Class

//**objects** = **Instance of a class that contains methods or attributes**.
//**examples**: phone, desk, computer, coffeeCup, Charger, Fan, Tank, SwitchBoard, Plant.....

Constructor

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

        //Constructor = The method that is called when an object is instantiated (created).
        Human h = new Human();
        h.show(1);

        System.out.println();

        Human h2 = new Human("Abbas Ali",17,50);
        h2.show(2);
    }
}

public class Human{
    String name;
    int age;
    double weight;
    Human(String name, int age, double weight)
    {
        this.name=name;
        this.age=age;
        this.weight = weight;
    }
    Human()
    {
        this.name = "Hassan Raza";
        this.age=22;
        this.weight = 75;
    }
    void show(int id)
    {
        System.out.println("Human #"+id);
        System.out.println("Name is: "+this.name);
```

```
        System.out.println("Age is: "+this.age);
        System.out.println("Weight is: "+this.weight);
    }
    public void nice()
    {
        System.out.println("nice");
    }
}
```

Java variable Scope

```
public class Main{
    public static void main(String[] args) {
        //local variable = declared inside a method or inside a code block and accessible only in that method..

        //Global variable = declared outside method but within the class and accessible by all the parts(methods...) of the class

        DiceRoller dr = new DiceRoller();
        dr.endline();
    }
}

import java.util.Random;


//Global Variables
public class DiceRoller {
    Random rand;
    int num;
    DiceRoller()
    {
        rand = new Random();
        roll();
    }
    void roll()
    {
        num = rand.nextInt(6)+1;
        System.out.println("Dice Number is: "+num);
    }
    void endline()
    {
```

```
        System.out.println();
    }
}
```

//local Variables.

```
// public class DiceRoller {
//     DiceRoller()
//     {
//         Random rand = new Random();
//         int num=0;
//         roll(rand,num);
//     }
//     void roll(Random rand, int num)
//     {
//         num = rand.nextInt(6)+1;
//         System.out.println("Dice Number is: "+num);
//     }
// }
```

Constructor Overloading

```
public class Main{
    public static void main(String[] args) {
        // overloaded constructor = multiple constructors with the same name but different
set of parameters.
        // name+parameters (signature)    It should be different.

        Pizza p = new Pizza();
        p.show();
    }
}


public class Pizza{
    String bread;
    String sauce;
    String cheeze;
    String topping;

    Pizza()
    {
        this.bread = "dawn";
        this.sauce = "tomato sauce";
        this.cheeze = "Mozilla";
```

```
        this.topping = "Vegetables";
    }

    Pizza(String bread)
    {
        this.bread = bread;
        this.sauce = "tomato sauce";
        this.cheeze = "Mozilla";
        this.topping = "Vegetables";
    }

    Pizza(String bread, String sauce)
    {
        this.bread = bread;
        this.sauce = sauce;
        this.cheeze = "Mozilla";
        this.topping = "Vegetables";
    }

    Pizza(String bread, String sauce, String cheeze)
    {
        this.bread = bread;
        this.sauce = sauce;
        this.cheeze = cheeze;
        this.topping = "Vegetables";
    }

    Pizza(String bread, String sauce, String cheeze, String topping)
    {
        this.bread = bread;
        this.sauce = sauce;
        this.cheeze = cheeze;
        this.topping = topping;
    }

    void show()
    {
        System.out.println("Bread: "+this.bread);
        System.out.println("Sauce: "+this.sauce);
        System.out.println("Cheeze: "+this.cheeze);
        System.out.println("Topping: "+this.topping);
    }

}
```

toString()

```
public class Main
{
   public static void main(String[] args)
   {
      //toString = special method that all objects inherit.
      //         returns a string that 'textually represents' an object.
      //         can be used both implicitly and explicitly.

      Car c = new Car();

      System.out.println(c);                  //implicitly
      System.out.println(c.toString());       //same as above //explicitly
      //without toString it will show Classname@address of c object in memory like
(Car@15db9742).

      //now we don't need that code below.
      // System.out.println("Car make: "+c.make);
      // System.out.println("Car made: "+c.made);
      // System.out.println("Car year: "+c.year);
      // System.out.println("Car color: "+c.color);
   }
}

public class Car {
   String make = "Honda";
   String made = "Civic";
   String color = "red";
   int year = 2022;

   public String toString() {
      return "Car Make: "+this.make+"\n"+"Car made: "+this.made+"\n"+"Car color:
"+this.color+"\n"+"Car year: "+this.year+"\n";
   }
}
```

Array of Objects

```
public class Main
{
   public static void main(String[] args)
   {
      //array of objects.

      //simple array syntax
      // int []numArray = new int[3];
      // char []charArray = {'a','b','c'};
      // String []strArray = new String[3];

      // objects array syntax 1.
      Food f1 = new Food("Pizza");
      Food f2 = new Food("Burger");
      Food f3 = new Food("Shawarma");

      Food []refrigerator = new Food[3];
      refrigerator[0] = f1;
      refrigerator[1] = f2;
      refrigerator[2] = f3;

      //syntax 2
      // Food []fridge = {f1,f2,f3};   //same as above

      for (int i = 0; i < refrigerator.length; i++) {
         System.out.println(refrigerator[i]);       //toString() is used here.
      }
   }
}
```

Java Object Passing

```
public class Main
{
   public static void main(String[] args)
   {
      //Java object passing.

      Car car1 = new Car("BMW");
      Car car2 = new Car("Tesla");
```

```
        Garage garage = new Garage();
        garage.park(car1);              //object passed
        garage.park(car2);              //object passed
    }
}

public class Garage
{
    void park(Car car)
    {
        System.out.println("The "+car.name+" is parked in the garage.");
    }
}

public class Car
{
    String name;
    Car()
    {
        this.name = "";
    }
    Car(String name)
    {
        this.name = name;
    }
}
```

Static

```
public class Main
{
    public static void main(String[] args)
    {
        //static keyword = It's a modifier. A single copy of variable/method that is created and shared by all the objects of that class.
        //              static member is 'owned' by a class.
        //              static members called by classname.
        //for Example =    Math.sqrt(25);       //It's also a static method.  (see it's definition by ctrl+leftClick)
        // int x = (int)Math.sqrt(5);


        Friend friend = new Friend("Rauf");
        Friend friend2 = new Friend("Riaz");
```

```
        Friend friend3 = new Friend("Sheraz");
        Friend friend4 = new Friend("Raju");

        // System.out.println("I have "+Friend.numOfFriends+" best friends.");
        Friend.displayFriendsCount();       //same as above.
    }
}

public class Friend {
    String name;                        //owned by object
    static int numOfFriends=0;       //owned by class

    Friend()
    {
        this.name="";
        numOfFriends++;
    }
    Friend(String name)
    {
        this.name=name;
        numOfFriends++;
    }

    static void displayFriendsCount()
    {
        System.out.println("I have "+numOfFriends+" best friends.");
    }
}
```

Remaining Concepts

For further OOP concepts, check out the [OOP Concepts].