1. BufferedReader $reader = new BufferedReader(new InputStreamReader(System.in));

InputStreamReader $input = new InputStreamReader(System.in);

BufferedReader $buffer = new BufferedReader($input);

// String reading internally

String str = $buffer .readLine();

// Integer reading internally

int it = Integer.parseInt(bfn.readLine());

Size --> 8kb , scanner -->1kb

2. console input

Console c = System.console();

a. String name = c.readLine();

b. int age = Integer.parseInt(c.readLine());

3. Inheritence:- When one object acquires all the properties and behaviors of a parent object, it is known as inheritance

4. class:- A class is a group of objects which have common properties. logical entity.

5. new keyword:- The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

6. A Java constructor cannot be abstract, static, final, and synchronized but We can use access modifiers.

7. JAVA METHOD:- A method is used to expose the behavior of an object.

8. Java Copy Constructor:- There is no copy constructor in Java. However, we can copy the values from one object to another like copy constructor in C++.

9. Copy constructor

public class CopyConstructor {

int age;

String name;

//constructor

CopyConstructor(int age, String name) {

this.age = age;

this.name = name;

}

\*\* //constructor to initialize another object\*\*

CopyConstructor(CopyConstructor obj){

this.age = obj.age;

this.name = obj.name;

}

void display1() {

System.out.println(name + " " + age);

}

public static void main(String[] args) {

CopyConstructor obj = new CopyConstructor(27, "Rameshwar");

CopyConstructor obj1 = new CopyConstructor(obj);

obj.display1();

obj1.display1();

}

}

10. inheritance call class mechanism

public static void main(String args[]){

//1. parent class object

Animal parent = new Animal();

parent.eat();

// parent.run(); //error because parent class can't call child class method

//2. chil class object

Dog child = new Dog();

child.eat();

child.run();

//3 parent refence variable

Animal AnimalRefence = new Dog();

AnimalRefence.eat();

// AnimalRefence.run();

//4 child reference variable -->can't

Dog DogRefence = new Animal();//error

DogRefence.eat();

DogRefence.run();

}

1. NOTE\*\*\*-> enheritence me jitni bhi class hogi sabhi ki .class file bnegi

12. Note \*\*\*\*\*

## Why multiple inheritance is not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

1. **class** A{
2. **void** msg(){System.out.println("Hello");}
3. }
4. **class** B{
5. **void** msg(){System.out.println("Welcome");}
6. }
7. **class** C **extends** A,B{//suppose if it were
9. **public** **static** **void** main(String args[]){
10. C obj=**new** C();
11. obj.msg();//Now which msg() method would be invoked?
12. }
13. }

### 2) Method Overloading: changing data type of arguments

1. **static** **int** add(**int** a, **int** b){**return** a+b;}
2. **static** **double** add(**double** a, **double** b){**return** a+b;}

But method return type not depend

1. Final
2. Final instance variable
3. Final static variable
4. Final local variable
5. Final class
6. Final method
7. **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

* Not:- \*\*\*\*\*It can have [constructors](https://www.javatpoint.com/java-constructor) and static methods also.

1. An ****interface in Java**** is a blueprint of a class

Since Java 8, we can have ****default and static methods**** in an interface.

Since Java 9, we can have ****private methods**** in an interface.

1. This:- this keyword is a reference variable that refers to the current class instance variable and current class object.
2. Super:- super keyword is a reference variable that refers to parent class object

Use :- super and this

class A{

    int a = 10;

}

class B extends A{

    int a = 20;

    public void show(){

        int a = 30;

        System.out.println(a);//30

        System.out.println(this.a);//20

        System.out.println(super.a);//10

    }

}

public class super1 {

public static void main(String args[]){

    B b = new B();

    b.show();

}

}

.