Q1. Explain super keyword in java programming?
Ans:
Using super we can call parent class constructor, we can access parent class member data, we can call parent class member function inside child class

1. Call parent class constructor
Syntax:
super(parameter1,parrameter2);

2. Access Parent class member data
Syntax:
super.memberDataName;
3. Call parent class method
Syntax:
super.methodName();
Note: super must be first line of the the block

```
class Point{
public int x;
public int y;//instance variable
public int z=101;
public Point(){
System.out.println("Point class Default
Constructor is called");
```

```
}

public Point(int x,int y){
this.x=x;
this.y=y;
System.out.println("Point class Parameterized
Constructor is called");
}

void showData(){
System.out.println("X_CO : "+x);
System.out.println("Y_CO : "+y);

}
void hi(){
System.out.println("Hi... Method is Called");

int x=111;
int y=222;
System.out.printf("\nx=%d  Y=%d ",this.x,this.y);


}
```

```
}
class Circle extends Point{
float r;
int z=102;
public Circle(){
System.out.println("Circle class Default
Constructor");
}
public Circle(int x,int y,float r){
super(x,y);//call  parent class constructor
this.r=r;
System.out.println("Circle class Parameterized
Constructor");
}
void showData(){
super.hi();
System.out.println("X_CO : "+x);
System.out.println("Y_CO : "+y);
System.out.println("Radius is : "+r);
System.out.println("circle z= : "+z);
System.out.println("Point class z= : "+super.z);
}
```

```
public static void main(String args[]){
//Circle c1=new Circle();
//Point class default constructor (1)and Circle
class default constructor(2)
Circle c2=new Circle(11,22,5.6f);
c2.showData();
}



}
```

Q2. Explain Polymorphism in java programming?
Ans:
One Name Multiple form is known as Polymorphism
There are two types of Polymorphism in java
1. Compile Time Polymorphism:(Method Overloading): Decide at compile time which method is called(Static Binding/Early Binding)

2. Run time Polymorphsm(Dynamic Binding/Late Binding) : (Method Overriding):
Decide at run time which method is called

Q3. Explain Method Overloading in java Programming?

Ans: Method overloading means defined multiple methods with same name but different signature.

Points Regarding Method Overloading

```
public return type methodName(Data Type v1,Data Type v2 ){

}

public void add(int a,int b ){
//definition of the method
}
```

1. Same Method Name: All overloaded methods must have same name
2. Different Parameters: Overloaded methods differ in

A. Number of Parameter

B. Data types of Parameters

C. Order of Parameters

3. Return type: Return type can be different or same but it does not affect the concept of method overloading

```
class Test{

public void add(){
int a,b,c;
a=1;
b=2;
c=a+b;
System.out.println("Addition without argument : "+c);
}
public void add(int a,int b){
int c;
c=a+b;
System.out.println("Addition with two int argument : "+c);

}
public void add(int a,float b){
float c;
c=a+b;
System.out.println("Addition with two int,float argument :
"+c);

}
public void add(float a,int b){
float c;
c=a+b;
System.out.println("Addition with two float,int argument :
"+c);
```

```
}
public void add(String a,String b){
int c;
c=Integer.parseInt(a)+Integer.parseInt(b);
System.out.println("Addition with two String argument :
"+c);

}

public static void main(String args[]){
Test t=new Test();
t.add();
t.add(10,20);
t.add(5,2.5f);
t.add(15.5f,10);
t.add("1","1");

}


}
```

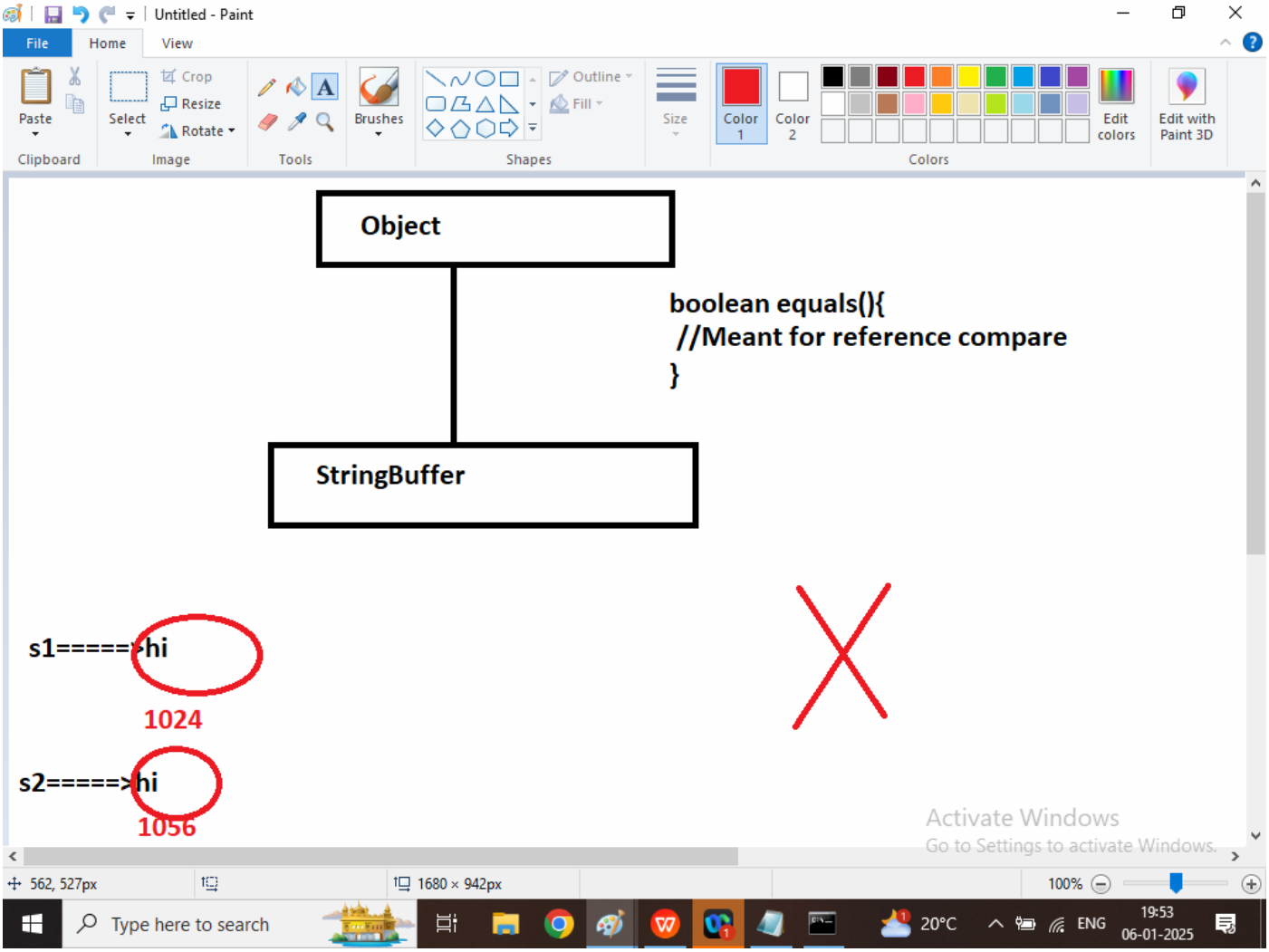
*S2 - Notepad

File Edit Format View Help

```
class S2{
        public static void main(String args[]){

        StringBuffer sb1=new StringBuffer("hi");
        StringBuffer sb2=new StringBuffer("hi");
        System.out.println(sb1.equals(sb2));//false

        }

 }
```


Ln 6, Col 45 | 90% | Windows (CRLF) | UTF-8

Type here to search | 20°C | ENG | 19:52 06-01-2025


Untitled - Paint
File
Home
View
Paste
Select
Crop
Resize
Rotate
Brushes
Outline
Fill
Size
Color 1
Color 2
Edit colors
Edit with Paint 3D
Clipboard
Image
Tools
Shapes
Colors
Object
boolean equals(){
//Meant for reference compare
}
StringBuffer
s1=====>hi
1024
s2=====>hi
1056
Activate Windows
Go to Settings to activate Windows.
562, 527px
1680 × 942px
100%
Type here to search
20°C
ENG
19:53
06-01-2025

S2 - Notepad

File Edit Format View Help

```
class S2{
        public static void main(String args[]){

        String sb1=new String("hi");
        String sb2=new String("hi");
        System.out.println(sb1.equals(sb2));//true

        }

}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>javac S2.java

C:\Users\Admin\Desktop\Java Task>java S2
true

C:\Users\Admin\Desktop\Java Task>
```

S2 - Notepad

File Edit Format View Help

```
class S2{
        public static void main(String args[]){

        String sb1=new String("hi");
        String sb2=new String("hi");
        System.out.println(sb1.equals(sb2));//true

        }

 }
```


Ln 6, Col 44 | 90% | Windows (CRLF) | UTF-8

Type here to search 20°C ENG 19:54 06-01-2025

S2 - Notepad

File Edit Format View Help

```
class S2{
        public static void main(String args[]){

        Object sb1=new String("hi");
        Object sb2=new String("hi");
        System.out.println(sb1.equals(sb2));//true

        }

}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>java S2
true

C:\Users\Admin\Desktop\Java Task>
```

Untitled - Paint

Object

```
boolean equals(){
//Meant for reference
}
```

String

```
override
boolean equals(){
//Meant for content compare
}
```


439, 513px 1680 × 942px 100%

*S2 - Notepad

File Edit Format View Help

```
class S2{
        public static void main(String args[]){

        Object sb1=new String("hi");
        Object sb2=new String("hi");
        System.out.println(sb1.equals(sb2));//true

        }

 }
```

Activate Windows
Go to Settings to activate Windows.

