**Date: 31-10-2021**

eternal JVM market architecture on class and object.

**Class loading section:** all code including . (dat) class file. It will execute on compile time.

**Method Section :** All method including main function

Stack Section : All Reference store hare.

**Heap Section :** All Objects represent in heep section.

Diagram

Description automatically generated

**Types Of Object in Java.**

1. **General Object :** When we need reference for use multiple time in class
2. **Anonymous Object.** When only one time use objectA picture containing graphical user interface

   Description automatically generated

**General Object:**

**Student s= new Student();**

**Anonymous Object :**

class Student{

    int sid=1;

    String sname="Ram";

    void show(){

        System.out.println(sid);

        System.out.println(sname);

    }

    public static void main(String[] args) {

        new Student().show();

    }

}

**Date: 06-11-2021**

Static Keyword: The static keyword in java is used for memory management mainly. We can apply static keyword with variables, methods, blocks.

The static keyword belongs to the class than an instance of the class.

**The Static can be:**

1. Variable (also known as a class variable)
2. Method (also known as class method)
3. Block

**Static Variable:**

If you declare any variables as static, it is known as a static variable

1. The static variable can be used to refer to the common property of all objects (Which is not unique for each object), for example-the company name of the employee class.
2. The static variable gets memory only one in the class area at the time of class loading.

Advantage of static variable

1. It makes your program memory efficient (i.e., it Save Memory).

Before:

class employee{

    int eid;

    String ename,cname;

    void input(int ei,String en, String cname){

        eid=ei;

        ename=en;

        cname=cname;

    }

    void disp(){

        System.out.println(eid);

        System.out.println(ename);

        System.out.println(cname);

            }

            public static void main(String... s){

                employee e1=new employee();

                e1.input(1,"sita","tcs");

                e1.disp();

                employee e2=new employee();

                e2.input(2,"Ram","tcs");

                e2.disp();

            }

}

Diagram

Description automatically generated

**After Use Static keywords:**

class employee{

    int eid;

    String ename;

    static String cname="TCS";

    void input(int ei,String en){

        eid=ei;

        ename=en;

    }

    void disp(){

        System.out.println(eid);

        System.out.println(ename);

        System.out.println(cname);

            }

            public static void main(String... s){

                employee e1=new employee();

                e1.input(1,"sita");

                e1.disp();

                employee e2=new employee();

                e2.input(2,"Ram");

                e2.disp();

            }

}

Diagram

Description automatically generated

**Date : 07-Nov-2021**

**Static Method:**

If you apply static keyword with any method, it is known as static method.

1. A static method belongs to class rather than the object of a class.
2. A static method can be invoked without the need for creating an instance of a class.
3. A static method can access static data member and can change the value of it.

Restriction for the static method

There are two main restrictions for the static method

1. The static method can not use non-static data member or call non-static method directly.
2. This and super can not be used in static context.

Exp:

static{

////Body of static block

}

1. class cal{
2. static int cube(int x){
3. return x\*x\*x;
4. }
5. public static void main(String... s){
6. int result=cal.cube(5);
7. System.out.println(result);
8. }
9. }

**Static Block:**

1. Is used to initialize the static data member
2. It is executed before the main method at the time of class loading

**Loading Structure**

Block>Method>Data Member

**Date: 20 Nov 2021**

**Need Of Constructor:**

* In java a constructor is a block of codes similar to the method. It is called when an instance of the class is created.
* It is a special type of method which is used to initialize the object
* Every time an object is created using the new keyword at least one constructor is called.
* It calls a default constructor if there is no constructor available into the class.in such case java compiler provides a default constructor by default.

There are two types of constructors

1. No arg constructor (Default Constructor)
2. Parameterized constructor.

Rule for creating constructors.

