[0] Reserved words

| | | | |
|---|---|---|---|
| abstract | assert | boolean | break |
| byte | case | catch | char |
| class | const | continue | default |
| do | double | else | enum |
| extends | final | finally | float |
| for | goto | if | implements |
| import | instanceof | int | interface |
| long | native | new | package |
| private | protected | public | return |
| short | static | strictfp | super |
| switch | synchronized | this | throw |
| throws | transient | try | void |
| volatile | while | | |

[1] Very basic example

```
public class MyFirstJavaProgram {

   /* This is my first java program.
    * This will print 'Hello World' as the output
    */

   public static void main(String []args) {
      System.out.println("Hello World"); // prints Hello World
   }
}
```

[2] Enum example

```
class FreshJuice {
   enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
   FreshJuiceSize size;
}
```

```
public class FreshJuiceTest {

   public static void main(String args[]) {
      FreshJuice juice = new FreshJuice();
      juice.size = FreshJuice.FreshJuiceSize.MEDIUM ;
      System.out.println("Size: " + juice.size);
   }
}
```

[3] Class and methods example

```
public class Dog {
   String breed;
   int age;
   String color;

   void barking() {
   }

   void hungry() {
   }

   void sleeping() {
   }
}
```

[4] Creating an object

```
public class Puppy {
   public Puppy(String name) {
      // This constructor has one parameter, name.
      System.out.println("Passed Name is :" + name );
   }

   public static void main(String []args) {
      // Following statement would create an object myPuppy
      Puppy myPuppy = new Puppy( "tommy" );
   }
}
```

[5] Instance variables

```
public class Puppy {
   int puppyAge;

   public Puppy(String name) {
      // This constructor has one parameter, name.
      System.out.println("Name chosen is :" + name );
   }

   public void setAge( int age ) {
      puppyAge = age;
   }

   public int getAge( ) {
      System.out.println("Puppy's age is :" + puppyAge );
      return puppyAge;
   }

   public static void main(String []args) {
      /* Object creation */
      Puppy myPuppy = new Puppy( "tommy" );

      /* Call class method to set puppy's age */
      myPuppy.setAge( 2 );

      /* Call another class method to get puppy's age */
      myPuppy.getAge( );

      /* You can access instance variable as follows as well */
      System.out.println("Variable Value :" + myPuppy.puppyAge );
   }
}
```

[6] Employee study (employee class)

```
import java.io.*;
public class Employee {

   String name;
```

```
   int age;
   String designation;
   double salary;

   // This is the constructor of the class Employee
   public Employee(String name) {
      this.name = name;
   }

   // Assign the age of the Employee  to the variable age.
   public void empAge(int empAge) {
      age = empAge;
   }

   /* Assign the designation to the variable designation.*/
   public void empDesignation(String empDesig) {
      designation = empDesig;
   }

   /* Assign the salary to the variable salary.*/
   public void empSalary(double empSalary) {
      salary = empSalary;
   }

   /* Print the Employee details */
   public void printEmployee() {
      System.out.println("Name:"+ name );
      System.out.println("Age:" + age );
      System.out.println("Designation:" + designation );
      System.out.println("Salary:" + salary);
   }
}
```

## [7] Employee study (main class)

```
import java.io.*;
public class EmployeeTest {

   public static void main(String args[]) {
```

```
      /* Create two objects using constructor */
      Employee empOne = new Employee("James Smith");
      Employee empTwo = new Employee("Mary Anne");

      // Invoking methods for each object created
      empOne.empAge(26);
      empOne.empDesignation("Senior Software Engineer");
      empOne.empSalary(1000);
      empOne.printEmployee();

      empTwo.empAge(21);
      empTwo.empDesignation("Software Engineer");
      empTwo.empSalary(500);
      empTwo.printEmployee();
   }
}
```

## [8] Loop example

```
public class Test {

   public static void main(String args[]) {
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ) {
         System.out.print( x );
         System.out.print(",");
      }
      System.out.print("\n");
      String [] names = {"James", "Larry", "Tom", "Lacy"};

      for( String name : names ) {
         System.out.print( name );
         System.out.print(",");
      }
   }
}
```

[9] Number methods

| N. | Method & Description |
|---|---|
| 1 | xxxValue()<br>Converts the value of ‘this’ Number object to the xxx data type and returns it. |
| 2 | compareTo()<br>Compares ‘this’ Number object to the argument. |
| 3 | equals()<br>Determines whether ‘this’ number object is equal to the argument. |
| 4 | valueOf()<br>Returns an Integer object holding the value of the specified primitive. |
| 5 | toString()<br>Returns a String object representing the value of a specified int or Integer. |
| 6 | parseInt()<br>This method is used to get the primitive data type of a certain String. |
| 7 | abs()<br>Returns the absolute value of the argument. |
| 8 | ceil()<br>Returns the smallest integer that is greater than or equal to the argument. Returned as a double. |
| 9 | floor()<br>Returns the largest integer that is less than or equal to the argument. Returned as a double. |
| 10 | rint()<br>Returns the integer that is closest in value to the argument. Returned as a double. |
| 11 | round()<br>Returns the closest long or int, as indicated by the method's return type to the argument. |
| 12 | min()<br>Returns the smaller of the two arguments. |
| 13 | max()<br>Returns the larger of the two arguments. |
| 14 | exp()<br>Returns the base of the natural logarithms, e, to the power of the argument. |
| 15 | log()<br>Returns the natural logarithm of the argument. |
| 16 | pow()<br>Returns the value of the first argument raised to the power of the second argument. |
| 17 | sqrt() |

| | |
|---|---|
| | Returns the square root of the argument. |
| 18 | sin()<br>Returns the sine of the specified double value. |
| 19 | cos()<br>Returns the cosine of the specified double value. |
| 20 | tan()<br>Returns the tangent of the specified double value. |
| 21 | asin()<br>Returns the arcsine of the specified double value. |
| 22 | acos()<br>Returns the arccosine of the specified double value. |
| 23 | atan()<br>Returns the arctangent of the specified double value. |
| 24 | atan2()<br>Converts rectangular coordinates (x, y) to polar coordinate (r, theta) and returns theta. |
| 25 | toDegrees()<br>Converts the argument to degrees. |
| 26 | toRadians()<br>Converts the argument to radians. |
| 27 | random()<br>Returns a random number. |