Ln 4, Col 23 | 90% | Windows (CRLF) | UTF-8

Type here to search | ENG | 19:56 06-01-2025

Q2.Explain Method Overriding in java Programming?

Ans: Method overriding in java is a features that allows a sub class to provide a specific implementation of a method already defined in parent class. It is used to achieve run time Polymorphism in java and customize or enhance behaviour of an inherited method

Some points about method overriding

1. A class must have a “IS-A” Relationship
2. Method name and Signature must be same as its parent class method
3. Inheritance: The sub class must inherit from the super class where method is defined

Child - Notepad

File Edit Format View Help

```
class Parent{
        public void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method....");
        }


        public static void main(String args[]){
        Parent c=new Parent();
        c.show();
        }

}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>java Child
This is Parent class show method....

C:\Users\Admin\Desktop\Java Task>
```


Type here to search

Earn... ENG 20:07 06-01-2025

Child - Notepad

File Edit Format View Help

```
class Parent{
        public void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method....");
        }


        public static void main(String args[]){
        Parent c=new Child();
        c.show();
        }

}
```

Ln 13, Col 20

Command Prompt

```
C:\Users\Admin\Deskto
This is Parent class

C:\Users\Admin\Deskto

C:\Users\Admin\Deskto
This is Child class s

C:\Users\Admin\Deskto
```


Type here to search

Earn... ENG 20:08 06-01-2025

Child - Notepad

File Edit Format View Help

```
class Parent{
        public void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method....");
        }


        public static void main(String args[]){
        Parent c=new Child();
        c.show();
        }

}
```


Ln 18, Col 1 | 70% | Windows (CRLF) | UTF-8

Type here to search | ENG | 20:18 06-01-2025

Q1. Explain final keyword in java?

Ans: It is used to perform some restrictions

1. Final variable: cannot be changes
2. Final method: cannot be override

Child - Notepad

File Edit Format View Help

```
class Parent{
        public final void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method....");
        }


        public static void main(String args[]){
        Parent c=new Child();
        c.show();
        }

}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>javac Child.java
Child.java:7: error: show() in Child cannot overri
) in Parent
        public void show(){
                         ^
  overridden method is final
```

Activate Windows
Go to Settings to activate Windows.

Type here to search 20°C ENG 20:20 06-01-2025

3. Final class: final class cannot be inherit

Child - Notepad

File Edit Format View Help

```
class Parent{
        final int MAX=10;
        public  void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method...."+MAX);
        }


        public static void main(String args[]){
        Parent c=new Child();
        c.show();
        }

}
```


Ln 4, Col 2 | 70% | Windows (CRLF) | UTF-8

Type here to search 20°C ENG 20:23 06-01-2025

Child - Notepad

File Edit Format View Help

```
final class Parent{
        public  void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method....");
        }


        public static void main(String args[]){
        Parent c=new Child();
        c.show();
        }

}
```

Select Command Prompt

```
Child.java:6: error: cannot inherit from final Par
class Child extends Parent{
                    ^
1 error

C:\Users\Admin\Desktop\Java Task>
```

Child - Notepad

File Edit Format View Help

```
class Parent{
        final int MAX=10;
        public  void show(){
        System.out.println("This is Parent class show method....");
        }
}
class Child extends Parent{
        public void show(){
        System.out.println("This is Child class show method...."+super.MAX);
        final int MAX=55;
        System.out.println("This is Child class show method...."+MAX);
        }


        public static void main(String args[]){

        Parent c=new Child();
        c.show();
        }

}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>javac

C:\Users\Admin\Desktop\Java Task>java C
This is Child class show method....10
This is Child class show method....55

C:\Users\Admin\Desktop\Java Task>
```


Type here to search

ENG 20:27 06-01-2025

Q1. Explain static block, static variables and static methods?

Ans:

The static keyword in java is used to indicate that a particular member(filed,block or methods) belongs to the class rather than object / instance of a class.

Note: static member recommended to access via name of the class

Syntax:

Static Variable:

ClassName.memberdataName;

Static Methods:

ClassName.methodName();

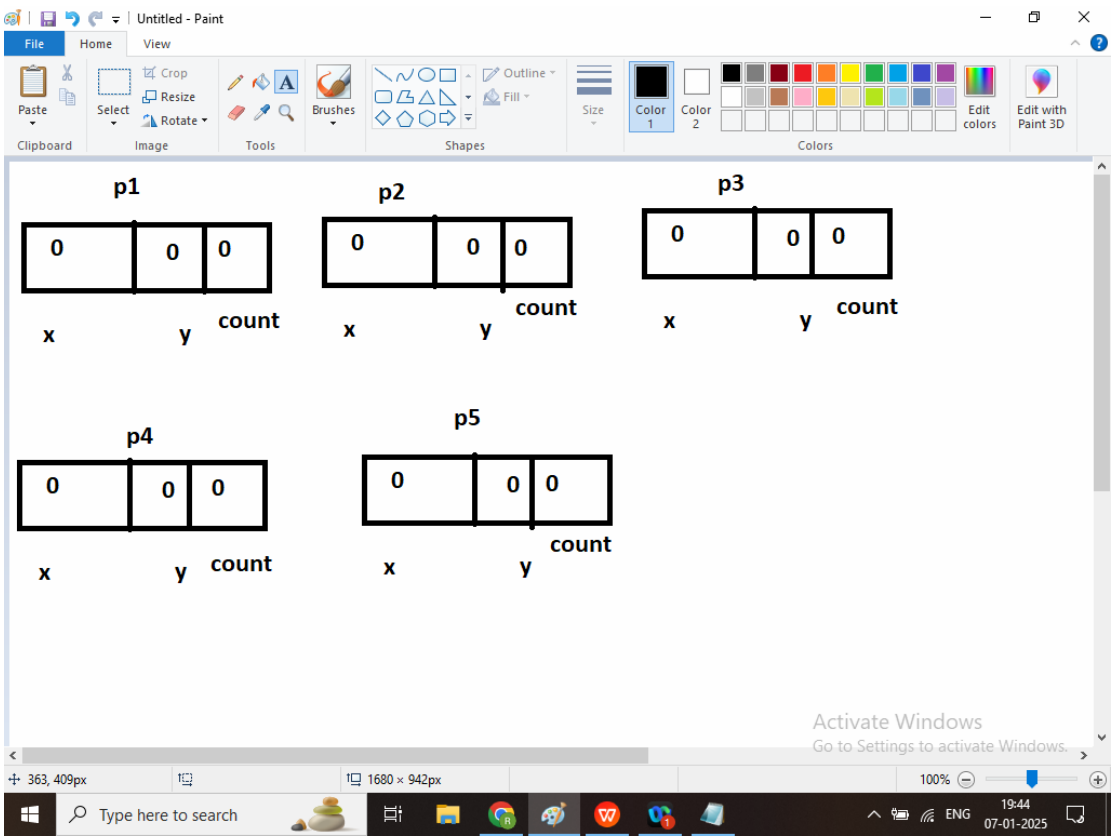
Static block is automatically called when object class is loaded.

Static block always executed before main method

If a class contain multiple static block then it will be executed according to the order in which they are defined top to bottom

Static Variables: A static variable is shared among all instance of a class

Memory is allocated once for static variables at the time of class loading



*Point - Notepad

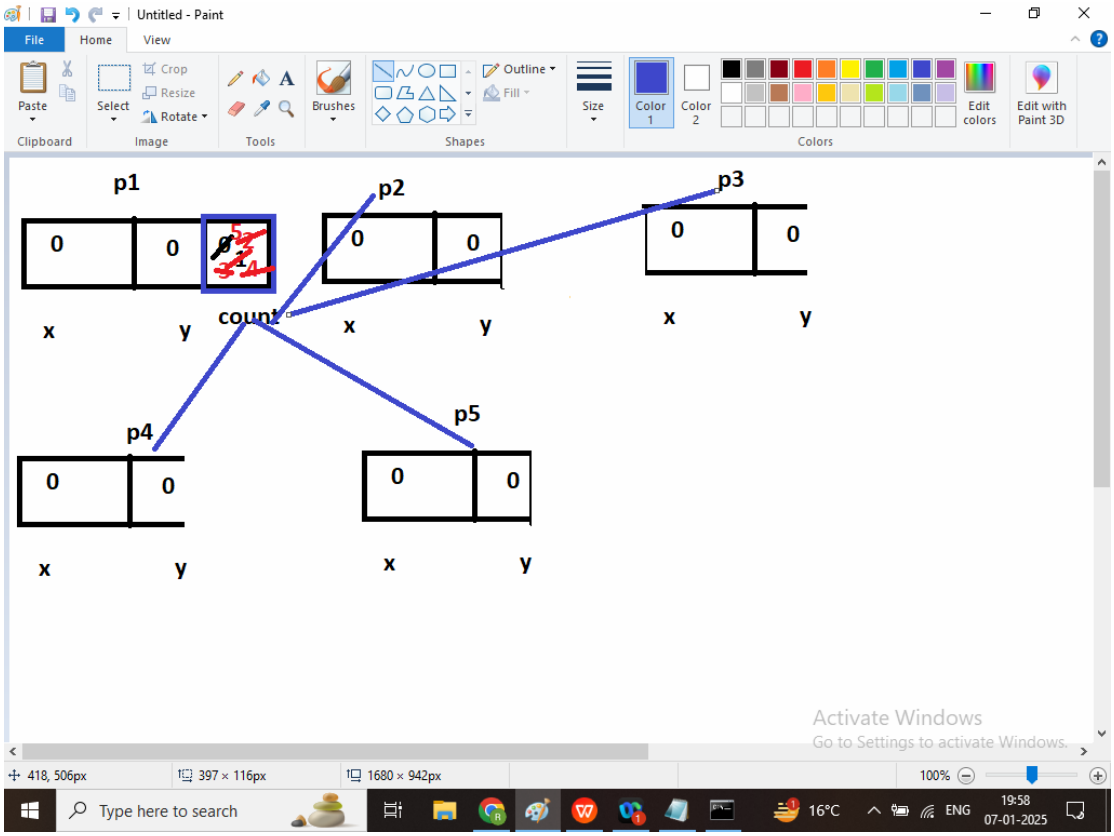
File Edit Format View Help

```
class Point{
int x;
int y;//instance variable
int count;

public Point(){
count++;
}
public static void main(String args[]){
Point p1=new Point();
Point p2=new Point();
Point p3=new Point();
Point p4=new Point();
Point p5=new Point();
System.out.println(p1.count);
System.out.println(p2.count);
System.out.println(p3.count);
System.out.println(p4.count);
System.out.println(p5.count);

}
```