1. Constructor name must be the same as class name.
2. A constructor must have no explicit return type.
3. A java constructor cannot be abstract, static and final.

class studentdetails{

    int sid;

    String sname,addr;

    void input(int sid,string sname,string addr){

        this.sid=sid;

        this.sname=sname;

        this.addr=addr;

    }

    void disp(){

        System.out.println(sid);

        System.out.println(sname);

        System.out.println(addr);

    }

    public static void main(String... s) {

        studentdetails s1=new studentdetails();

        s1.disp();

        s1.input(1, "ram", "patna");

    }

}

Diagram

Description automatically generated

**Default Constructor (No arg Constructor):**

**A constructor is called default constructor when it doesn’t have any parameters.**

**Date : 21-Nov-2021**

**Constructor is a part of class .**

**Parameterized Constructor:**

**A constructor who has a specific number of parameterize is called a parameterized constructor.**

**Q. Why use the parameterize constructor.**

The parameterized constructor is used to provide different value to distinct object. However you can provide the same value also.

Syntax of Parameterized constructor.

ClassName(datatype para1, datatype para2, datatype para3….){

///Body of constructor.

}

class emp{

    int eid;

    String ename,post;

//////Parameterized constructor

    emp(int eid,String ename,String post){

        this.eid=eid;

        this.ename=ename;

        this.post=post;

    }

    void disp(){

        System.out.println(eid);

        System.out.println(ename);

        System.out.println(post);

}

public static void main(String ...s) {

    emp e1=new emp(1,"ram","prog");

    e1.disp();

    emp e2=new emp(2,"Mohan","prog");

    e2.disp();

}

}

**Signature:** Data type , parameters.

In a class should be different parameter or data type.

Constructor Overloading:

class student{

    int sid;

    String sname,stype;

    int age;

    student(int sid,String sname){

        this.sid=sid;

        this.sname=sname;

    }

    student(int sid,String sname,String stype){

        this.sid=sid;

        this.sname=sname;

        this.stype=stype;

    }

    student(int sid,String sname,String stype,int age){

        this.sid=sid;

        this.sname=sname;

        this.stype=stype;

        this.age=age;

    }

    void disp(){

        System.out.println(sid);

        System.out.println(sname);

        System.out.println(stype);

        System.out.println(age);

    }

    public static void main(String...s) {

        student s1=new student(1,"ram");

        s1.disp();

        student s2=new student(1,"sita","two");

        s2.disp();

        student s3=new student(1,"ram","three",34);

        s3.disp();

    }

}

Date: 27-Nov-2021

**Copy Constructor:**

Java does not support copy construction. But we can copy behaves of the construction.

class copycon{

    int x,y;

    copycon(int x,int y){

        this.x=x;

        this.y=y;

    }

    copycon(copycon z){

        this.x=z.x;

        this.y=z.y;

    }

    void show(){

        System.out.println(x);

        System.out.println(y);

    }

    public static void main(String...s) {

        copycon c1=new copycon(10,20);

        c1.show();

        copycon c2=new copycon(c1);

        c2.show();

    }

}

Diagram

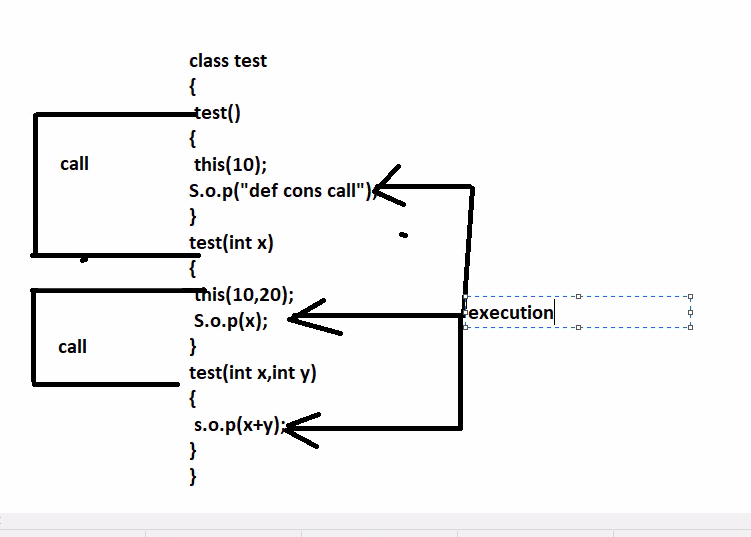
Description automatically generated

**Chaining Constructor:**

* In java constructor chaining is a sequence of invoking constructor upon initialization an object.
* It is used when we want to invoke a number of constructors one after another by using only an instance.
* In constructor chain a constructor is called from another constructor in the same class this process is known as constructor chaining.

