HelloWorld

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Greetings");
    }
}
```

DataTypes

```
public class DataTypeTest
{

 int dataInt = 30;
 static int numStat = 15;

 public int getCount()
 {
   int count = 20;
   return count;
 }

}
```

Primitive DataTypes

Primitive Data Type	Size
int	4 bytes
short	2 bytes
long	8 bytes
float	4 bytes
double	8 bytes
byte	1 byte
char	2 bytes
boolean	1 bit

Operators

Operator Type	Operators
Bitwise	^, &, |
Logical	&&, ||
Conditional	?:
Increment	++
Decrement	—
Arithmetic	+, -, /, *, %
Relational	<,>,<=, >=,==,!=
Access	.

If Else Statement

```
public class IfElseStatement
{
  public static void main(String args[])
  {
    float salary = 50000;
    if(salary >= 25000)
    {
      System.out.println("Eligible for Income Tax filing");
    }
   else
   {
     System.out.println("You are under the basic Income");
   }
  }
}
```

For Loop Control Flow

```
public class ForLoopControlFlow
{
  public static void main (String args[])
  {
    for(int i=0;i<=10;i++)
     System.out.println("printing "+i);
  }
}
```

Switch Statement

```
public class SwitchStatement
{
  public static void main(String args[])
  {
    int color = 0;
    switch(color)
    {
      case 0 :
      System.out.println("White");
      break;
      case 1 :
      System.out.println("Yellow");
      break;
      case 2 :
      System.out.println("Green");
      break;
      case 3 :
      System.out.println("Blue");
      break;
      default :
      System.out.println("Black");
    }
  }
}
```

While Control Flow

```
public class WhileLoopControlFlow
{
  public static void main(String args[])
  {
    int i = 3;
    while(i<=11)
    {
      System.out.println("Printing "+i);
      i++;
    }
  }
}
```

Do While Control Flow

```
public class DoWhileControlFlow
{
  public static void main(String args[])
  {
    int i = 4;
    do
    {
      System.out.println("printing"+i);
      i++;
    }
    while(i<=18);
  }
}
```

Break Statement

```
public class BreakExample
{
   public static void main(String args[])
   {
      int [] integers = {30, 60, 70, 87, 97};
      for(int i : integers ) {
         if( i == 70 ) {
            break;
         }
         System.out.println("printing "+ i );

      }
   }
}
```

Continue Statement

```
public class ContinueExample
{
   public static void main(String args[])
   {
      int [] integers = {13, 21, 54, 80, 90};
      for(int i : integers ) {
         if( i == 80 ) {
            continue;
         }
         System.out.println( "printing "+ i );

      }
   }
}
```

Arrays

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

    int [] integers = {2,4,10,5,7,9};

    for(int i=0;i< integers.length;i++)
    {
      System.out.print("Array element "+ integers[i]);
    }


    int product =1;
    for(int i=0;i<integers.length;i++)
    {
      product = product * integers[i];
    }
    System.out.println("The product of array elements is "+ product);
  }
}
```

Multi Dimensional Arrays

```
public class MultiDimensionArray
{
  public static void main(String args[])
  {
    int [][] multiArray1 = { {1,5,7}, {2,4,6}};
    int [][] multiArray2 = {{1,2,1},{4,4,3}};
    int [][] differenceArray = new int [2][3];

    for(int i=0;i< 2;i++)
    {
      for(int j=0;j< 3;j++)
      {
        System.out.print("Matrix element in multiArray1 "+multiArray1[i][j]);
      }

    }

    for(int i=0;i< 2;i++)
    {
      for(int j=0;j< 3;j++)
      {
        System.out.print("Matrix element in multiArray2 "+multiArray2[i][j]);
      }

    }

    for(int i=0;i< 2;i++)
    {
      for(int j=0;j< 3;j++)
      {
        differenceArray[i][j] = multiArray1[i][j] + multiArray2[i][j];
        System.out.print("difference Array element "+ differenceArray[i][j]);
      }
    }
  }
}
```

Annotations

```
class Shape
{
     public void display()
     {
          System.out.println("Shape display()");
     }
}
public class Rectangle extends Shape
{
     @Override
     public void display()
     {
          System.out.println("Rectangle display(int )");
     }

     public static void main(String args[])
     {
          Rectangle rect = new Rectangle();
          rect.display();
     }
}
```

