Java Fundamentals

Siddharth Sharma

JUMP STATEMENTS

- **Break:**

In Java, the **break** statement has three uses. First, as you have seen, it terminates a statement sequence in a **switch** statement. Second, it can be used to exit a loop. Third, it can be used as a “civilised” form of goto.

- **Using break to Exit a Loop:**

By using **break**, you can force immediate termination of a loop, bypassing the conditional expression and any remaining code in the body of the loop.

```
class example
{
    public static void main(String args[])
    {
        for(int i=0;i<3;i++)
        {
            System.out.print("Pass "+(i+1)+": ");
            for(int j=0;j<100;j++)
            {
                System.out.print(j+" ");
                if(j==4)
                break;
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

```
Pass 1: 0 1 2 3 4
Pass 2: 0 1 2 3 4
Pass 3: 0 1 2 3 4
```

- The **break** statement in the inner loop only causes termination of that loop. The outer loop is unaffected.

- **Using break as a Form of Goto:**

The goto can be useful when you are exiting from a deeply nested set of loops. To handle such situations, Java defines an expanded form of the break statement. By using this form of break, you can, for example, break out of one or more blocks of code. **These blocks need not be part of a loop or a switch.** They can be any block. Further, you can specify precisely where execution will resume, because this form of **break** works with a label.

The general form of the labeled **break** statement is shown here:

break *label;*

```
class example
{
   public static void main(String args[])
   {
      boolean t=true;
      first:
      {
         second:
         {
            third:
            {
               System.out.print("Java is a ");
               if(t)
               break second;
            }
            System.out.print("beautiful ");
         }
         System.out.print("language.");
      }
   }
}
```

OUTPUT:

```
Java is a language.
```

- Using **labelled break** to exit from nested loops:

```
class example
{
   public static void main(String args[])
   {
      outer: for(int i=0;i<3;i++)
      {
         System.out.print("Pass "+i+":");
         for(int j=0;j<100;j++)
```

```
            {
                System.out.print(" "+j+" ");
                if(j==10)
                break outer;
            }
            System.out.println();
        }
    }
}
```

OUTPUT:

```
Pass 0: 0  1  2  3  4  5  6  7  8  9  10
```

- **Continue:**

Sometimes it is useful to force an early iteration of a loop. That is, one might want to continue running the loop but stop processing the remainder of the code in its body for this particular iteration.

- In **while** and **do-while** loops, a **continue** statement causes control to be transferred directly to the conditional expression that controls the loop.
- In a **for** loop, control goes first to the iteration portion of the **for** statement and then to the conditional expression.
- For all three loops, any intermediate code is bypassed.

```
class example
{
    public static void main(String args[])
    {
        for(int i=0;i<10;i++)
        {
            System.out.print(i+" ");
            if(i%2==0)
            continue;
            System.out.println();
        }
    }
}
```

OUTPUT:

```
0 1
2 3
4 5
6 7
8 9
```

• As with the **break** statement, **continue** may specify a label to describe which enclosing loop to continue.

```
class example
{
   public static void main(String args[])
   {
      outer:for(int i=0;i<10;i++)
      {
         for(int j=0;j<10;j++)
         {
            if(j>i)
            {
               System.out.println();
               continue outer;
            }
            System.out.print(" "+i*j);
         }
      }
      System.out.println();
   }
}
```

OUTPUT:

```
 0
 0 1
 0 2 4
 0 3 6 9
 0 4 8 12 16
 0 5 10 15 20 25
 0 6 12 18 24 30 36
 0 7 14 21 28 35 42 49
 0 8 16 24 32 40 48 56 64
 0 9 18 27 36 45 54 63 72 81
```