**Rule of constructor chaining**.

* An expression that uses **this** keyword must be the first line of the constructor.
* Order does not matter in constructor chaining.
* There must exist at least one constructor that does not use **this**.



\*Constructor chaining it will find the this keyword in the constructor then it will continue, If this keyword not found in the first line of the constructor then break chaining and start executing body area

**Date: 28-Nov-2021**

**Method Overloading**

If a class has multiple method having same name but different in parameter, it is known as **method overloading**.

**Parameter Differ:**

<returntype> functionname(datatype para1, datatype para2){

// Body of method

}

<returntype> functionname(datatype para1, datatype para2, datatype para3){

// Body of method

}

* If parameters are different then we can define same name method multiple times.

class poly{

    void sum(int x,int y){

        System.out.println(x+y);

    }

    void sum(int x,int y,int z){

        System.out.println(x+y+z);

    }

    public static void main(String...s){

        poly p1=new poly();

        p1.sum(10,20);

        p1.sum(10,20,30);

    }

}

Method overloading check on compile time, if we define same name method and signature multiple time the it will throw error on

**Alternative way of method overloading**

We can use static keyword to load method overloading on compiletime.

class poly{

  /\*  void sum(int x,int y){

        System.out.println(x+y);

    }

    void sum(int x,int y,int z){

        System.out.println(x+y+z);

        \*/

    }

  static  void sum(int x,int y){

        System.out.println(x+y);

    }

  static  void sum(int x,int y,int z){

        System.out.println(x+y+z);

    }

    public static void main(String...s){

       /// poly p1=new poly();

       // p1.sum(10,20);

       // p1.sum(10,20);

       sum(10,20); /// use static keyword no need to create object

       sum(10,20,30);

    }

}

**Data Type Differ:**

<returntype> functionname(datatype1 para1, datatype1 para2){

// Body of method

}

<returntype> functionname(datatype2 para1, datatype2 para2){

// Body of method

}

class poly{

  /\*  void sum(int x,int y){

        System.out.println(x+y);

    }

    void sum(int x,int y,int z){

        System.out.println(x+y+z);

        \*/

  //  }

 /\* static  void sum(int x,int y){

        System.out.println(x+y);

    }

  static  void sum(int x,int y,int z){

        System.out.println(x+y+z);

    }

    public static void main(String...s){

       /// poly p1=new poly();

       // p1.sum(10,20);

       // p1.sum(10,20);

       sum(10,20); /// use static keyword no need to create object

       sum(10,20,30);

    } \*/

///// Call by datatype differ

static  void sum(int x,int y){

    System.out.println(x+y);

}

static  void sum(Float x,Float y){

    System.out.println(x+y+z);

}

public static void main(String...s){

   /// poly p1=new poly();

   // p1.sum(10,20);

   // p1.sum(10,20);

   sum(10,20); /// use static keyword no need to create object

   sum(10.4f,20.5f);

}

}

**Why method overloading is possible by changing return type.**

Yes, But if we change parameter then it’s show error.

**Errorr:** poly.java:55: error: no suitable method found for sum(float,float)

System.out.println(p1.sum(10.5f, 20.4f));

^

method poly.sum(int,char) is not applicable

(argument mismatch; possible lossy conversion from float to int)

method poly.sum(int,int) is not applicable

(argument mismatch; possible lossy conversion from float to int)

class poly{

  /\*  void sum(int x,int y){

        System.out.println(x+y);

    }

    void sum(int x,int y,int z){

        System.out.println(x+y+z);

        \*/

  //  }

 /\* static  void sum(int x,int y){

        System.out.println(x+y);

    }

  static  void sum(int x,int y,int z){

        System.out.println(x+y+z);

    }

    public static void main(String...s){

       /// poly p1=new poly();

       // p1.sum(10,20);

       // p1.sum(10,20);

       sum(10,20); /// use static keyword no need to create object

       sum(10,20,30);

    } \*/

///// Call by datatype differ

/\*

static  void sum(int x,int y){

    System.out.println(x+y);

}

static  void sum(Float x,Float y){

    System.out.println(x+y+z);

