CONTROL STATEMENTS

Velocity

JULY 22, 2022

VELOCITY

Praveen bhosle

Control Statements in java

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, **Java provides statements that can be used to control the flow of Java code**. Such **statements are called control flow statements.** It is one of the fundamental features of Java, which provides a smooth flow of program.

We can majorly divide control statements in Java into three major types:

- Decision-making statement
- Loop statement
- Break/Continue statement

Decision-making statement

- As the name suggests, decision-making statements decide which statement to executed and when it should be executed.

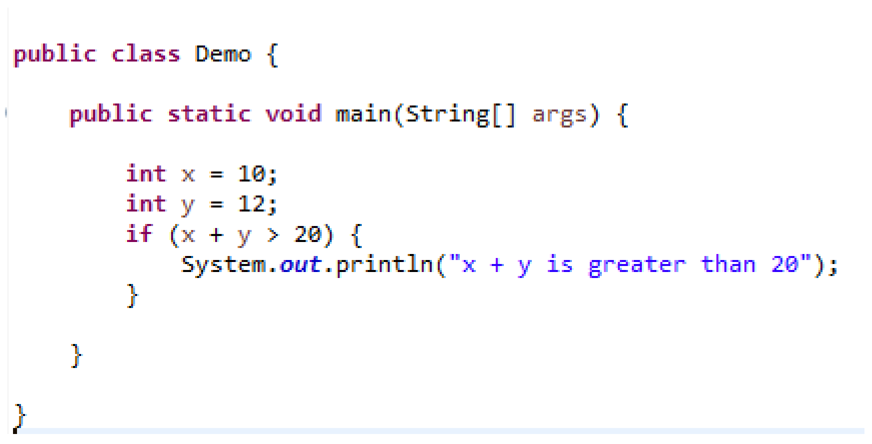
- Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided.

There are four types of control statements in java

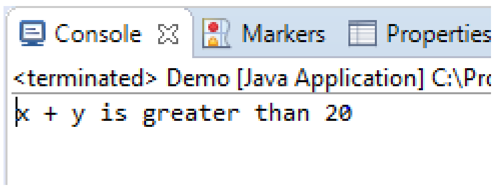
1. If statement
2. If else statement
3. if-else-if ladder statement
4. Nested if statement
5. Switch statement.

1. If statement: It is the most basic statement among all control flow statements in Java. It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.

   Syntax:
   ```
   if (Condition) {
   Statement 1;
   }
   Statement 2;
   ```



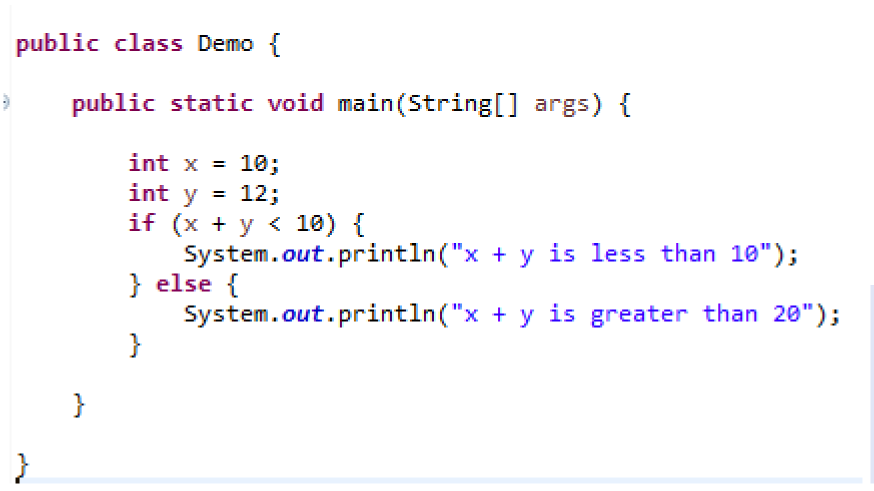
```
public class Demo {

    public static void main(String[] args) {

        int x = 10;
        int y = 12;
        if (x + y > 20) {
            System.out.println("x + y is greater than 20");
        }

    }

}
```



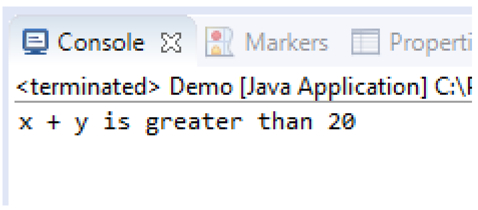
```
<terminated> Demo [Java Application] C:\Pr
x + y is greater than 20
```

2. If-else statement: The if-else statement is an extension to the if-statement, which uses another block of code, i.e., else block. The else block is executed if the condition of the if-block is evaluated as false.