Ln 4, Col 1 | 70% | Windows (CRLF) | UTF-8

Command Prompt

```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd Desktop

C:\Users\Admin\Desktop>cd "Java Task"

C:\Users\Admin\Desktop\Java Task>javac Point.java

C:\Users\Admin\Desktop\Java Task>java Point
1
1
1
1
1

C:\Users\Admin\Desktop\Java Task>
```



*Point - Notepad

```
//static variable example
class Point{
int x;
int y;//instance variable
static int count;

public Point(){
count++;
}
public static void main(String args[]){
Point p1=new Point();
Point p2=new Point();
Point p3=new Point();
Point p4=new Point();
Point p5=new Point();
System.out.println(p1.count);
}
}
```

Static method: static methods can be called without creating an object of the class

These methods can only access static variables and other static methods directly (without using this and super)

Static method is recommended to call using class Name
ClassName.methodName();

```
//static methods example
class Point{
int x;
int y;//instance variable
static int count;

public Point(){
count++;
}
public static int getCtr(){
```

```
return count;
}
public static void main(String args[]){
Point p1=new Point();
Point p2=new Point();
Point p3=new Point();
Point p4=new Point();
Point p5=new Point();
System.out.println(p1.count);
System.out.println("Number of Object
is created : "+count);
System.out.println("Number of Object
is created : "+Point.count);
System.out.println("=======static
methods call=======>");
System.out.println("No. of Object
Created : "+p1.getCtr());
System.out.println("No. of Object
Created : "+Point.getCtr());
System.out.println("No. of Object
Created : "+getCtr());
```

```
}
}
```

Static block: static block is used to initialize static variables

Executed only once when the class is loaded in the memory

```
//static methods example
class Point{
int x;
int y;//instance variable
static int count;
static{
count=100;
System.out.println("This is Static Block 1 here");
}
public Point(){
```

```
count++;
}
public static int getCtr(){
return count;
}
static{
count=300;
System.out.println("This is Static Block
3 here");
}
public static void main(String args[]){
System.out.println("This is Main
Method Here");
Point p1=new Point();
Point p2=new Point();
Point p3=new Point();
Point p4=new Point();
Point p5=new Point();
System.out.println(p1.count);
System.out.println("Number of Object
is created : "+count);
```

```
System.out.println("Number of Object
is created : "+Point.count);
System.out.println("=======static
methods call=======>");
System.out.println("No. of Object
Created : "+p1.getCtr());
System.out.println("No. of Object
Created : "+Point.getCtr());
System.out.println("No. of Object
Created : "+getCtr());


}
static{
count=200;
System.out.println("This is Static Block
2 here");
}

}
```

```
class Test{

    public static void add(){
    int a,b,c;
    a=10;
    b=20;
    c=a+b;
    System.out.println("Addition : "+c);

    }

public static void add(int a){
    int c;
    c=a+a;
    System.out.println("Addition : "+c);

    }

public static void add(int a,int b){
    int c;
    c=a+b;
    System.out.println("Addition : "+c);
```

```
    }

public static void add(int a,int b,int d){
    int c;
    c=a+b+d;
    System.out.println("Addition : "+c);

    }
public static void add(int a,int b,int d,int
e){
    int c;
    c=a+b+d+e;
    System.out.println("Addition : "+c);

    }
    public static void add(int f,int a,int
b,int d,int e){
    int c;
    c=a+b+d+e+f;
    System.out.println("Addition : "+c);

    }
```

```
    public static void add(int g,int f,int
a,int b,int d,int e){
    int c;
    c=a+b+d+e+f+g;
    System.out.println("Addition : "+c);

    }

    public static void main(String…args){

    add();
    add(10);
    add(1,2);
    add(10,20,30);
    add(10,20,30,40);
    add(1,2,3,4,5);
    add(1,2,3,4,5,6);
    }


}
```

Q1. Explain Variable Argument(…) in Java Programming?

Ans : In Java Variable arguments(varargs) allow a method to accept zero or more arguments of the same type. This is useful when exact number of arguments is unknown or varies.

Here datatype… indicates the method can accept a variable number of arguments of the specified data types

Example:

```
void add(int…a){
}
void add(float…a){
}

void add(double…a){
}
void add(String…a){
}
```

Example : Variable Argument

```
class Test{

    public static void add(int...x){
    int sum=0;
    for(int a:x){
    sum=sum+a;
    }
    System.out.println("Sum is : "+sum);
    }

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

    add();
    add(10);
    add(1,2);
    add(10,20,30);
    add(10,20,30,40);
    add(1,2,3,4,5);
    add(1,2,3,4,5,6);
    }
```

}

Note:

1. The varargs parameter is treated as an array within the method.
2. A Method can have exactly one varargs parameter
3. The varargs parameter must be the last parameter in the method

```
class Test{

    public static void add(int y,int...x){
    int sum=0;
    for(int a:x){
    sum=sum+a;
    }
    System.out.println("Sum is : "+sum);
    }

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

    //add();
    add(10);
```

```
    add(1,2);
    add(10,20,30);
    add(10,20,30,40);
    add(1,2,3,4,5);
    add(1,2,3,4,5,6);
    }


}
```

Q2. Explain Abstraction in java Programming?

Ans: Abstraction in java is a process to hiding the implementation details of a class and showing only the essential features. It is achieved by abstract class and interface. Abstraction helps to reduce complexity and increase maintainability

An abstract class is a class that is declared with the abstract keyword. Abstract class can contain abstract method or non

abstract method. An Abstract class can have constructor

We cannot create an instance of abstract class But it can store reference of its child class

In abstract class we can provide only declaration the method we cannot implement it inside abstract class.

If we extends any abstract class we must override its abstract method or we declare child class also abstract class.

College - Notepad

File Edit Format View Help

```
abstract class Rgpv{
        public abstract void lab();
        public abstract void cctv();
        public void copyCheck(){
        System.out.println("This is Copy Check Method of RGPV class and this is non
abstract method");
        }

}

abstract class College extends Rgpv{

}
```

Activate Windows
Go to Settings to activate Windows.

Ln 12, Col 2 | 60% | Windows (CRLF) | UTF-8

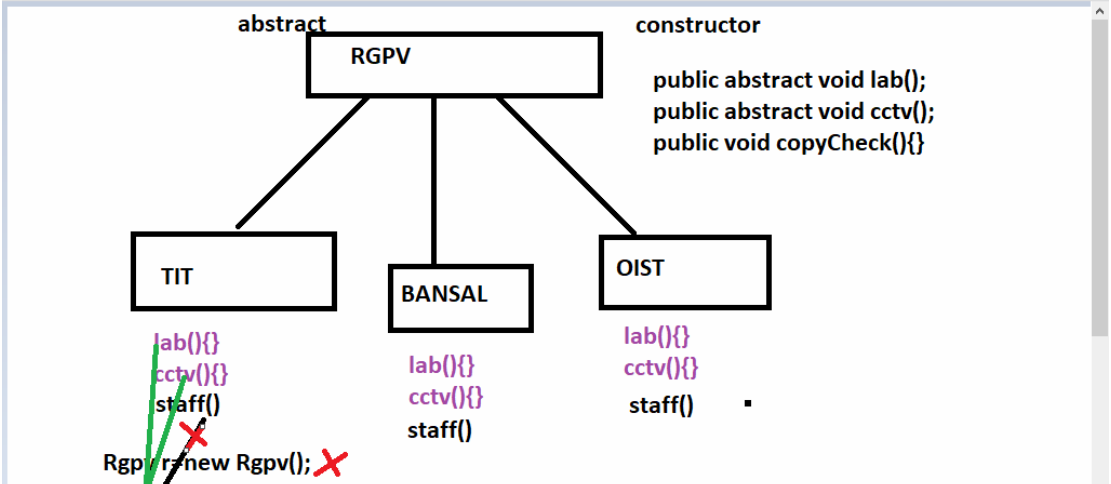
Type here to search | 16°C | ENG | 20:18 08-01-2025

