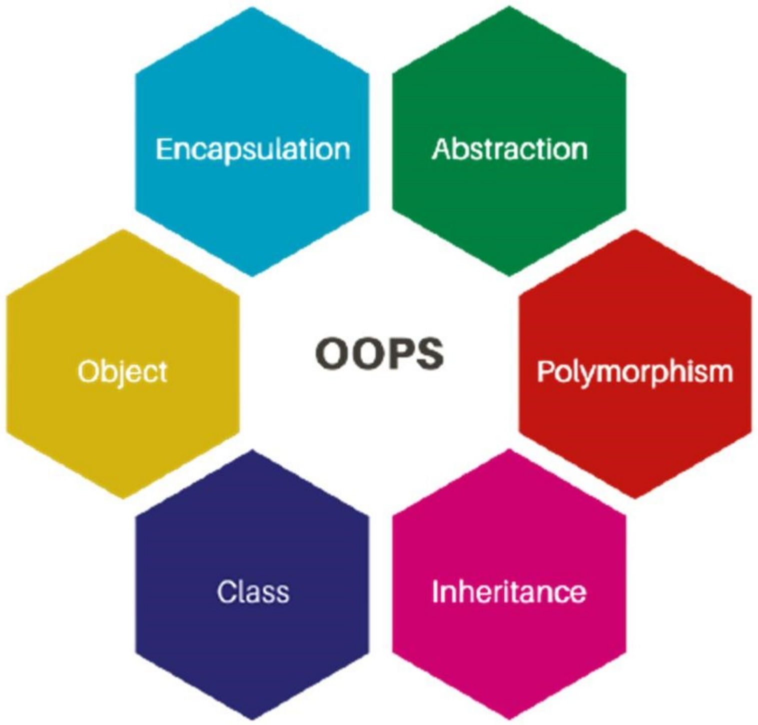


ABESIT COLLEGE OF ENGINEERING

CRC-Training

Core Java

CRC-Training

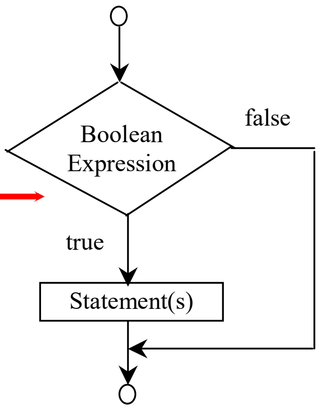
Flow Control or (Branching) or (Selection)

- One-way `if` Statements
- It is used to decide whether block of statements are executed or not based on the <condition>.
- **Syntax:**

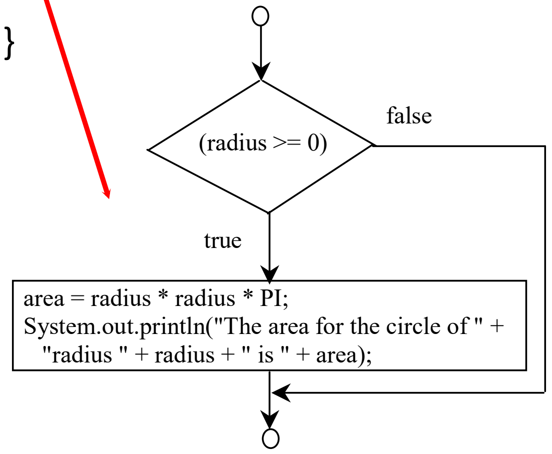
```
if (boolean-expression)
{
        statement(s);
}
```

Example:

