ABESIT
COLLEGE OF ENGINEERING

CRC-Training

Java™

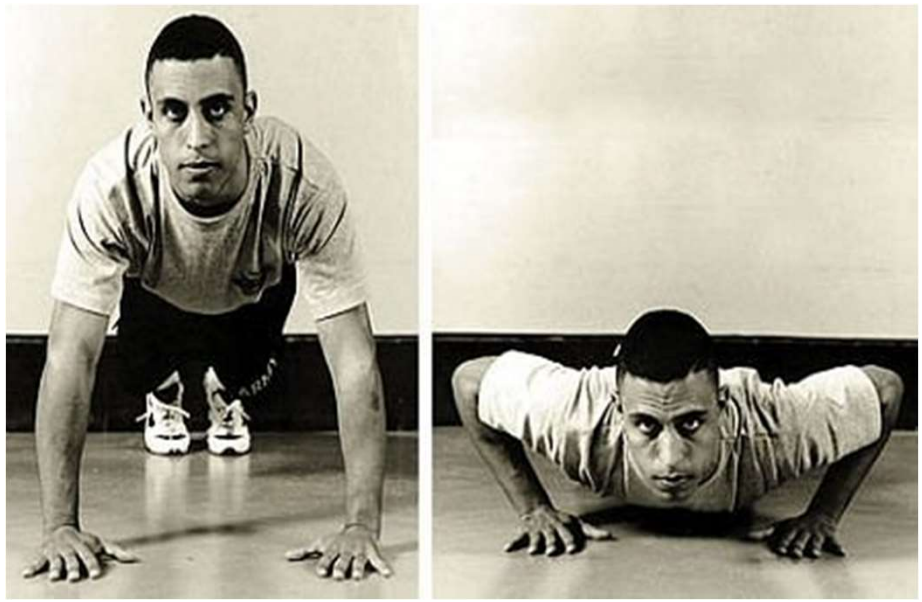
Core Java

Loops (Iteration Statements)

- When you want to repeat something for definite number of times, it is called loop.

or

- The iterate statements are those which used to execute block of statements repeatedly until the boolean <condition> becomes false.



Loops (Iteration Statements)

- **Java has three basic loops**
  - while
  - do-while
  - for

Note: In JDK 1.5 java introduce one more loop called ***Enhance for loop***.

Loops (Iteration Statements)

❖while loop

- In case of while loop, the <condition> will be evaluated before executing body.

  Syntax:

```
while (<condition>){
        < body>
}
```

- The <condition> is always boolean value.

Loops (Iteration Statements)

- **while loop**

```
1  // Program to print numbers from 1 to 10
2
3  class Main
4  {
5    public static void main(String[] args) {
6        int I = 1;
7
8        while(I <= 10){
9            System.out.print(I + "\t");
10           I++;
11       }
12   }
13 }
```

```
1   2   3   4   5   6   7   8   9   10
```

Loops (Iteration Statements)

❖do - while loop

- In case of do-while, the body is executed before <condition> is evaluated i.e., the body will be executed atleast once.

Syntax:

```
do{
        < body>
} while (<condition>);
```

*//**Watch:** Semicolon at the end.*

Loops (Iteration Statements)

❖ **do - while loop**

```
1  // Program to add and print numbers from 1 to 10
2
3  class Main
4  {
5    public static void main(String[] args) {
6        int i = 1;
7        int sum = 0;
8        do{
9            sum += i;
10           i++;
11         }while(i<=10);
12         System.out.println("sum = " + sum);
13   }
14 }
```

```
sum = 55
```

Loops (Iteration Statements)

❖ **Difference between while and do-while**

while	do - while
➢ The <condition> is evaluated before executing the body. ➢ Hence, while loop executes zero or more times.	➢ The body will be executed before the <condition> is evaluated. ➢ Hence, the body would be executed one or more times i.e., body will be executed at least once.

Loops (Iteration Statements)

❖ **Basic for loop (Simple or General for loop)**

- This loop is best suitable if we know the number of iterations in advance. **For example**, reading array elements or reading collection elements.

Syntax:

```
for (<initialization>; <condition>; <increment or decrement>){
        <body>
}
```

Loops (Iteration Statements)

- **Basic for loop (Simple or General for loop)**

```
1  // Program to add array elements using for loop
2
3  class Main
4  {
5    public static void main(String[] args) {
6        int[] arr = {10,20,30,40};
7        int sum = 0;
8
9        for(int i=0; i < arr.length; i++){
10           sum += arr[i];
11       }
12
13       System.out.print("The total = "+sum);
14   }
15 }
```

```
The total = 100
```

Transfer Statements

❖ **break**

- The '**break**' is used to come out of the loops.
- We can use break statement in one of the following cases:
  - Inside loops
  - Inside switch

Example:

```
for(int i=0 ; i<10; i++){
 for(int j=0; j<10; j++){
     if(i == j) break;  //the nearest for loop will be terminated.
  }
}
```

Transfer Statements

❖ **continue**

- The **'continue'** statement is used to transfer the execution back to the start of the loop i.e., the remaining statements after continue will be skipped.

Program:

```
1  // Program to add even numbers but print odd numbers
2
3  class Main
4  {
5    public static void main(String[] args) {
6        int sum = 0;
7
8        for(int i = 0; i<=10; i++ ){
9            if(i%2 == 0){ //even number
10               sum += i;  //Adding even numbers
11           continue;
12           }
13           System.out.print (i+"\t"); //Printing odd numbers
14       }
15        System.out.println("\nThe sum of even numbers is : "+sum);
16   }
17 }
```

```
1       3       5       7       9
The sum of even numbers is : 30
```

Transfer Statements

❖ **return**

- The **'return'** statement is always used from method block. The return statement is used to stop execution of a method and transfer control back to the method calling.
- There are two forms of the return statement:
  - **return;** //Optionally used with methods whose return type is void or from constructor.
  - **return <expression>;** //Must be used with methods whose return type is other than void.
- **A void method need not have a return statement** – in which case the control normally returns to the caller after the last statement in the method's body has been executed.
- However, **a void method can optionally specify the first form of the return statement** – in which case the control immediately transferred to the method caller.

Transfer Statements

- return

Program to return maximum number

```
int max(int x, int y)
{
    if(x > y)
        return x;
    else
        return y;
}
```

or

```
int max(int x, int y)
{
  return x>y?x:y; //ternary operator
}
```