Flow Control Statements

Objectives

At the end of this module, you will be able to work with :

- Selection statements
- Iteration statements
- Jumping statements

Flow Control

Control Statements

- Control statements are statements which alter the normal execution flow of a program.

- There are three types of Control Statements in java :

Selection statement	Iteration Statement	Jumping Statement
if	while	break
if – else	for	continue
switch	do – while	return

Simple if statement

syntax :

```
if(boolean expression)
{
    statement-block;
}
Next statement;
```
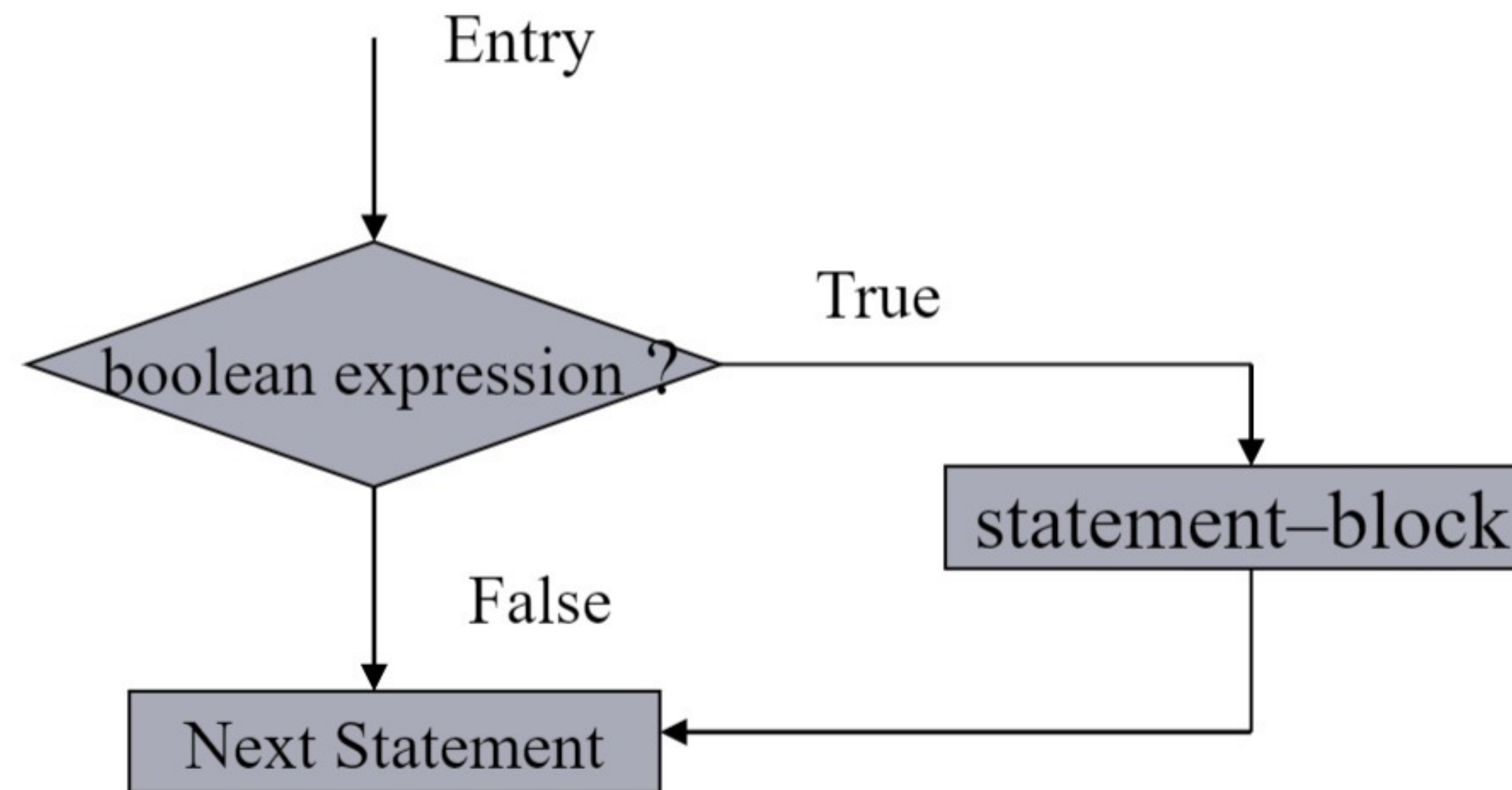
If - Example

```java
/* This is an example of a if statement */

    public class Test {

        public static void main(String args[]) {

            int x = 5;

            if( x < 20 ) {

                System.out.print("This is if statement");

            }

        }

    }
```
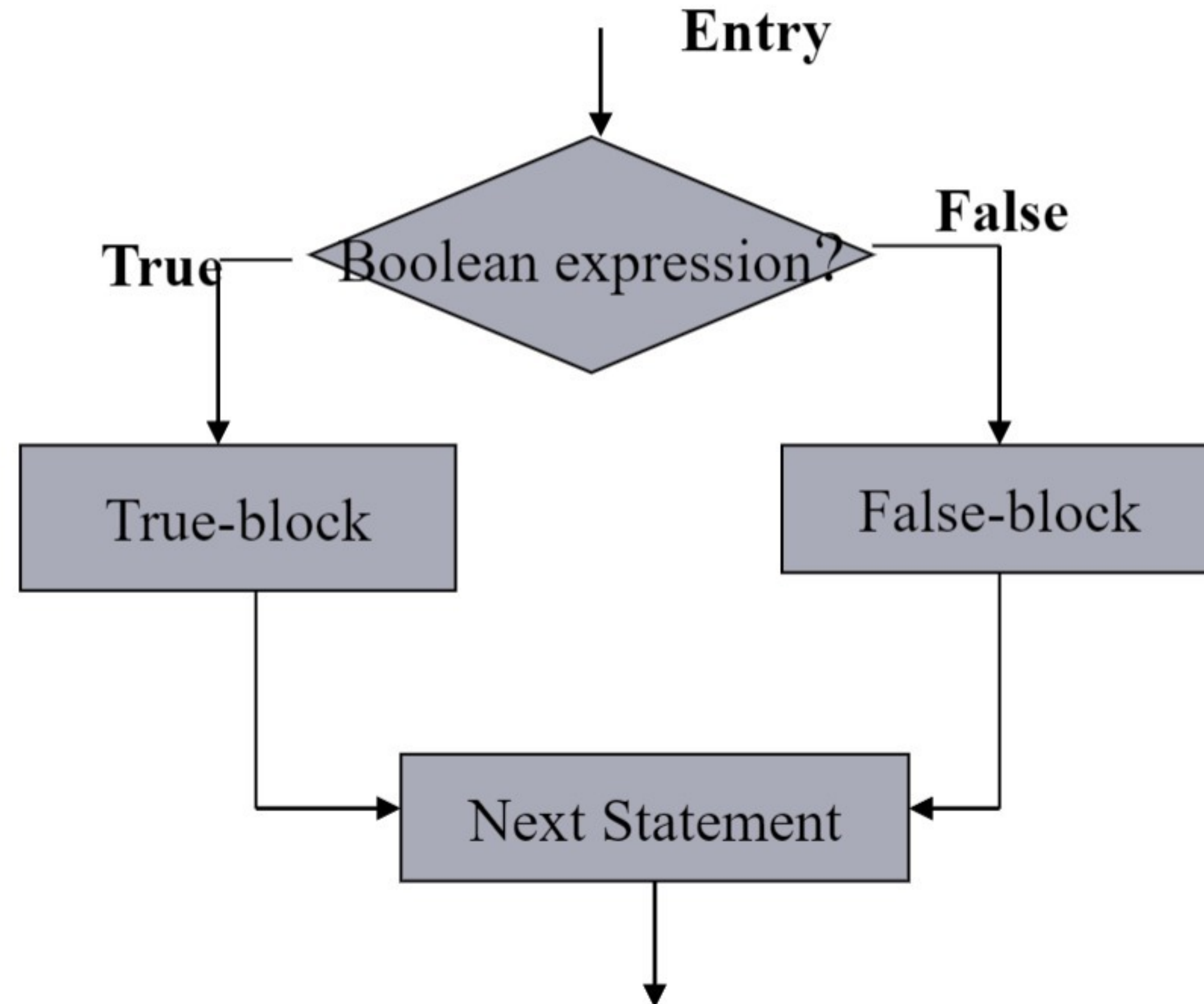
Output:

