Java Fundamentals

Siddharth Sharma

CONTROL STATEMENTS

Selection Statements:

If and **switch** are the selection statements in Java. These statements allow you to control the flow of your program's execution based upon **conditions known only during run time**.

IF:

if (*condition*) *statement1*;

else *statement2*;

The *condition* is any expression that returns a **boolean** value. The **else** clause is optional.

If-else-if Ladder:

if(*condition*) *statement*;
else if(*condition*) *statement*;
else if(*condition*)
statement; .
.
. else
statement;

SWITCH:

```
switch (expression)
 {
        case value1:
                // statement sequence
                break;
```

```
        case value2:
                // statement sequence
                break; .
                . .
        case valueN:
                // statement sequence
                break;
        default:
                // default statement sequence
}
```

The *expression* must be of type **byte**, **short**, **int**, or **char**; each of the *values* specified in the **case** statements must be of a type compatible with the expression. (An enumeration value can also be used to control a **switch** statement.) Each **case** value must be a unique literal (that is, it must be a constant, not a variable). Duplicate **case** values are not allowed.

```
class example
{
  public static void main(String args[])
  {
     int i;
     for(i=0;i<6;i++)
     {
        switch(i)
        {
              case 0:
             {
                System.out.println("i is zero");
                break;
             }
             case 1:
             {
                System.out.println("i is one");
                break;
             }
             case 2:
             {
                System.out.println("i is two");
                break;
             }
             case 3:
             {
                System.out.println("i is three");
                break;
             }
             default:
             {
                System.out.println("i is greater than three");
```

```
            }
        }
    }
  }
}
```

OUTPUT:

```
i is zero
i is one
i is two
i is three
i is greater than three
i is greater than three
```

- **What if the break statement is not included?**

```
class example
{
  public static void main(String args[])
  {
     int i;
     for(i=0;i<6;i++)
     {
        switch(i)
        {
                case 0:
               {
                  System.out.println("i is zero");
                  break;
               }
               case 1:
               {
                  System.out.println("i is one");
                  break;
               }
               case 2:
               {
                  System.out.println("i is two");

               }
               case 3:
               {
                  System.out.println("i is three");

               }
               default:
               {
                  System.out.println("i is greater than three");
               }
        }
     }
  }
}
```

OUTPUT:

```
i is zero
i is one
i is two
i is three
i is greater than three
i is three
i is greater than three
i is greater than three
i is greater than three
```

All cases after the first case that matches the value of switch expression are evaluated.

```
class Switch {
   public static void main(String args[]) {
      int month = 4;
      String season;
      switch (month) {
        case 12:
        case 1:
        case 2:
          season = "Winter";
          break;
        case 3:
        case 4:
        case 5:
          season = "Spring";
          break;
        case 6:
        case 7:
        case 8:
          season = "Summer";
          break;
        case 9:
        case 10:
        case 11:
          season = "Autumn";
          break;
        default:
          season = "Bogus Month";
      }
      System.out.println("April is in the " + season + ".");
   }
}
```

OUTPUT:

```
April is in the Spring.
```

SUMMARY:

- The **switch** differs from the **if** in that **switch** can only test for equality, whereas **if** can evaluate any type of Boolean expression. That is, the **switch** looks only for a match between the value of the expression and one of its **case** constants.

- No two **case** constants in the same **switch** can have identical values. Of course, a **switch** statement and an enclosing outer **switch** can have **case** constants in common.

- A **switch** statement is usually more efficient than a set of nested **if**s.

The last point is particularly interesting because it gives insight into how the Java compiler works. When it compiles a **switch** statement, the Java compiler will inspect each of the **case** constants and create a "jump table" that it will use for selecting the path of execution depending on the value of the expression. Therefore, if you need to select among a large group of values, a **switch** statement will run much faster than the equivalent logic coded using a sequence of **if-else**s. The compiler can do this because it knows that the **case** constants are all the same type and simply must be compared for equality with the **switch** expression. The compiler has no such knowledge of a long list of **if** expressions.