   Syntax:
   if(condition) {
   statement 1; //executes when condition is true
   }
   else {
   statement 2; //executes when condition is false
   }



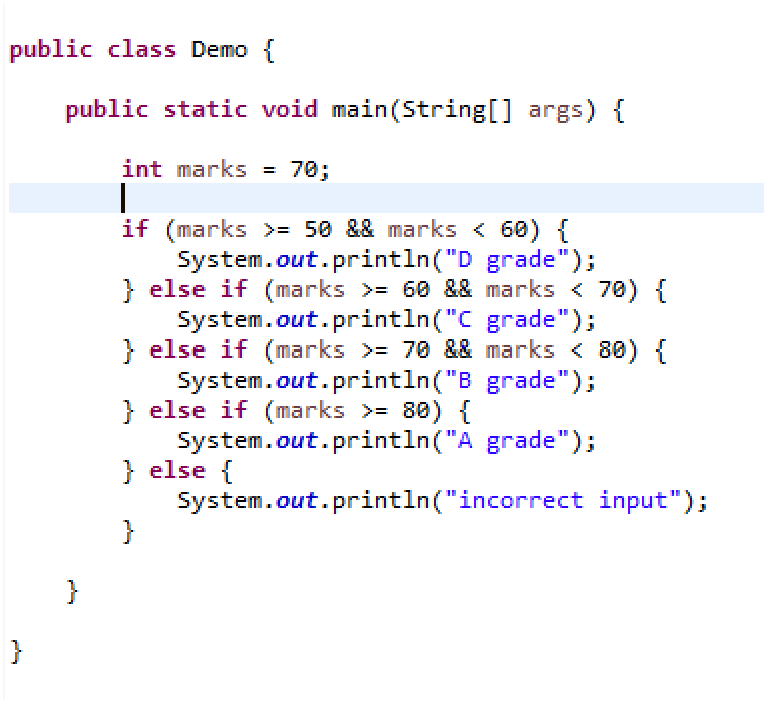
```
public class Demo {

    public static void main(String[] args) {

        int x = 10;
        int y = 12;
        if (x + y < 10) {
            System.out.println("x + y is less than 10");
        } else {
            System.out.println("x + y is greater than 20");
        }

    }

}
```



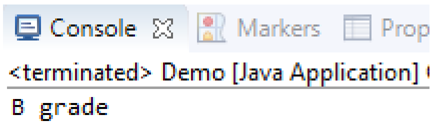
```
<terminated> Demo [Java Application] C:\
x + y is greater than 20
```

3. if-else-if ladder statement: The if-else-if statement contains the if-statement followed by multiple else-if statements.
   In other words, we can say that it is the chain of if-else statements that create a decision tree where the program may enter in any of the block of code where the condition is true. We can also define an else statement at the end of the chain.

Syntax:
```
If (condition 1) {
statement 1; //executes when condition 1 is true
}
else if (condition 2) {
statement 2; //executes when condition 2 is true
}
else if (condition 3) {
statement 3; //executes when condition 3 is true
}
else {
statement 2; //executes when all the conditions are false
}
```