Untitled - Paint

File Home View

Paste | Select | Crop | Resize | Rotate | Brushes | Outline | Fill | Size | Color 1 | Color 2 | Edit colors | Edit with Paint 3D

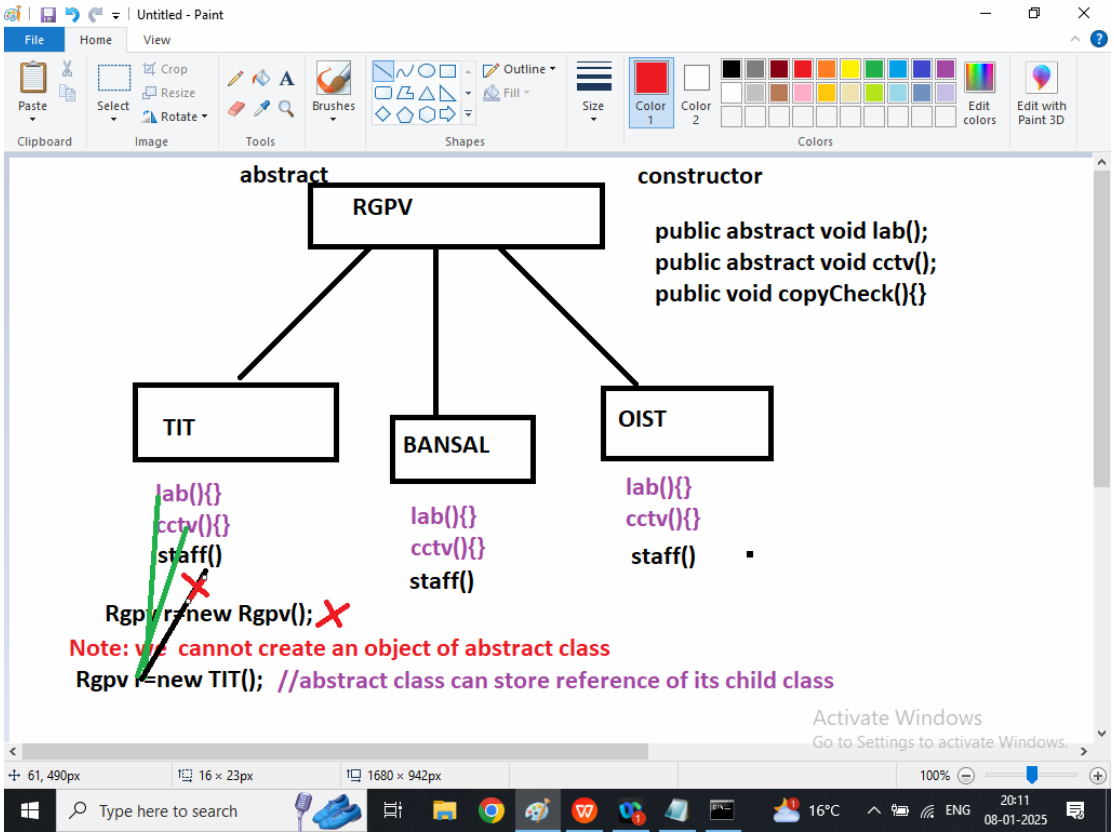
Clipboard | Image | Tools | Shapes | Colors



Activate Windows
Go to Settings to activate Windows.

16 × 23px | 1680 × 942px | 100%

Type here to search | 16°C | ENG | 20:14 08-01-2025



```
abstract class Rgpv{
    public Rgpv(){
    System.out.println("This is RGPV Class Constructor");
    }
    public abstract void lab();
    public abstract void cctv();
    public void copyCheck(){
    System.out.println("This is Copy Check Method of RGPV class and this is non abstract method");
    }
```

```
}

class College extends Rgpv{
public College(){
      System.out.println("This is College
Class Constructor");
      }
public void cctv(){
System.out.println("This is Rgpv CCTV
method");
}
public void lab(){
System.out.println("This is Rgpv lab
method");
}
public void staff(){
System.out.println("This is College staff
method");
}

public static void main(String args[]){
Rgpv r=new College();
```

```
r.cctv();
r.lab();
r.copyCheck();
//r.staff();
College t1=new College();
t1.cctv();
t1.lab();
t1.copyCheck();
t1.staff();
}
}
```

Q1. Explain interface in java programming?

Ans: if we want to achieve 100% abstraction then we should go for interface

Interface is the collection of method declaration and prototype

By default methods of interface is public or abstract.

Interface keyword is used to declare an interface

A class can implements an interface

An Interface does not have any constructor

In java 8 we also define static method inside interface

A class can implements more than one interface at a time
If any class implements an interface then must override all interface methods

Untitled - Paint

Syntax:

```
interface IA{
   void a();
}

class Test implements IA{
     void a(){
  sop("This  is IA interface method");
    }
 public void hello(){
 sop("This is Test class hello method");
}

}
```

IA (Interface)

void a();

implements

Test

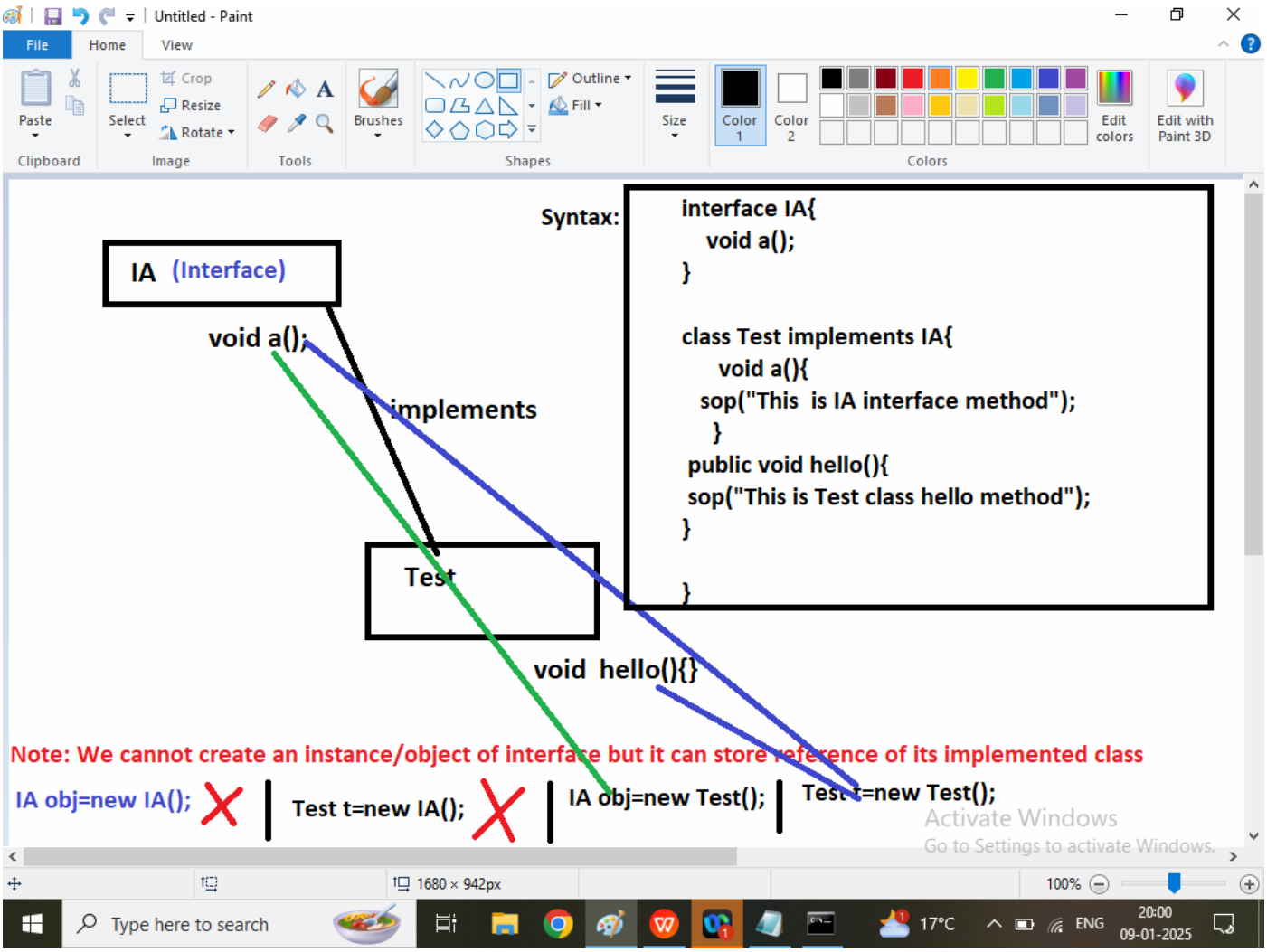
void hello(){}

Note: We cannot create an instance/object of interface but it can store reference of its implemented class

IA obj=new IA(); ✗ | Test t=new IA(); ✗ | IA obj=new Test(); | Test t=new Test();