Classes

```
public class Car {


    public String model;
    public String engineType;
    public int vehicleNum;


    public Car(String model,String engineType,int vehicleNum) {
        this.model = model;
        this.engineType = engineType;
        this.vehicleNum = vehicleNum;
    }


    public void setModel(String model) {
        this.model = model;
    }

    public void setEngineType(String engineType) {
        this.engineType = engineType;
    }

    public void setVehicleNum(int vehicleNum) {
        this.vehicleNum = vehicleNum;
    }

    public String getModel() {
        return this.model;
    }

    public String getEngineType() {
        return this.engineType;
    }

    public int getVehicleNum() {
        return this.vehicleNum;
    }

    public void printInfo() {

        System.out.println("Model " + getModel());
        System.out.println("engineType " + getEngineType());
        System.out.println("vehicleNum " + getVehicleNum());

    }

    public static void main(String[] args)
    {
        Car car = new Car("Toyota Tercel","Single Cylinder",2342334);

        car.printInfo();

        System.out.println("Changing the car properties");
        car.setModel("Honda Civic");
        car.setEngineType("Four Cylinder");
        car.setVehicleNum(454534434);

        car.printInfo();
    }
}
```

Objects

```
public class ObjectCreator {

    public static void main(String[] args) {

        Car car1 = new Car("Toyota Tercel","Single Cylinder",2342334);
        Car car2 = new Car("Ford Mustang","DOHC",2394434);

        car1.printInfo();
        car2.printInfo();

        System.out.println("Changing the car2 properties");
        car2.setModel("Chevorlet Bolt");
        car2.setEngineType("Four Cylinder");
        car2.setVehicleNum(2234234);

        car2.printInfo();
    }
```

Encapsulation

```
public class Employee {
   private String name;
   private String id;
   private int age;

   public Employee(String name, String id, int age)
   {
       this.name = name;
       this.id = id;
       this.age = age;
   }
   public int getAge() {
      return age;
   }

   public String getName() {
      return name;
   }

   public String getId() {
      return id;
   }

   public void setAge( int age) {
      this.age = age;
   }

   public void setName(String name) {
      this.name = name;
   }

    public void setId( String id) {
       this.id = id;
    }
}
```

```
public class EncapsulationExample {

   public static void main(String args[]) {
      Employee employee = new Employee();
      employee.setName("Steve Smith");
      employee.setAge(32);
      employee.setId("243243");

      System.out.println("Name : " + employee.getName()
   }
}
```

Interfaces

```
interface Machine
{
    int  velocity=50;
    public int getDistance();
}
interface Vehicle
{
    int distanceTravelled=100;
    public int getVelocity();
}
public class Truck implements Machine, Vehicle
{
    int time;
    int velocity;
    int distanceTravelled;

    public Truck(int velocity, int time)
    {
        this.velocity = velocity;
        this.time = time;
    }

    public int getDistance()
    {
        distanceTravelled= velocity*time;
        System.out.println("Total Distance  is : "+distanceTravelled);
        return distanceTravelled;
    }
    public int getVelocity()
    {
        int velocity=distanceTravelled/time;
        System.out.println("Velocity  is : " + velocity);
        return velocity;
    }
    public static void main(String args[])
    {
        Truck truck = new Truck(50,2);
        truck.getDistance();
        truck.getVelocity();
    }
}
```

Inheritance

```
public class SalariedEmployee extends Employee {
   private double empSalary;

   public SalariedEmployee(String name, String id, int age, double empSalary) {
      super(name, id, age);
      setEmpSalary(empSalary);
   }

   public double getEmpSalary() {
      return empSalary;
   }

   public void setEmpSalary(double empSalary) {
      if(empSalary >= 0.0) {
         this.empSalary = empSalary;
      }
   }

  public static void main(String[] args)
  {
      SalariedEmployee salaryEmp = new SalariedEmployee("Steve Smith", "Sanjose, CA",
      Employee emp = new SalariedEmployee("John Ray", "Dallas, TX", 43, 44000.00);

      System.out.println("Employee "+salaryEmp.getName()+" salary " +salaryEmp.getEmpSa

      System.out.println("Employee "+ emp.getName()+ " age "+ emp.getAge());
  }
}
```