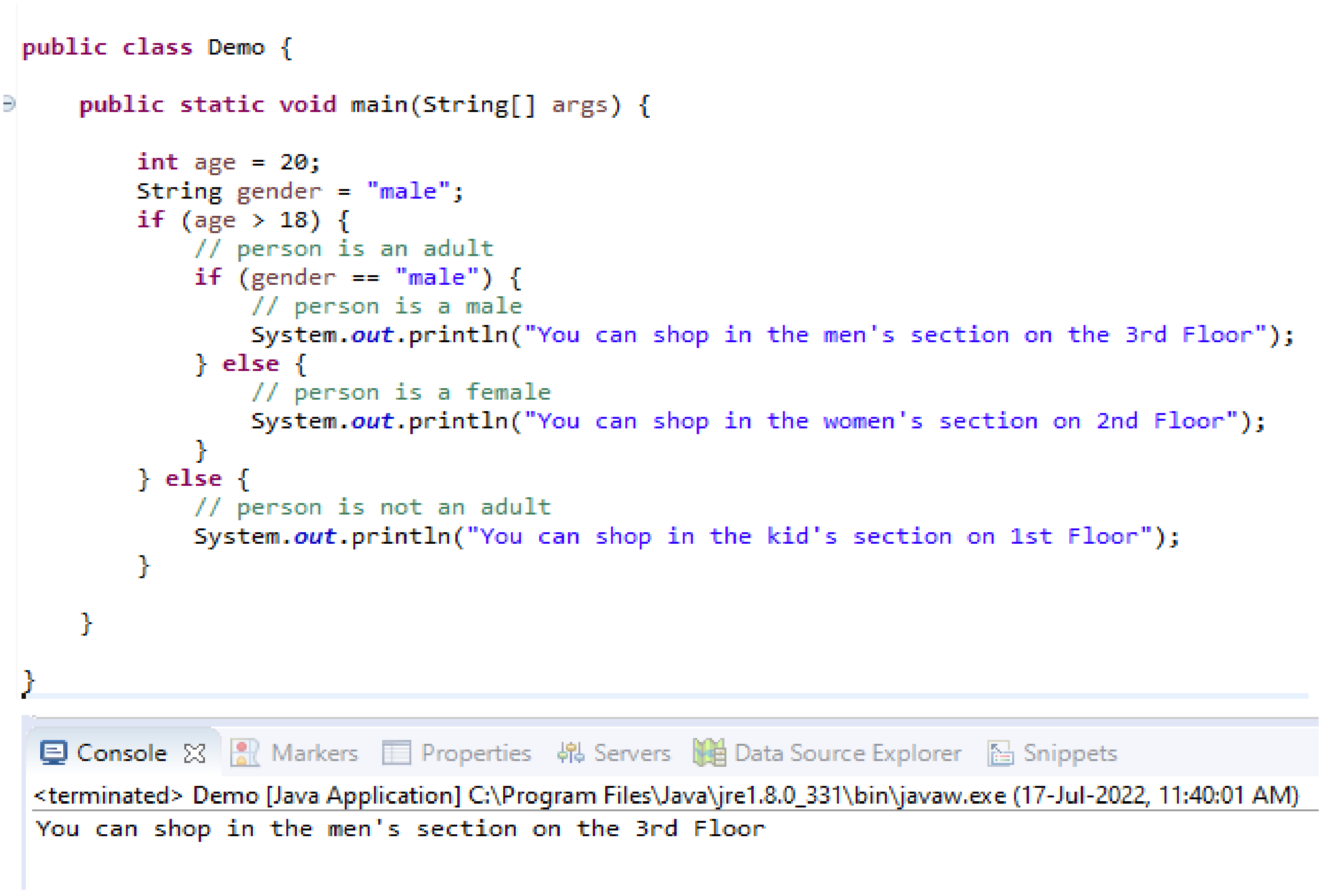
```java
public class Demo {

    public static void main(String[] args) {

        int marks = 70;

        if (marks >= 50 && marks < 60) {
            System.out.println("D grade");
        } else if (marks >= 60 && marks < 70) {
            System.out.println("C grade");
        } else if (marks >= 70 && marks < 80) {
            System.out.println("B grade");
        } else if (marks >= 80) {
            System.out.println("A grade");
        } else {
            System.out.println("incorrect input");
        }

    }

}
```



```
<terminated> Demo [Java Application]
B grade
```

4. Nested if statement: The nested if statement represents the if block within another if block. Here, the inner if block condition executes only when outer if block condition is true.
In nested if-statements, the if statement can contain a if or if-else statement inside another if or else-if statement.

```
Syntax:
if (condition 1) {
statement 1; //executes when condition 1 is true
if (condition 2) {
statement 2; //executes when condition 2 is true
}
else {
statement 2; //executes when condition 2 is false
}
}
```