}

public static void main(String...s){

   /// poly p1=new poly();

   // p1.sum(10,20);

   // p1.sum(10,20);

   sum(10,20); /// use static keyword no need to create object

   sum(10.4f,20.5f);

}

\*/

  int sum(int x,char a){

    return(x+a);

}

  double sum(int x,int y){

    returnrintln(x+y);

}

public static void main(String...s){

    poly p1=new poly();

   System.out.println(p1.sum(10, 'a'));

}

}

**Date: 04 Dec 2021**

**Inheritance:**

**There is two types of inheritance**

* IS A (association)
* Has A (Aggregation)

**Inheritance (IS A):-**

* Inheritance in java a mechanism in which one object acquires all the property and behavior of a parent object.
* Create a new class that are build upon existing classes. When you inherit from existing class can reuse method and fields of parent class.
* Inheritance represents the IS-A relationship.

**Q. Why use inheritance in java**

* For code reusability
* Run time polymorphism (Method Overriding)

**Types of inheritance in java.**

1. Single
2. Multilevel

\*Java does not support multiple inheritance because if we found same method in another class then it will through error (Java does not support pointer).

Diagram

Description automatically generated

**Syntax of Single inheritance:**

Class subclass\_name extends superclass{

////Fields and methods

}

While creating child class object **implicit** parent class object has been created by **JVM**

Private method or data member can be inherit but we can’t access any data member or methods

**Multilevel inheritance:**

When there is a chain of inheritance, it is known as **multilevel inheritance**.

Date: 05-Dec 2021

Super Keyword

When we try to call parent construction except default, we need to use **super keyword** in child class in the first line of the child class.

**Super** is a keyword immediate parent class.

\*Super keywords hold parent class reference.

**Data Overriding:** Whenever a parent class and child class both are having same data member then this concept is known as data overriding

**Method Overriding:** Whenever a parent class and child class both are having same method then this concept is known as **Method Overriding**.

Overriding always achieve on runtime.

Overriding always achieve in IS-A relationship.

Method overloading always achieve on compile time

Method overriding achieve on run time.

**This and super Keyword**

This keyword store current class reference

Super keyword store parent class reference.

Diagram

Description automatically generated with medium confidence

**Relationship always works stronger to weaker.**

Public ------Stronger

default

Protected

Private---------Weaker

|  |  |
| --- | --- |
| Parent | child |

Stronger weaker

Weaker stronger

Stronger Stronger

Static Static Not overridden.

This situation is known as method hiding (when both have static method).

1.Static method can’t be overridden.

2.Super and this always works with none-static

Date: 11-Dec-2021

**Dynamic Binding or Runtime Binding:** When we store child class object in base class reference it is known as **Dynamic Binding**

class base {

    void show(){

        System.out.println("Base show call");

    }

}

class child extends base {

    void show(){

        System.out.println("Child Class show");

        super.show(); //// access parent class show method

    }

    void disp(){

        System.out.println("Child disp show");

        super.show(); //// access parent class show method

    }

    public static void main(String...s){

        base b= new child(); //////upcasting

        b.show();

        ////We can't access parent class refferance to child class .

        child c=(child)b; //// Downcasting

        c.disp();

    }

}

**Abstraction:**

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

There are two ways to achieve abstraction in java

* Abstract Class
* Interface

**Abstract Class:**

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract method. It needs to be extended and its method implemented. It can’t be instantiated.

* An abstract class must be declared with an abstract keyword.

abstract class Animal{

}

* It can have abstract and non-abstract method

abstract class animal{

Void nonfly(){

//////////Body of the methods

}

Abstract void fly();

}

* It can’t be instantiated.

abstract class animal{

    abstract void fly();

    void nonfly(){

        System.out.println("non fly method");

    }

}

class bird extends animal{

    void fly(){

        System.out.println("Bird fly call");

    }

    public static void main(String...s){

        bird b= new bird();

        b.fly();

        b.nonfly();

    }

}

* It can have final method which will force the sub class not to change the body of the method.

abstract class animal{

    abstract void fly();

    ///////

   final  void nonfly(){

        System.out.println("non fly method");

    }

}

class bird extends animal{

    void fly(){

        System.out.println("Bird fly call");