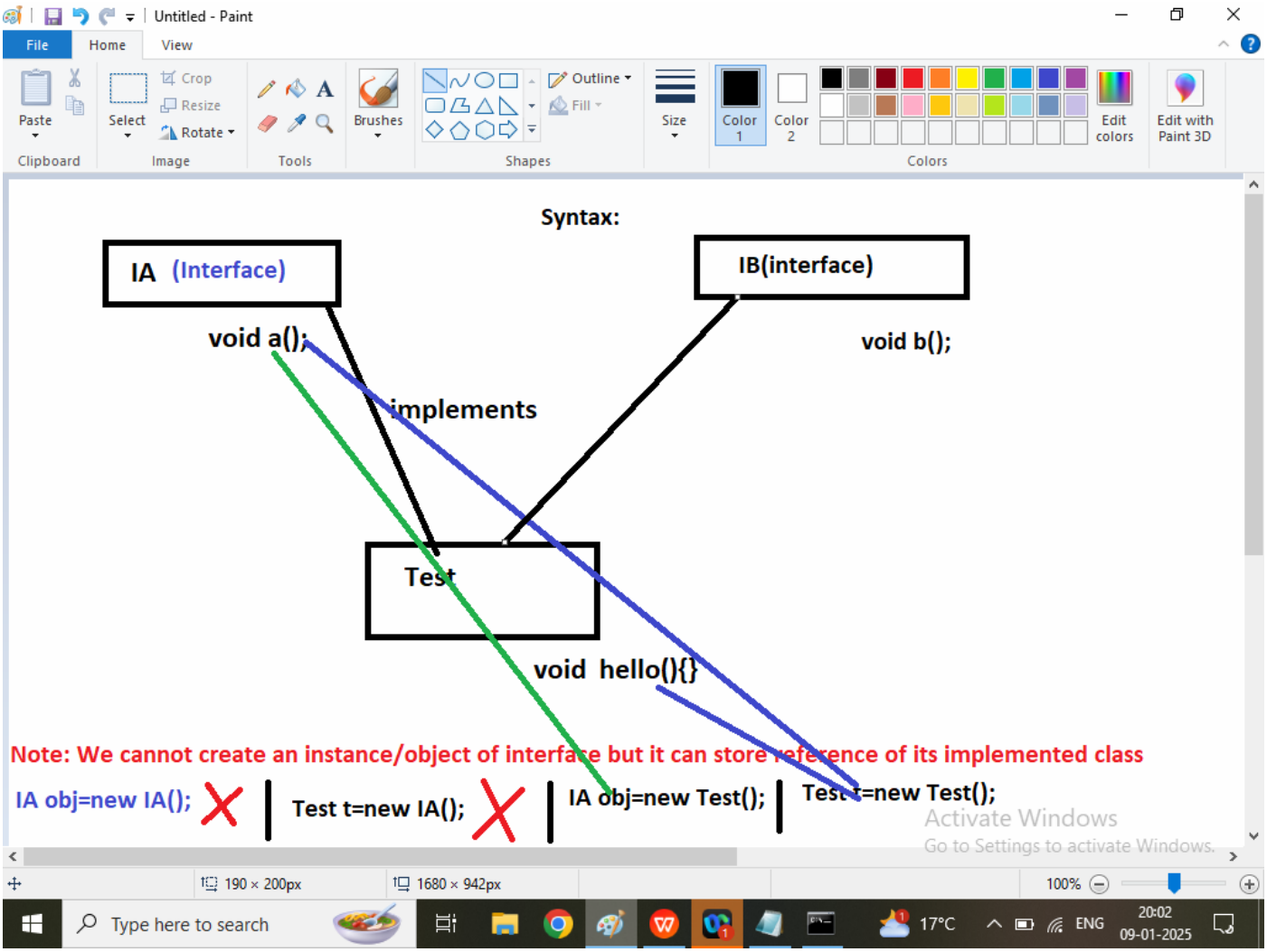
930, 427px | 1680 × 942px | 100%

Type here to search | Earn... | ENG | 19:41 09-01-2025



Example:

A class Implements Multiple Interface




```
interface IA{
        void a();
}
interface IB{
        void b();
}

class Test implements IA,IB{
  public void a(){
  System.out.println("This is IA Interface a  method");
   }
public void b(){
  System.out.println("This is IB Interface b  method");
   }
 public void hello(){
  System.out.println("This is Test class hello  method");
   }
  public static void main(String args[]){
        //IA obj=new IA();
        //Test obj=new IA();
        IA obj=new Test();
    obj.a();
        //obj.hello();
        Test t=new Test();
        t.a();
        t.b();
        t.hello();
  }
}
```

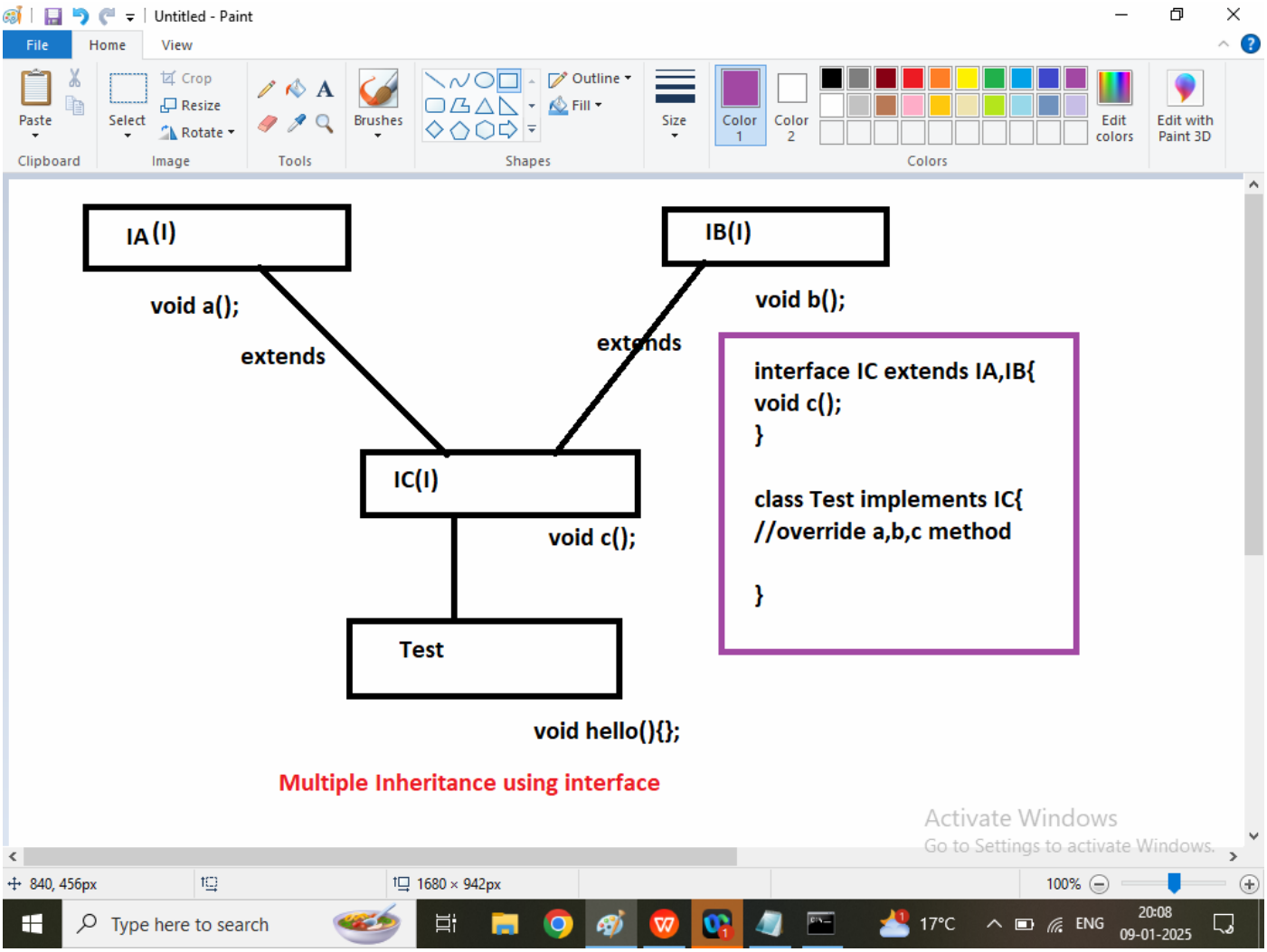
Q3.Multiple Inheritance in java programming?

Ans: There are no way to achieve Multiple Inheritance Through the class but it can be achieve using interface.

One Interface can extends More than one Interface in java Programming

Syntax:

```
interface IA{
Void a();
}
interface IB{
Void b();
}
```



```
interface IA{
        void a();
}
interface IB{
        void b();
}
interface IC extends IA,IB{
        void c();
}

class Test implements IC{
  public void a(){
  System.out.println("This is IA Interface a  method");
   }
public void b(){
  System.out.println("This is IB Interface b  method");
   }
public void c(){
  System.out.println("This is IC Interface c  method");
   }
 public void hello(){
  System.out.println("This is Test class hello  method");
   }
  public static void main(String args[]){
        //IA obj=new IA();
        //Test obj=new IA();
        IA obj=new Test();
     obj.a();
        //obj.hello();
        Test t=new Test();
        t.a();
        t.b();
        t.c();
        t.hello();
  }
}
```

Q1. Explain Exception Handling in java Programming?