Keywords

Utilization Category	Key Word
Class	class
Interface	implements
Class	abstract
Object	new
Class Type	static
Parent Class	super
Current Object	this
Constant set	enum
Exception	try
Exception	catch
Exception	throw
Exception	finally
Constant	final
Inheritance	extends

Java Code Geeks

JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

Polymorphism

Plane and Truck Class

```
interface Machine
{

    int distanceTravelled=100;
    public int getDistance();                    )
}
interface Vehicle
{
    int  velocity=50;
    public int getVelocity();
}
class Plane implements Machine, Vehicle hicle
{
    int time;
    int velocity;
    int distanceTravelled;
    {
        this.velocity = velocity;
        this.time = time;
    }

    public int getDistance()
    {
        distanceTravelled= velocity*time;
        System.out.println("Total Distance  is : "+distanceTravelled);
        return distanceTravelled;
    }
    public int getVelocity()
    {
        int velocity=distanceTravelled/time;
        System.out.println("Velocity  is : "+ velocity);
        return velocity;
    }
}
public class Truck implements Machine, Vehicle
{
    int time;
    int velocity;
    int distanceTravelled;


    public Truck(int velocity, int time)
    {
        this.velocity = velocity;
        this.time = time;
    }

    public int getDistance()
    {
        distanceTravelled= velocity*time;
        System.out.println("Total Distance  is : "+distanceTravelled);
        return distanceTravelled;
    }
    public int getVelocity()
    {
       int velocity=distanceTravelled/time;
        System.out.println("Velocity  is : "+ velocity);
        return velocity;
    }

}
```

File Operations

```
import java.io.File;
import java.io.FileWriter;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;

class FileOperations {
    public static void main(String[] args) {
        try {

            File file = new File("input.txt");
            Scanner dataReader = new Scanner(file);

            FileWriter fwrite = new FileWriter("output.txt");
            while (dataReader.hasNextLine()) {
                String fileData = dataReader.nextLine();
                System.out.println(fileData);
                fwrite.write(fileData+System.lineSeparator());
            }
            dataReader.close();
            fwrite.close();
            System.out.println("output file is written");

            File fileO = new File("checkDelete.txt");
               if (fileO.delete()) {
                  System.out.println(fileO.getName()+ " file is deleted ");
                } else {
                  System.out.println("Unexpected exception");
                }

        }
        catch (FileNotFoundException exception) {
            System.out.println(" exception occurred - file is not found");
            exception.printStackTrace();
        }
        catch (IOException exception) {
                System.out.println("Unable to write to a file");
                exception.printStackTrace();
        }
    }
}
```

Comments

```
/**
 * <h2> demonstrating coments </h2>
 * This program implements shows different types of comments
 * <p>
 * <b>Note:</b> Comments help the developer to read the code
 *
 * @author Bhagvan Kommadi
 * @version 1.0
 * @since 2021-17-10
 */
public class CommentDemo {
/* static public main method */
public static void main(String[] args) {
int intValue=11; // integer with value 11
System.out.println(intValue);  //printing the integer variable
}
}
```

TypeCasting

```
public class TypeCasting
{
  public static void main(String[] args)
  {
   int x = 6;
   long y = x;
   float z = y;
   System.out.println("Before conversion, integer value "+x);
   System.out.println("After conversion, long value "+y);
   System.out.println("After conversion, float value "+z);

   double doub = 256.76;

   long lon = (long)doub;

   int intValue = (int)lon;
   System.out.println("Before conversion: "+doub);

   System.out.println("After conversion  long type: "+lon);

   System.out.println("After conversion  int type: "+intValue);
  }
}
```

Abstract Class

```
interface Mobile{
void move();
void jump();
void leap();
void eat();
}

abstract class Animal implements Mobile{
public void eat(){System.out.println("living on other animals");}
}

class Tiger extends Animal{
public void move(){System.out.println("moving");}
public void jump(){System.out.println("jumping");}
public void leap(){System.out.println("leaping");}
}

class AbstractClassExample{
public static void main(String args[]){
Mobile animal =new Tiger();
animal.move();
animal.jump();
animal.leap();
animal.eat();
}

}
```

Java Code Geeks

JAVA 2 JAVA DEVELOPERS RESOURCE CENTER