This is if statement

If..else statement

The if….else statement is an extension of simple if statement.

Syntax:

```
if(boolean expression)
  {
       True-block statements;
  }
 else
  {
       False-block statements;
  }
 Next statement;
```

If – else Example

/* program to check given age input is eligible to vote or not using if- else*/

```java
public class Check {
    public static void main(String[ ] args) {
        int age;
        age = Integer.parseInt(args[0]);
        if(age>18) {
            System.out.println("Eligible to vote");
        }
        else {
            System.out.println("Not eligible to vote");
        }
    }
}
```

Cascading if- else

- **Syntax:**

```
if (condition1)  {
    statement-1
}

….
else if(conditio-n) {
    statement-n
}
else {
    default statement
}
next statement
```

if - else if Example

```java
/* program to print seasons for a month input using if & else if */
public class ElseIfDemo {
    public static void main(String[] args) {
        int month = Integer.parseInt(args[0]);
        if(month == 12 || month == 1 || month == 2)
            System.out.println("Winter");
        else if(month == 3 || month == 4 || month == 5)
            System.out.println("Spring");
        else if(month == 6 || month == 7 || month == 8)
            System.out.println("Summer");
        else if(month == 9 || month == 10  || month == 11)
            System.out.println("Autumn");
        else
            System.out.println("invalid month");
    }
}
```

If args[0] is 6 then the Output is : Summer

Switch Case

- The switch-case conditional construct is a more structured way of testing for multiple conditions rather than resorting to a multiple if statement.

Syntax:

```
switch (expression)          {
            case value-1 : case-1 block
                                    break;
            case value-2 : case-2 block
                                    break;
            default   : default block
                                    break;
        }
statement-x;
```

Switch Case - Example

```java
/* This is an example of a switch case statement*/

public class SwitchDemo {
    public static void main(String[] args) {
        int weekday = Integer.parseInt(args[0]);
        switch(weekday) {
            case 1:    System.out.println("Sunday");  break;
            case 2:    System.out.println("Monday"); break;
            case 3:    System.out.println("Tuesday"); break;
            case 4:    System.out.println("Wednesday"); break;
            case 5:  System.out.println("Thursday");  break;
            case 6:    System.out.println("Friday");  break;
            case 7:    System.out.println("Saturday");  break;
            default:  System.out.println("Invalid day");
        }
    }
}
```
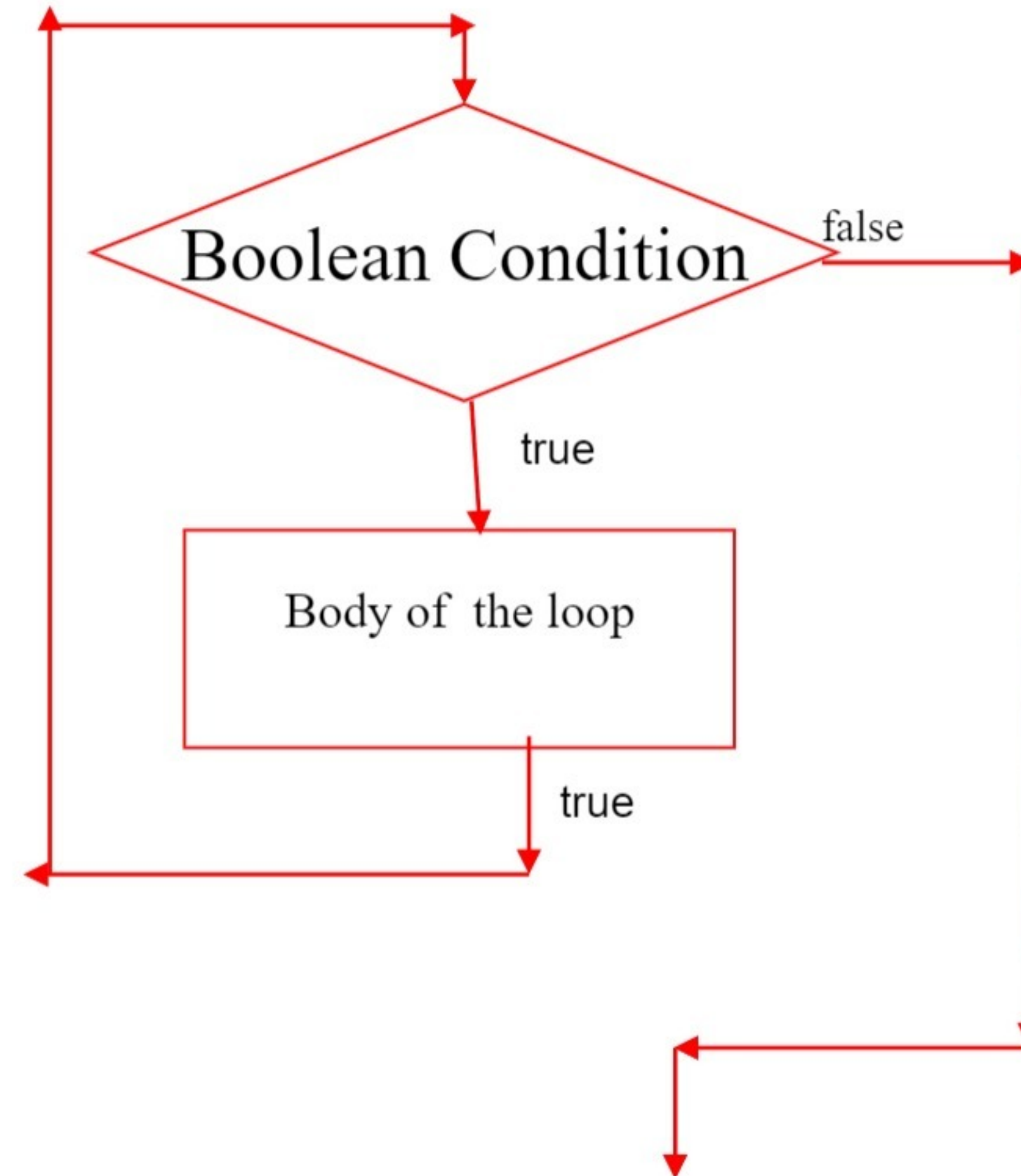
<u>If args[0] is 6 then the Output is :</u> Friday