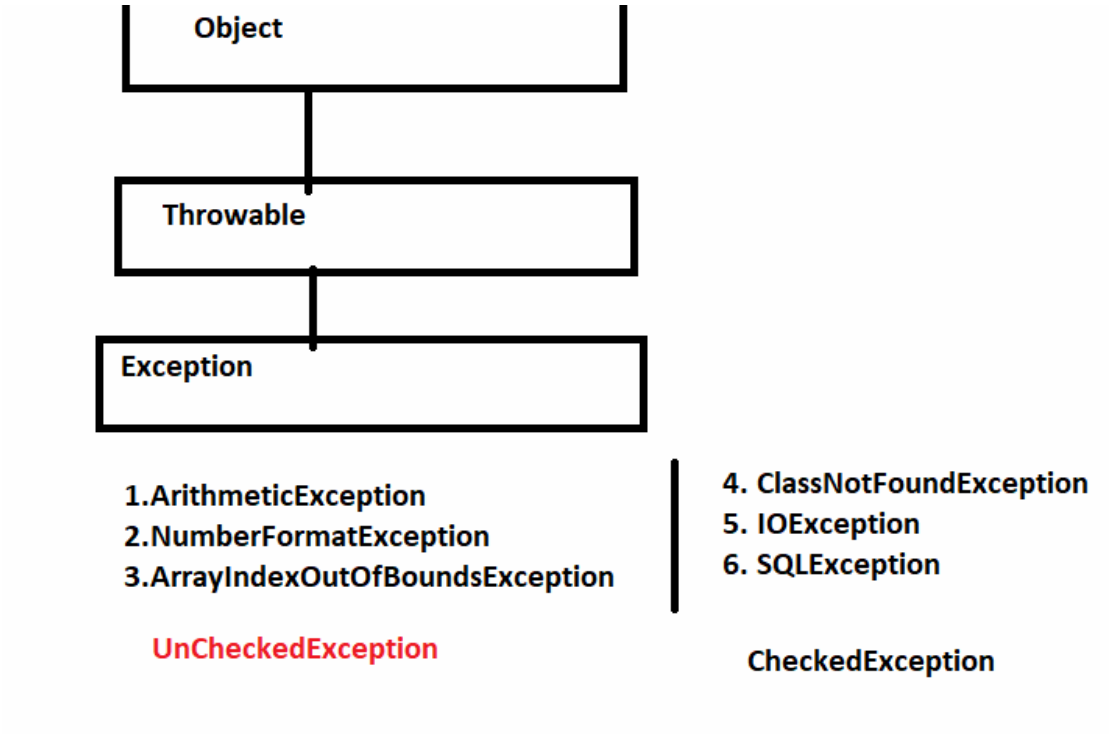
Ans: Exception Handling in java is a mechanism to handle run time errors(exception) in order maintain normal flow of the program.It helps in managing exceptions(unexpected event) and avoiding termination of the program during execution

Throwable: The super class for all exceptions and error.

Exception: It is a child class of Throwable Represents a conditions that a program might want to catch.

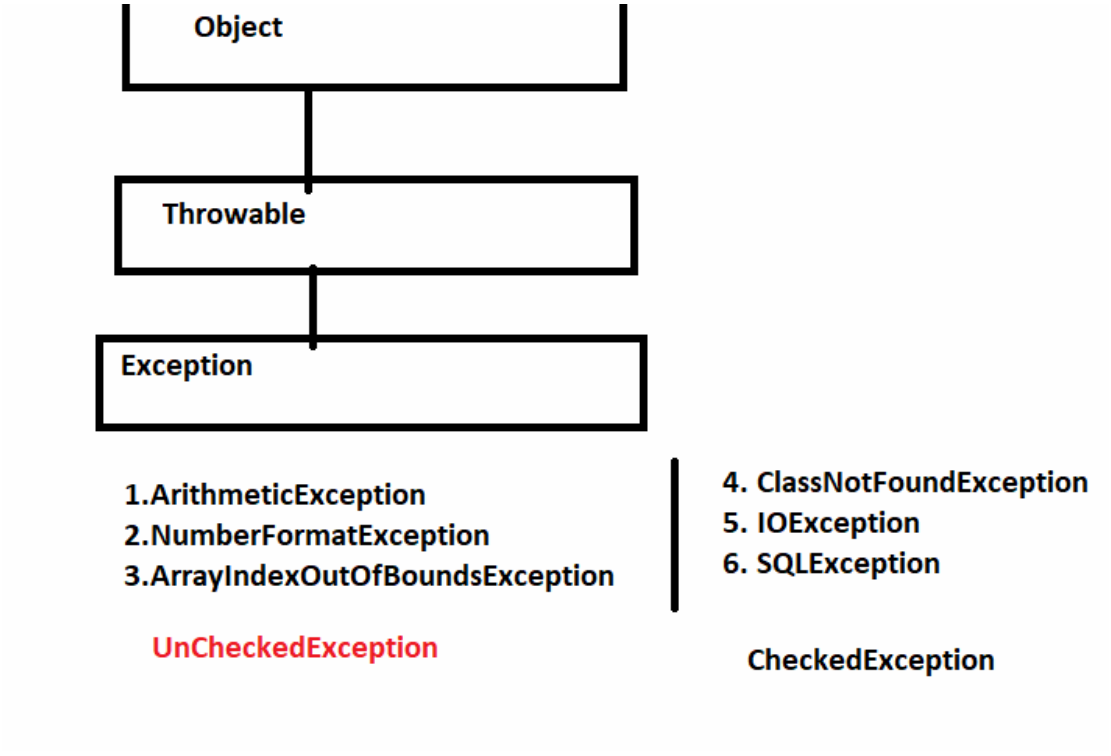
There are two types Exception
1. Checked Exception : Handle to Mandatory

2. Unchecked Exception: Handle to Optional



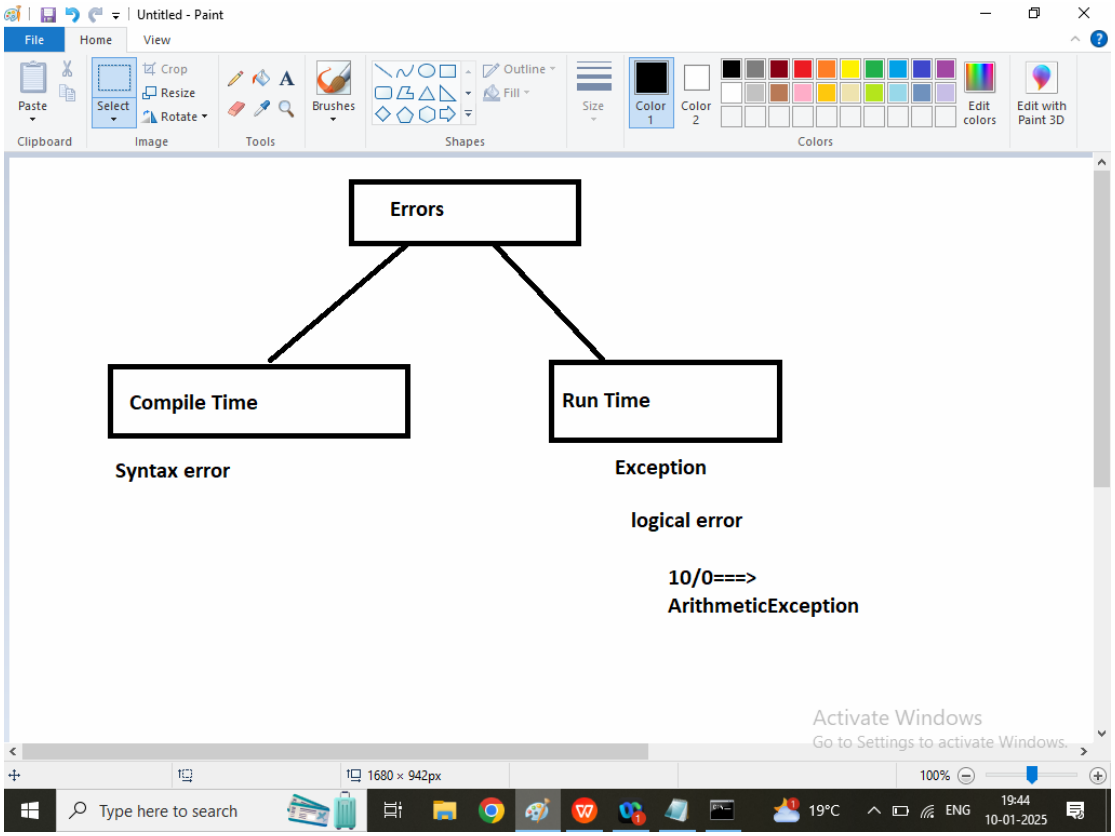
There are two types of Exceptions
1. Pre-Defined Exception
2. User Defined Exception

Errors: Represents serious problem that application should not attempt to catch(outOfMemoryError, stackOverFlow)



```
class E1{
        public  static void main(String args[]){
        System.out.println("Hello...hi");
        int a,b,c;
        a=10;
        b=0;
        System.out.println("before Exception");
        c=a/b;
        System.out.println(c);
        System.out.println("Hello...Bye");
        System.out.println("After Exception");

        }
}
```

```
class E1{
        public  static void main(String args[]){
        System.out.println("Hello...hi");
        int a,b,c;
        a=10;
        b=0;
        System.out.println("before Exception");
        c=a/b;
        System.out.println(c);
        System.out.println("Hello...Bye");
        System.out.println("After Exception");

        }
}
```



There are 5 keywords to handle exception in java

1. try:
2. catch:
3. finally
4. throw
5. throws

Syntax: (try and catch)

```
try{
//code that result may be exception

}catch(Exception or Its Derived Class){
//handling code

}
```

Syntax: (try..finally)

```
try{
//code
}finally{

}
```

```
try{
//code
}catch(){
//handling codetry{
//code
}catch(){
//
}

finally{

}
```

Q1. Explain Exception Handling Keywords in java?

