
```

⇒ output -

```
2
4
6
8
10
```

Practice Question (ii) Print only odd, skipping even number (use while loop)

(iii)

```
int i=5;
if (i==5) {
      Continue;
}
```

⇒ what would be output ?

Labeled continue :-

```
outer loop ↗ for(int i = 1 ; i<=5; i++){
                 System.out.println("Outer loop iteration : " +i);
inner loop ↗     for(int j=1 ;j<=5; j++) {
                     if (j==3) {
                         System.out.println("skipping inner loop iteration when
                                                  j is " +j);
                         continue;
                     }
                     System.out.println("Inner loop iteration: " +j);
                 }
             }
```

IMP :- Here continue is called inside inner loop. So it skips the rest of the code inside the inner loop & continues with the next iteration of that loop.

- It doesn't affect the outer loop, it only applies to the loop in which it is called.

⇒ So here if we need to skip the current iteration of an outer loop from within a nested loop then we can use $\boxed{\text{labeled continue}}$.

By using a label, you can specify which loop to continue when continue statement is executed.

⇒ In above example use labeled continue like:-

```
outerfor :  for(int i=1; i<=5 ; i++) {
                 - .
                 - .
                 - -
                 if(j==3) {
                 - -
                     continue outerfor;
                 }
                 - -
```

→ A label should be a valid java identifier with a colon (:)

→ labeled break & continue statement must be called within their scope.

```
outerloop: for(int i = 1; i <=5; i++) {
        innerloop: for(int j = 1; j<=5; j++) {
                        if(j = = 3) {
                              Continue innerloop;
                        }
                   }
                   if(i = = 3) {
                        Continue innerloop;   //error
                   }
           }
```

Scope of innerloop label is till end of innerloop.

scope of outerloop

→ In Java, a labeled statement is a way to define a label for a block of code or a loop. This label can be used to identify the block of code, making it easier to control the flow of execution, especially in nested loops or when using break & continue.

Syntax :- $\boxed{\text{label : statement ;}}$

Here statement can be a loop, a block of code or any executable statements.

Practice Time:-

```
int num=0;
while(num<20) {
    num++;
   if(num<=7) {
        continue;
   }
   System.out.println("number = " +num);
   if(num>=15) {
        break;
   }
}
```