While loop

- **Syntax**

```
while(condition)
{
    Body of the loop
}
```


Boolean Condition → false / true

Body of the loop

while loop – Example

```java
/* This is an example for a while loop */

public class Sample{
    public static void main(String[] args) {
        int i = 0;
        while (i < 5) {
            System.out.println("i: "+i);
            i = i + 1;
        }
    }
}
```
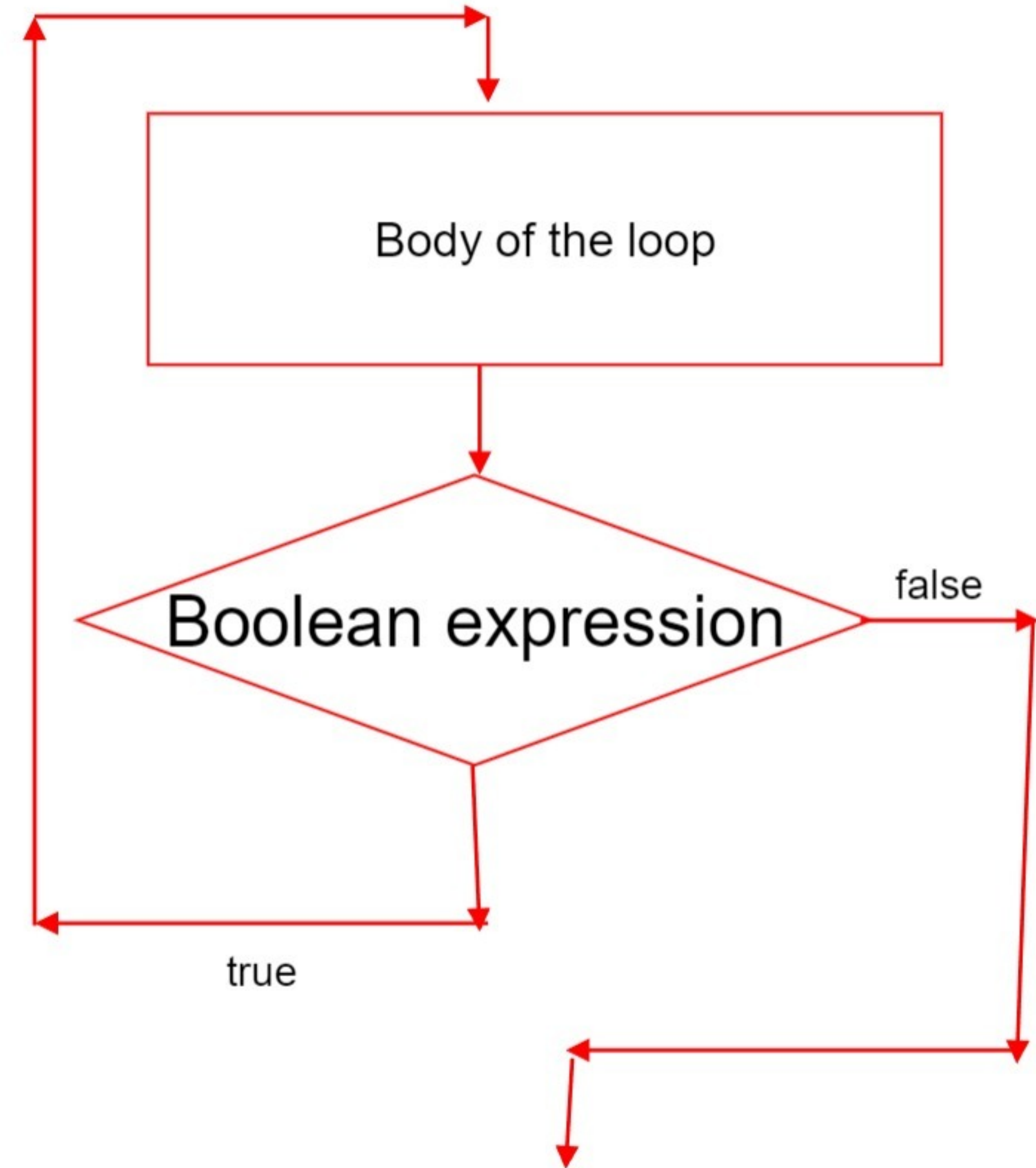
Output:
i: 0
i: 1
i: 2
i: 3
i: 4