1. try:defines the block of code to detection of exception
2. catch: defines a block of code to handle specific exception
3. finally: defines block of code that always will be executed. finally is unconditional, we can use only one finally after try
4. throw: used to explicitly throw an exception
5. throws: declares exceptions that a method can throw to the caller.

```
class E1{
    public  static void main(String args[]){
    System.out.println("Hello...hi");
    int a,b,c=0;
    a=10;
    b=2;
    System.out.println("before
Exception");
```

```
	try{
	System.out.println("Enter Try");
	c=a/b;

	System.out.println("Exit try");
	}catch(ArithmeticException ae){
	System.out.println("This is catch
block");
	System.out.println("Denominator
should not be zero");
	}
	System.out.println(c);
	System.out.println("Hello...Bye");
	System.out.println("After Exception");

	}
 }
```

```
class E1{
	public  static void main(String args[]){
	System.out.println("Hello...hi");
	int a,b,c=0;
	a=10;
	b=0;
```

```
	System.out.println("before
Exception");
	try{
	System.out.println("Enter Try");
	c=a/b;

	System.out.println("Exit try");
	}catch(ArithmeticException ae){
	System.out.println("This is catch
block");
	System.out.println("Denominator
should not be zero");
	}
	System.out.println(c);
	System.out.println("Hello...Bye");
	System.out.println("After Exception");

	}
 }
```

```
class E1{
	public  static void main(String args[]){
	System.out.println("Hello...hi");
	int a,b,c=0;
```

```
    a=10;
    b=2;
    System.out.println("before
Exception");
    try{
    System.out.println("Enter Try");
    c=a/b;

    System.out.println("Exit try");
    }finally{
    System.out.println("This is finally
block");

    System.out.println(c);
    System.out.println("Hello...Bye");
    System.out.println("After Exception");
    }
    }
 }
```

```
class E1{
    public  static void main(String args[]){
    System.out.println("Hello...hi");
    int a,b,c=0;
```

```
    a=10;
    b=0;
    System.out.println("before
Exception");
    try{
    System.out.println("Enter Try");
    c=a/b;

    System.out.println("Exit try");
    }finally{
    System.out.println("This is finally
block");

    System.out.println(c);
    System.out.println("Hello...Bye");
    System.out.println("After Exception");
    }
    }
 }
```

```
class E1{
    public  static void main(String args[]){
    System.out.println("Hello...hi");
    int a,b,c=0;
```

```
    a=10;
    b=0;
    System.out.println("before
Exception");
    try{
    System.out.println("Enter Try");
    c=a/b;

    System.out.println("Exit try");
    }finally{
    System.out.println("This is finally
block");

    System.out.println(c);
    System.out.println("Hello...Bye");
    System.out.println("After Exception");
    }
    }
 }
```

Q1. Explain Command Line Argument in java Programming?

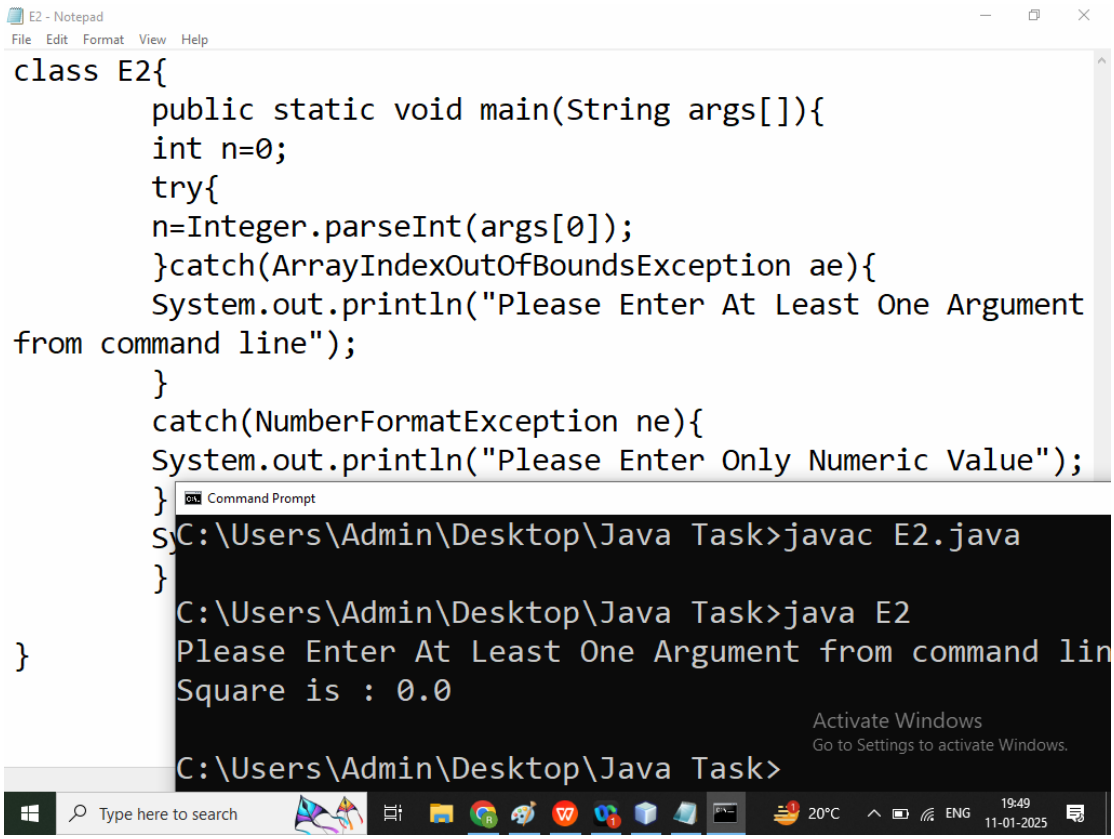
Ans: Command Line Argument is used to allow to pass information to a program at the time of execution of the Program

Syntax:

java ClassName Hello Hi How Are You

java ClassName 1 2 3 4 5

Java ClassName



E2 - Notepad

File Edit Format View Help

```
class E2{
        public static void main(String args[]){
        int n=0;
        try{
        n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
        System.out.println("Please Enter At Least One Argument
from command line");
        }
        catch(NumberFormatException ne){
        System.out.println("Please Enter Only Numeric Value");
        }
}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>java E2 five 2 3
Please Enter Only Numeric Value
Square is : 0.0

C:\Users\Admin\Desktop\Java Task>
```

Activate Windows
Go to Settings to activate Windows.

Type here to search 20°C ENG 19:50 11-01-2025

E2 - Notepad

File Edit Format View Help

```
class E2{
        public static void main(String args[]){
        int n=0;
        try{
        n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
        System.out.println("Please Enter At Least One Argument
from command line");
        }
        catch(NumberFormatException ne){
        System.out.println("Please Enter Only Numeric Value");
        }
        System.out.println("Square is : "+Math.pow(n,2));
        }
}
```

Command Prompt

```
C:\Users\Admin\Desktop\Java Task>java E2 5
Square is : 25.0

C:\Users\Admin\Desktop\Java Task>
```

Activate Windows
Go to Settings to activate Windows.

11 January 2025
Saturday

Type here to search 20°C ENG 19:50 11-01-2025

```
class E2{
        public static void main(String args[]){
        int n=0;
        try{
```

```
        n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
        System.out.println("Please Enter At Least One Argument from command line");
        }
        catch(NumberFormatException ne){
        System.out.println("Please Enter Only Numeric Value");
        }
        System.out.println("Square is : "+Math.pow(n,2));
        }

}
```

```
class E2{
        public static void main(String args[]){
        int n=0;
        try{
        n=Integer.parseInt(args[0]);
        }catch(ArrayIndexOutOfBoundsException ae){
        System.out.println("Please Enter At Least One Argument from command line");
        ae.printStackTrace();

        }
        catch(NumberFormatException ne){
        System.out.println("Please Enter Only Numeric Value");
        ne.printStackTrace();

        }
        System.out.println("Square is : "+Math.pow(n,2));
        }

}
```

Q1. Explain User Defined Exception class or Customize Exception class in java Programming?
Ans: In java if you are not satisfied from the pr-defined exception then you also create your own Exception class/ User Defined Exception/ Customize Exception

You can create a User defined Exception or Customize Exception by extending Throwable or Exception Class