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("The area" + " for the circle
     of radius " + radius + " is " + area);
}
```



(A)



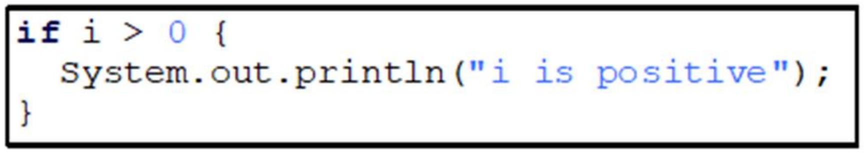
```
area = radius * radius * PI;
System.out.println("The area for the circle of " +
  "radius " + radius + " is " + area);
```

(B)

CRC-Training

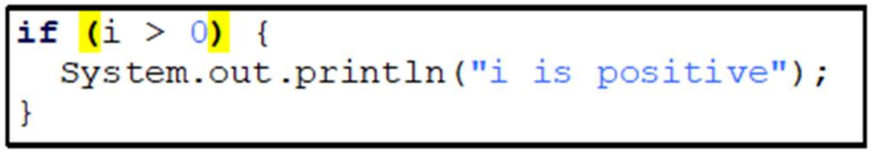
Flow Control or (Branching) or (Selection)

- The <condition> must be **boolean value** but not numeric number.
- The if block will be executed only if the <condition > is true.



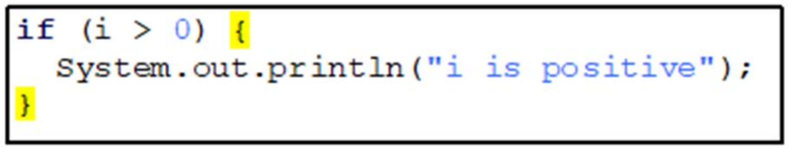
```
if i > 0 {
  System.out.println("i is positive");
}
```

(a) Wrong



```
if (i > 0) {
  System.out.println("i is positive");
}
```

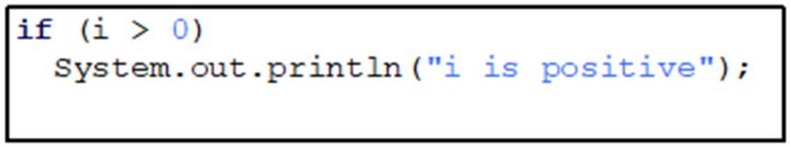
(b) Correct



```
if (i > 0) {
  System.out.println("i is positive");
}
```

(a)

Equivalent



```
if (i > 0)
  System.out.println("i is positive");
```

(b)

ABESIT COLLEGE OF ENGINEERING

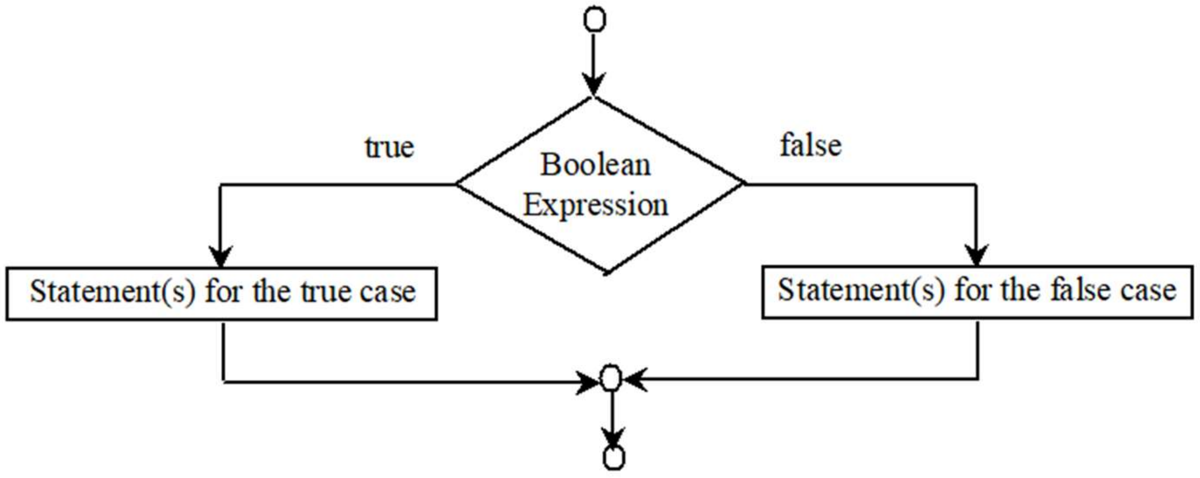
CRC-Training

Flow Control or (Branching) or (Selection)

- The Two-way `if` Statement (**if-else**)
  - The if-else statement is used to decide between two actions.

Syntax:

```
 if (boolean-expression) {
  statement(s)-for-the-true-case;
}
else {
  statement(s)-for-the-false-case;
}
```


true
Boolean Expression
false
Statement(s) for the true case
Statement(s) for the false case

- If the <condition > is evaluated to true then **if** block is executed, otherwise **else** block is executed.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Flow Control or (Branching) or (Selection)

- The Two-way `if` Statement (**if-else**)

Example

```
int age = scan.nextInt();

if(age < 18){
System.out.println(“Your age does not permit to login to our website.”);

}

else{

SOP(“Logged into Inbox”);

}
```

Flow Control or (Branching) or (Selection)

- **else if ladder**
    - It is used to select one among many alternatives.

Syntax:

```
if(<condition1>) {
                <statement(s)>
 } else if(<conditon2>) {
                <statement(s)>
}
…
 else{
                <statement(s)>
}
```

The final **else** block is executed only if none of the above <condition>s are true.

Flow Control or (Branching) or (Selection)

❖ **else if ladder**

Example:

```
if(false) { }

else if(false) { }

else if(false) {

 } else {          //  else block is executed since all above if() blocks are evaluated to false.


 }
```

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Flow Control or (Branching) or (Selection)

- Note
  - The else clause matches the most recent if clause in the same block.

```java
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    System.out.println("A");
else
    System.out.println("B");
```

(a)

Equivalent

```java
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    System.out.println("A");
  else
    System.out.println("B");
```

(b)

CRC-Training

Flow Control or (Branching) or (Selection)

- Note
  - Nothing is printed from the preceding statement.
  - To force the else clause to match the first if clause, you must add a pair of braces:

```
 int i = 1;
 int j = 2;
 int k = 3;
 if (i > j){
   if (i > k)
     System.out.println("A");
}else
   System.out.println("B");
```

- Now this statement prints B.

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Flow Control or (Branching) or (Selection)

❖Common Errors

❖Adding a semicolon at the end of an if clause is a common mistake.

```
if (radius >= 0); ←—— Wrong
{
  area = radius * radius * PI;
  System.out.println("The area for the circle of radius " + radius + " is " + area);
}
```

❖This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Flow Control or (Branching) or (Selection)

- **switch**
  - Switch statement is used to choose one among many alternative actions.

Syntax:

```
switch(<switch expression>){
        case label 1: <statement(s)>
        case label 2: <statement(s)>
        …
        case label n: <statement(s)>
        default: <statement(s)>
}
```

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Flow Control or (Branching) or (Selection)

- **switch**

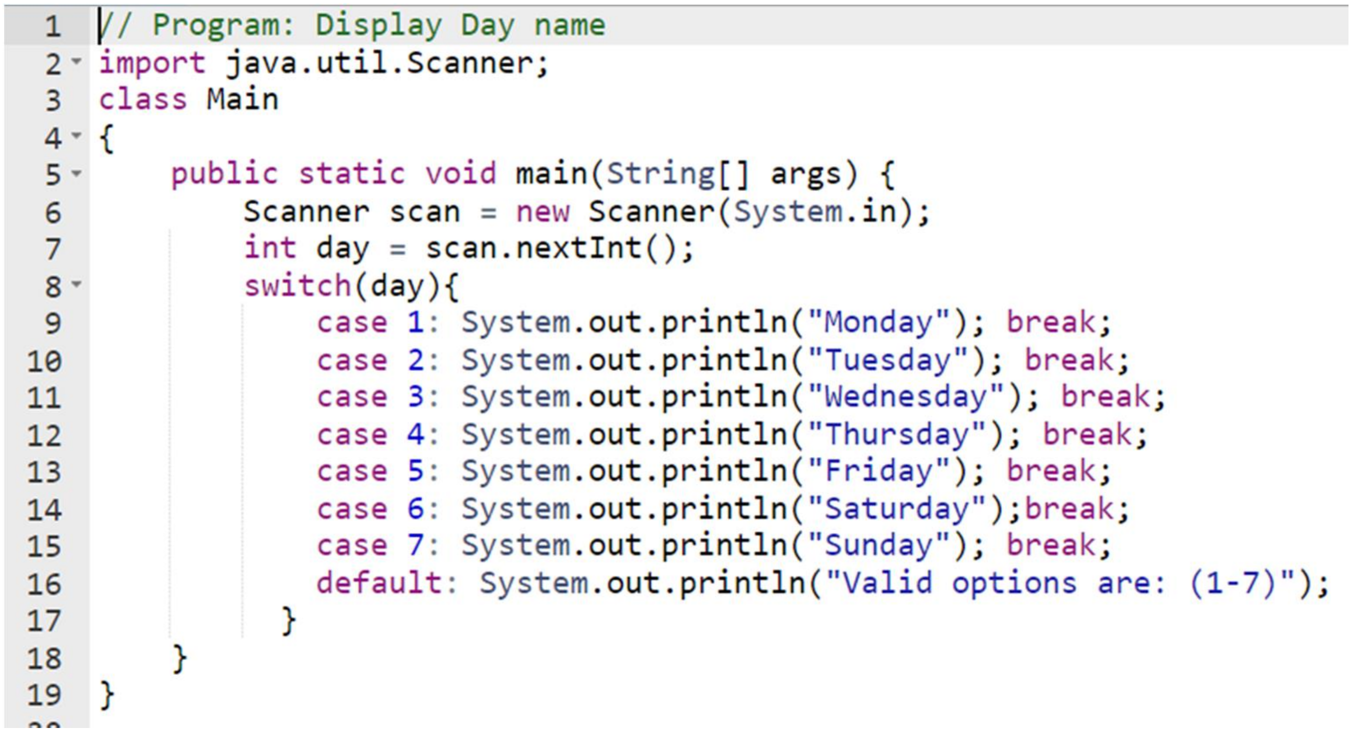
- The valid <switch expression> types are:
  - byte, short, int, char, but not long
  - Byte, Short, Integer, Character, but not Long

- The valid case label types are:
  - byte, short, int, char values but not long value, but not any wrapper classes.

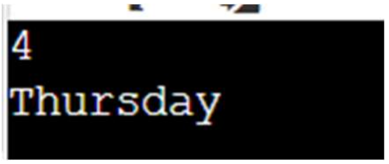
CRC-Training

Flow Control or (Branching) or (Selection)

- switch



```
// Program: Display Day name
import java.util.Scanner;
class Main
{
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int day = scan.nextInt();
        switch(day){
            case 1: System.out.println("Monday"); break;
            case 2: System.out.println("Tuesday"); break;
            case 3: System.out.println("Wednesday"); break;
            case 4: System.out.println("Thursday"); break;
            case 5: System.out.println("Friday"); break;
            case 6: System.out.println("Saturday");break;
            case 7: System.out.println("Sunday"); break;
            default: System.out.println("Valid options are: (1-7)");
        }
    }
}
```



```
4
Thursday
```

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Math Package

The `Math` Class

- **Class constants:**
  - `PI`
  - `E`
- **Class methods:**
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

CRC-Training

Math Package

- `min(a, b)`
  - Returns the minimum of two parameters.
- `max(a, b)`
  - Returns the maximum of two parameters.
- `int round(float x)`
  - Returns the int value if the argument is float.
- `long round(double x)`
  - Returns the long value if the argument is double.

Examples:

```
Math.max(2, 3) returns 3
Math.max(2.5, 3) returns 3.0
Math.min(2.5, 3.6) returns 2.5

Math.round(2.6f) returns 3
Math.round(2.0) returns 2
Math.round(-2.0f) returns -2
Math.round(-2.6) returns -3
```

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Math Package

- `double ceil(double x)`
  - x rounded up to its nearest integer. This integer is returned as a double value.
- `double floor(double x)`
  - x is rounded down to its nearest integer. This integer is returned as a double value.

