# JUMP Statements

# break statement

- Forces the termination of a loop
- When encountered in a loop,
  - the loop terminates immediately and the execution resumes with the next statement following the loop
- If used in a nested loop, quits only the loop from where it is called

# Example:

```
 1class breakTest
 2{
 3    public static void main(String args[])
 4    {
 5        for (int j = 0; j < 5; j++)
 6        {
 7            // come out of loop when i is 4.
 8            if (j == 4)
 9                break;
10            System.out.println(j);
11        }
12        System.out.println("After loop");
13    }
14}
```

# continue statement

- 'continue' statement forces the next iteration of the loop to take place and skips the code between 'continue' statement and the end of the loop

- In case of for loop, 'continue' makes the execution of the increment portion of the statement and then evaluates the conditional part

- In case of while and do-while loops, 'continue' makes the conditional statement to be executed

- Continue statement works same as break but the difference is it only comes out of loop for that iteration and continue to execute the code for next iterations. So it only bypasses the current iteration.

```
1class continueTest
2{
3   public static void main(String args[])
4   {
5       for (int j = 0; j < 10; j++)
6       {
7           // If the number is odd then bypass and continue with next value
8           if (j%2 != 0)
9               continue;
10          // only even numbers will be printed
11          System.out.print(j + " ");
12      }
13  }
14}
```

# Continue : Control flow

# Labeled break and continue statement

```
Line 0:          mylabel:   for(i=0;i<2;i++)
                 {
Line 1:                     for(j=0;j<5;j++)
                    {
Line 2:                          if(i==2)
Line 3:                     break  mylabel;
Line 4:                          Statement 1;
Line 5:                          Statement 2;
                    }
Line 6 :                 Statement 3;
              }
Line 7:   Statement 4;
```

# Labeled break and continue statement

```
Line 0:          mylabel:   for(i=0;i<2;i++)
                {
Line 1:                    for(j=0;j<5;j++)
                   {
Line 2:                              if(i==2)
Line 3:            continue   mylabel;
Line 4:                              Statement 1;
Line 5:                              Statement 2;
                   }
Line 6 :                 Statement 3;
              }
Line 7:   Statement 4;
```

# Return Statement

- **Return**
- Return statement is used to transfer the control back to calling method. Compiler will always bypass any sentences after return statement. So, it must be at the end of any method. They can also return a value to the calling method.

Example:

```
class prg
{
  public static void main(String args[])  {
          int  a=2,n=3,s=0;
          s = add(a,n)
         System.out.println("s="+s)        }

  static  int add(int b, int n)  {
    int res= b+n;
    return  res;    }
}
```

# Reading Input from Keyboard
# Arrays
# Conditional and Loop Constructs

# Reading input from keyboard

**Declaration:**

```
int a;

Scanner sc = new Scanner(System.in);
 a = sc.nextInt();
```

**Inbuilt Class to be used belongs to the java.util package:**

```
import java.util.Scanner;
```

```
import java.util.Scanner;
class prg3
{       public static void main(String args[]){
                int a,b,c;
                 Scanner sc = new Scanner(System.in);
                System.out.println("Enter a first number");
                 a = sc.nextInt();
                 System.out.println("Enter a second number");
                 b = sc.nextInt();
                 c = a+ b;
                 System.out.println("sum is :"+c);
        }   }
```

# Scanner class methods:

| Method | Description |
|---|---|
| nextBoolean() | Reads a boolean value from the user |
| nextByte() | Reads a byte value from the user |
| nextDouble() | Reads a double value from the user |
| nextFloat() | Reads a float value from the user |
| nextInt() | Reads an int value from the user |
| nextLine() | Reads a String value from the user |
| nextLong() | Reads a long value from the user |
| nextShort() | Reads a short value from the user |

```
1import java.util.Scanner;
2
3public class Example {
4public static void main(String[] args) {
5Scanner s = new Scanner(System.in);
6
7System.out.println("Enter name, age and salary");
8
9// String input
10String name = s.nextLine();
11
12// Numerical input
13int age = s.nextInt();
14double salary = s.nextDouble();
15
16// Output input by user
17System.out.println("Name: "+ name);
18System.out.println("Age: "+ age);
19System.out.println("Salary: "+ salary);
20}
21}
```