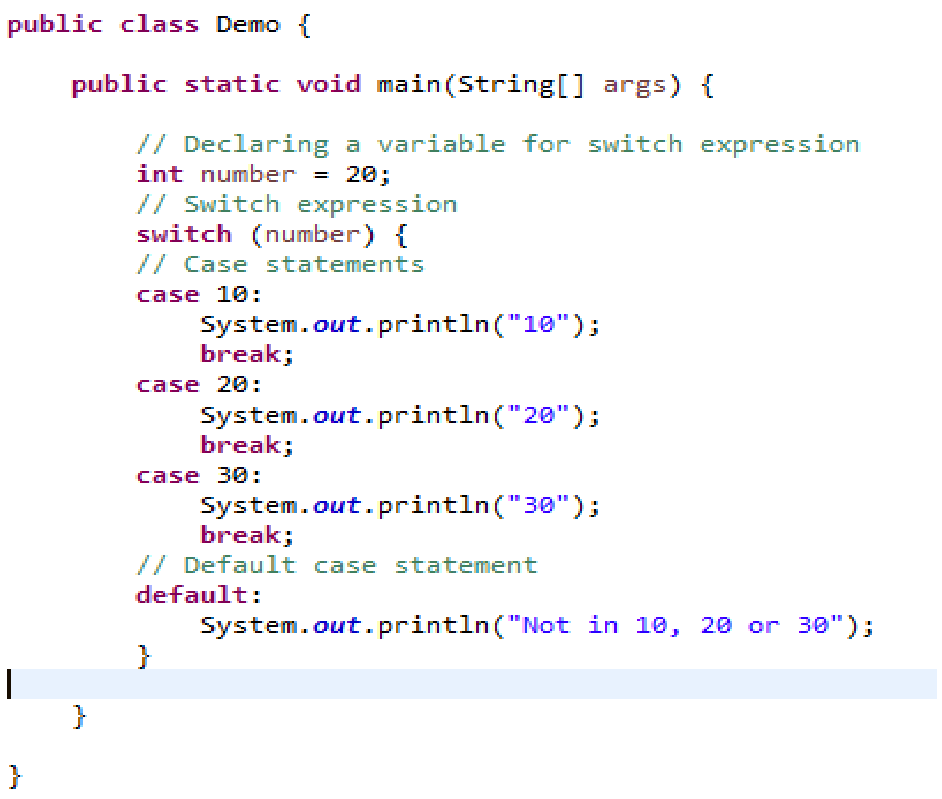


```
public class Demo {

    public static void main(String[] args) {

        int age = 20;
        String gender = "male";
        if (age > 18) {
            // person is an adult
            if (gender == "male") {
                // person is a male
                System.out.println("You can shop in the men's section on the 3rd Floor");
            } else {
                // person is a female
                System.out.println("You can shop in the women's section on 2nd Floor");
            }
        } else {
            // person is not an adult
            System.out.println("You can shop in the kid's section on 1st Floor");
        }

    }

}
```

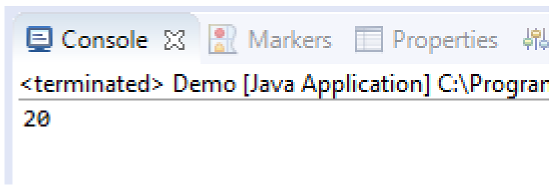
```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (17-Jul-2022, 11:40:01 AM)
You can shop in the men's section on the 3rd Floor
```

5. Switch statement: A switch statement in java is used to execute a single statement from multiple conditions. The switch statement can be used with short, byte, int, long, enum types, etc. Usage of break statement is made to terminate the statement sequence. It is optional to use this statement.

```
Syntax:
switch(expression) {
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
......

default:
  code to be executed if all cases are not matched;
}
```



```java
public class Demo {

    public static void main(String[] args) {

        // Declaring a variable for switch expression
        int number = 20;
        // Switch expression
        switch (number) {
        // Case statements
        case 10:
            System.out.println("10");
            break;
        case 20:
            System.out.println("20");
            break;
        case 30:
            System.out.println("30");
            break;
        // Default case statement
        default:
            System.out.println("Not in 10, 20 or 30");
        }

    }

}
```



```
<terminated> Demo [Java Application] C:\Prograr
20
```