```
Examples:

Math.ceil(2.1) returns 3.0
Math.ceil(2.0) returns 2.0
Math.ceil(-2.0) returns -2.0
Math.ceil(-2.1) returns -2.0

Math.floor(2.1) returns 2.0
Math.floor(2.0) returns 2.0
Math.floor(-2.0) returns -2.0
Math.floor(-2.1) returns -3.0
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Math Package

- `sqrt(double a)`
  - Returns the square root of `a`.
- `pow(double a, double b)`
  - Returns `a` raised to the power of `b`.
- `random()`
  - Returns a random `double` value in the range [0.0, 1.0).

```
Examples:

Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 3.24

Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(3.5, 2.5) returns 22.91765
```

CRC-Training

Math Package

❖ The random Method

- ➢ Generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() < 1.0).

➢ Examples:

`(int)(Math.random() * 10)` $\longrightarrow$ Returns a random integer between 0 and 9.

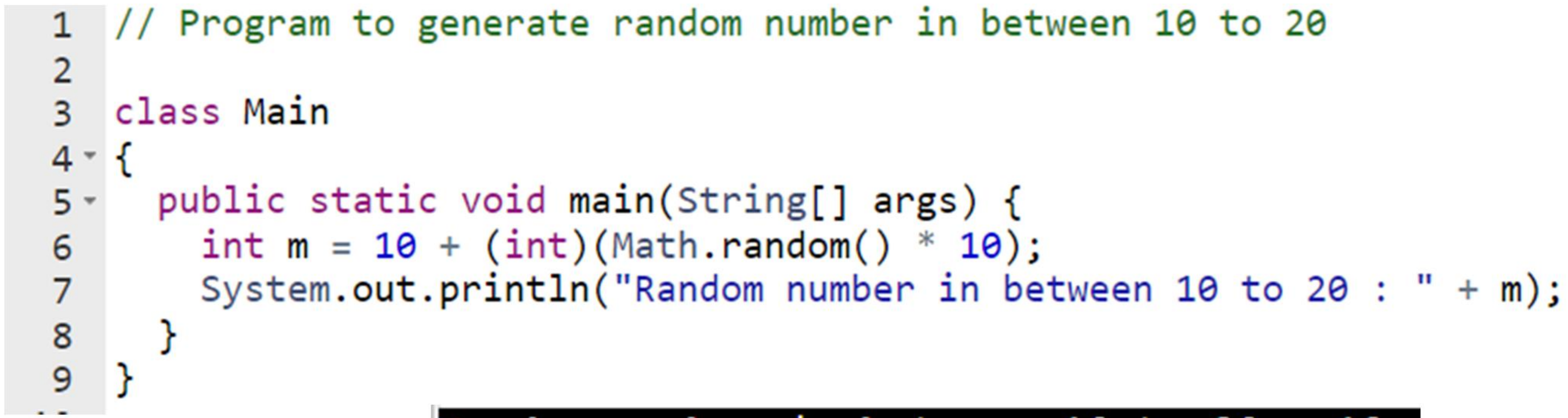
`50 + (int)(Math.random() * 50)` $\longrightarrow$ Returns a random integer between 50 and 99.

➢ In general,

`a + Math.random() * b` $\longrightarrow$ Returns a random number between a and a + b, excluding a + b.

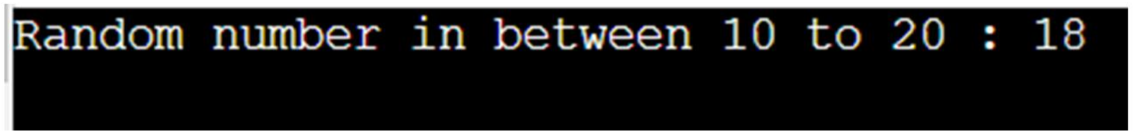
CRC-Training

Math Package



```java
// Program to generate random number in between 10 to 20

class Main
{
  public static void main(String[] args) {
    int m = 10 + (int)(Math.random() * 10);
    System.out.println("Random number in between 10 to 20 : " + m);
  }
}
```



```
Random number in between 10 to 20 : 18
```