    }

      void nonfly(){

        System.out.println("non bird class method");

    }

    public static void main(String...s){

        bird b= new bird();

        b.fly();

        b.nonfly();

    }

}

Error:

animal.java:13: error: nonfly() in bird cannot override nonfly() in animal

      void nonfly(){

           ^

  overridden method is final

1 error

Date: 12 Dec 2021

Q. Can we create constructor in abstract class.

Yes

abstract class base{

    base(){

        System.out.println("Base class constroctor");

    }

    abstract void display();

}

class child extends base{

    void display(){

        System.out.println("Child class method ");

    }

    public static void main(String[] args) {

        base b=new child();

        b.display();

    }

}

**Interface:**

* Interface are blueprint of a class
* Interface are contract between java programmer and programming language

Diagram

Description automatically generated

Interface Rule:

* Interface cannot be instantiated.
* By default, all the method of interface are public or abstract.
* By default, interface are abstract.

interface My{

    void show();

}

class child implements My{

    public void show(){

        System.out.println("Child class override interface");

    }

    public static void main(String[] args) {

        My m=new child();

        m.show();

    }

}

If we try to override interface method, then we must be declared as public because stronger to weaker not working.

**Keyword use to inherit class or interface**

Diagram

Description automatically generated

Date: 18-Dec-2021

Interface:

By default, all data member of interface is public, static or final.

In java 8 and above version we can create in interface body method. But make it default.

Final:

In java final is a reserved keyword.

3 ways to use final keyword

* Variable Final--- Variable unchangeable
* Method final---Doesn’t override
* Class final---Doesn’t Inherit

We can’t use non static variable inside static variable

Instance data member memory allocate on creating on class object

**Final Methods**

We can’t override final methods..

class Base{

    final void show(){

        System.out.println("Hello Show");

    }

}

class Child extends Base{

    void show(){

        System.out.println(" Show method override");

    }

    public static void main(String[] args) {

        Child c1=new Child();

        c1.show();

    }

}

//It will throw error final method can’t be overridden.

But we can inherit final methods.

class Base{

    final void show(){

        System.out.println("Hello Show");

    }

}

class Child extends Base{

    public static void main(String[] args) {

        Child c1=new Child();

        c1.show();

    }

}

**Final class**

We can’t extend final class.

Q. What is **Blank final variable** or **uninitialized Variable**?

If you make any final variable is uninitialized variable is known as Blank Final variable.

Class Temp{

Final int x; Blank final variable.

}

Without constructor we can’t overwrite final variable

class Temp{

    final x;

    void show(){

        x=100;

        System.out.println(x);

    }

    public static void main(String[] args) {

        Temp t=new Temp();

        t.show();

    }

}

Using constructor, we can overwrite final variable.

class Temp{

    final x;

    Temp(){

        x=100;

        System.out.println(x);

    }

    public static void main(String[] args) {

        Temp t=new Temp();

        t.show();

    }

}

Date: 19-Dec-2021

* Blank Final variable can’t be initialized in default constructor.
* Blank Final variable only initialized in custom constructor.

**Package**:

* + Collection of similar classes or interfaces.
  + Cauterization easy to use and search.
  + In C header is a collection of predefine function.
  + No class exist without any package in java.
  + Always write import statement on the top.
  + Package name should be in lowercase;
  + Class path we use for runtime set path of any . class file

Set Classpath= D:\package;

* + Check class path set or not

Set classpath;

* + If we want to use class or method, then we must be making class and method public

* There are three ways to use package;

1. Import all classes.

Import java.util.\*;

Class temp{

Public static void main(Sting[] args){

Scanner s=new scanner()

}

}

1. Import all one by one classes.

Import java.util.Scanner;

Import java.swing.Jbutton;

Class temp{

Public static void main(Sting[] args){

Scanner s=new scanner();

}

}

1. Without using import statement

Class temp{

Public static void main(String[] args){

Java.util.scanner s1=new java.util.Scanner(System.in);

Java.awt.Farme f=new java.awt.Farme();

}

}

How to create Package:

Package packagename;

Class className{

}

**For creating package command:**

**Javac -d . Temp.java**

**-for switch**

**d for designation**

**. for path where package will be created.**