do-while loop

- **Syntax:**

```
do
{
    Body of the loop
} while(boolean expression);
```

Body of the loop

Boolean expression

false

true

do...while loop – Example

```java
/* This is an example of a do-while loop */

public class Sample {
    public static void main(String[] args) {
        int i =5;
        do {
            System.out.println("i: "+i);
            i = i + 1;
        } while (i < 5);
    }
}
```

Output:
i: 5

for loop

Syntax

```
for(initialization; condition; increment/decrement)
{
    Body of the loop
}
```

for loop - Example

```
/* This is an example of a for loop */
public class Sample {
    public static void main(String[] args) {
        for (int  i=1; i<=5; i++ ) {
            System.out.println("i: "+i);
        }
    }
}
```

Output:
i: 1
i: 2
i: 3
i: 4
i: 5

Enhanced for loop

Syntax:

```
for(declaration : expression)
 {
    Body of loop
 }
```

Enhanced for loop - Example

```java
/* This is an example of a enhanced for loop */


public class Sample {

    public static void main(String[] args) {

        int [] numbers = {10, 20, 30, 40, 50};

        for(int i : numbers ) {

            System.out.println( "i: "+i );

        }

    }

}
```

Output:
i:10
i:20
i: 30
i:40
i:50

break statement

- While the execution of program, the break statement will terminate the iteration or switch case block.

- When a break statement is encountered in a loop, the loop is exited and the program continues with the statements immediately following the loop.

- When the loops are nested, the break will only terminate the corresponding loop body.

break - Example

```
/* This is an example of a break statement */


public class Sample{
    public static void main(String[] args) {
        for (int  i=1; i<=5; i++ ) {
            if(i==2)
                break;
            System.out.println("i: "+i);
        }
    }
}
```

Output:
i: 1

continue statement

- The continue statement skips the current iteration of a loop.

- In while and do loops, continue causes the control to go directly to the test-condition and then continue the iteration process.

- In case of for loop, the increment section of the loop is executed before the test-condition is evaluated.

continue - Example

```java
/* This is an example of a continue loop */

public class Sample {
  public static void main(String[] args) {
    int [] numbers = {1, 2, 3, 4, 5};
    for(int i : numbers ) {
      if( i == 3 ) {
        continue;
      }
      System.out.println( "i: "+i );
    }
  }
}
```

Output:
i: 1
i:2
i:4
i:5

Good Programming Practices

if statement

➢ Always use {} for if statements

➢ Avoid the following error prone

if (condition)   //ERROR

    statement;

Number per Line

➢ One declaration per line is recommended

```
int height;
int width;
```

Do not put different types on the same line

```
int height,width[];      //WRONG
```

Quiz

1. What will be the result, if we try to compile and execute the following code ?

```java
class Sample{
  public static void main(String[]args) {
        boolean b = true;
     if(b){
         System.out.println(" if block ");
     }
     else {
         System.out.println(" else block ");
     }
  }
}
```

Quiz (Contd..)

2. **What will be the result, if we try to compile and execute the following code snippets :**

```
a. class Sample {

      public static void main(String[] args) {

            while(false)

                  System.out.println("while loop");

      }

  }

b. class Sample {

      public static void main(String[] args) {

            for( ; ; )

                  System.out.println("For loop");

      }

  }
```

Summary

In this session, you were able to :

- Learn the various Flow control statements.

Thank You