Steps to create a User Defined Class in java

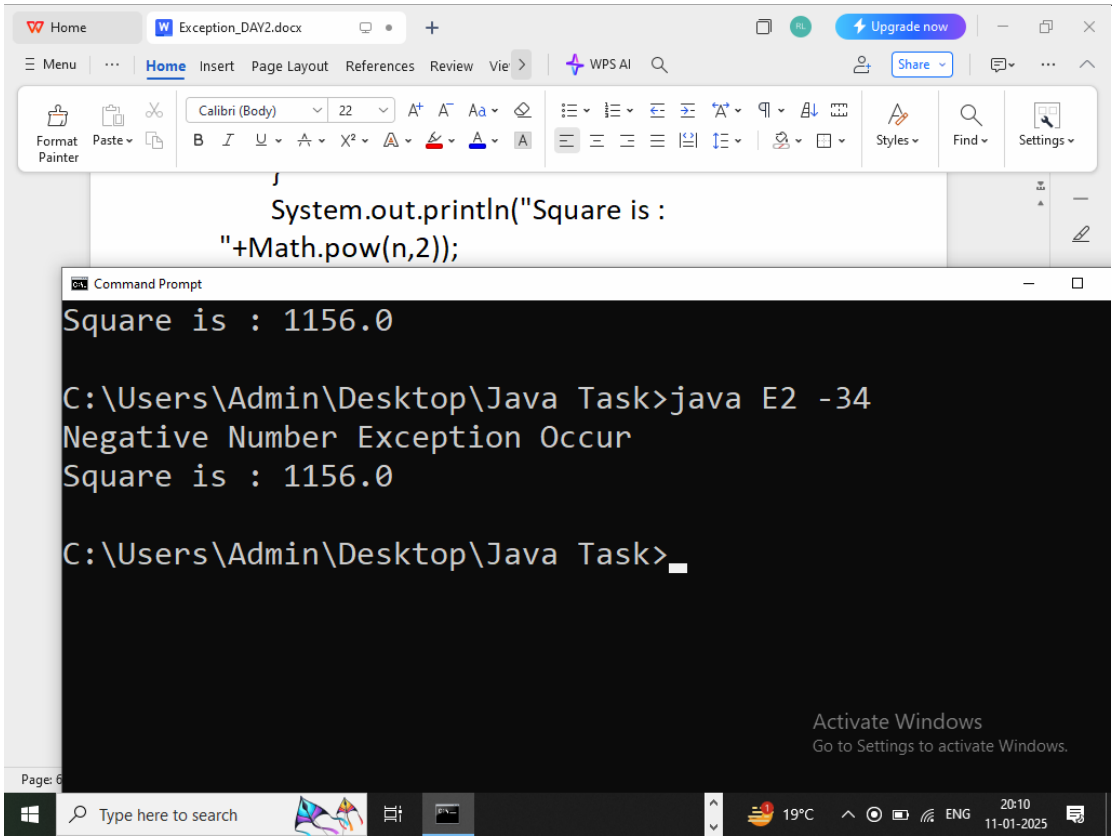
1. Create a Custom Class/ User Defined Class for the exception by extending Throwable or Exception(define Member data String)
2. Define a Parameterized Constructor and method to return message
3. Throw the Exception from where you handle it.
4. Handle The Exception using try catch

```
class NegativeException extends Throwable{
private String msg;
public NegativeException(String msg){
this.msg=msg;
}

public String getMsg(){
return msg;
}

}
```

```
class E2{
	public static void main(String args[]){
	int n=0;
	try{
	n=Integer.parseInt(args[0]);
	if(n<0){
```

```
	NegativeException x=new
NegativeException("Negative Number
Exception Occur");
	throw  x;
	}
	}catch(ArrayIndexOutOfBoundsException
ae){
	System.out.println("Please Enter At Least
One Argument from command line");
	ae.printStackTrace();

	}
	catch(NumberFormatException ne){
	System.out.println("Please Enter Only
Numeric Value");
	ne.printStackTrace();

	}
	catch(NegativeException y){
	System.out.println(y.getMsg());
	}
	System.out.println("Square is :
"+Math.pow(n,2));
	}
```

}


Exception_DAY2.docx
System.out.println("Square is :
"+Math.pow(n,2));
Command Prompt
Square is : 1156.0
C:\Users\Admin\Desktop\Java Task>java E2 -34
Negative Number Exception Occur
Square is : 1156.0
C:\Users\Admin\Desktop\Java Task>
Activate Windows
Go to Settings to activate Windows.

Syllabus : Object Oriented Programming in java

1. Class
2. Object
3. Member data
4. Member Function
5. Access specifier
6. Constructor
7. Default Constructor
8. Parameterized constructor
9. Encapsulation
10. Inheritance
11. This
12. Super
13. Static variable
14. Static block
15. Static method
16. Final class
17. Final variable
18. Final method
19. Polymorphism
20. Variable argument (…)

21. Compile Time Polymorphism(Method Overloading)
22. Run Time Polymorphism(Method Overriding)
23. Abstraction
24. Interface
25. Multiple Inheritance Using Interface

Q1. Explain class in java programming?
Ans:
Purpose: It is used to create a user defined data type in java

Definition:
Class is the collection of member data and member function

A class is a set of rules

A class is an blue print of an object

Q2. Explain Object?
Ans: Object is an variable / instance of a class

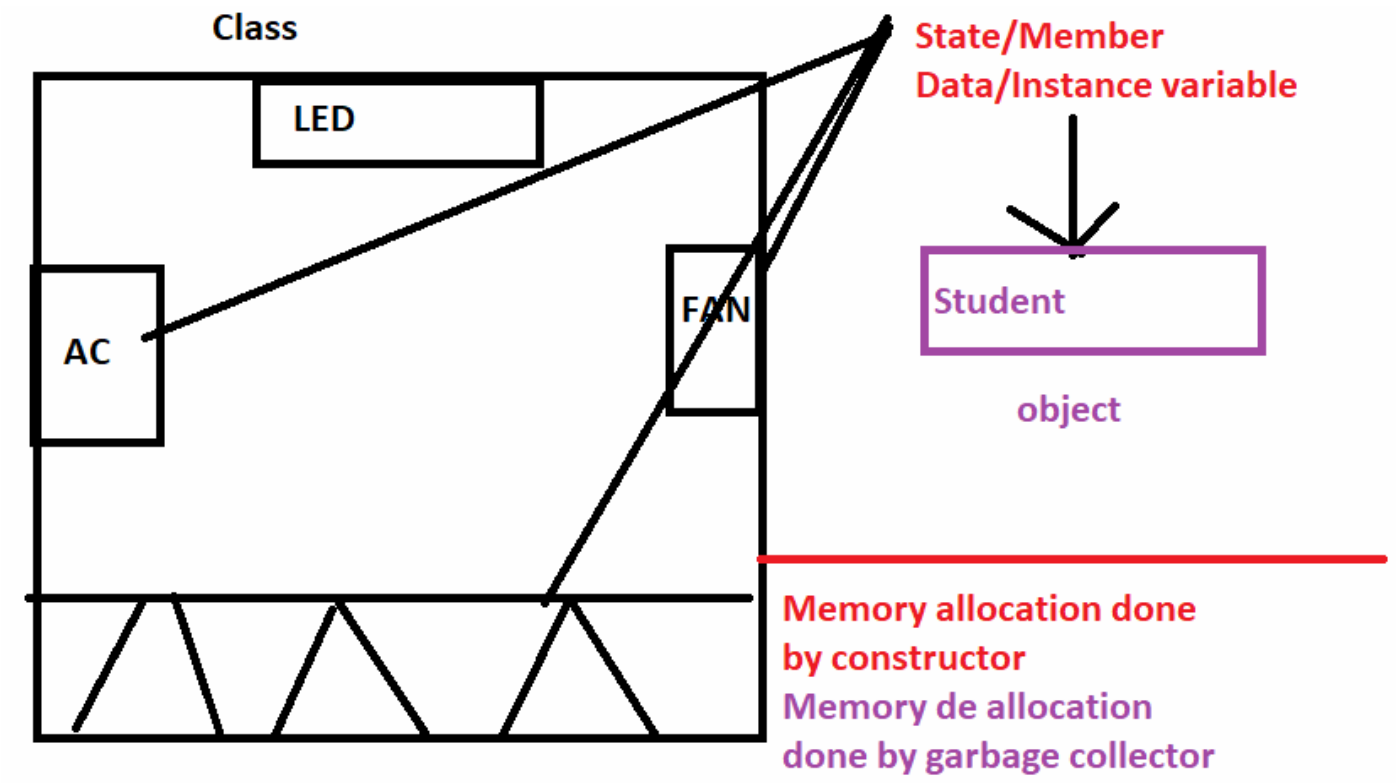
If we want to access instance variable / instance method of class then we should use object

How to create an object of the class
ClassName obj=new ClassName();
String s1=new String();
Scanner s2=new Scanner(System.in);
Student st=new Student();

Access Member data using Object
objectName.memberDataName;
Access Member function using Object
objectName.memberFunctionName();

Note:

1. Method name should be written in cameCase
2. MethodName startWith Verb



Untitled - Paint

```
class Rectangle{

psvm(){
  //declare 4 variable   int l,b;

  //input
    output
  //operation

  //print result

}
```

```
class Rectangle{

int l;
int b;        Member data/state/instance variable

void acceptData(){}

void showData(){}

void getArea(){}

void getPira(){}
psvm(){
 Rectangle r=new Rectangle();

}
```

```
r.l;
r.b;
r.acceptData();
r.showData();
r.getArea();
r.getPira();
```

meet.google.com is sharing your screen. Stop sharing Hide

779, 284px 1680 × 942px 100%

Type here to search 18°C ENG 20:03 21-12-2024

Q1. Explain class in java?
Ans:
Purpose: It is used to create a user defined data type in java

Definition: Class is the collection member data and Member Function

A class is a set of rules

A class is a blue print of an object

Member Data: Member data can specify what type data can be occur
Member function: Mediator between user and data

Q2. Write a java program to create a rectangle class and perform the operations?

```
import java.util.Scanner;
class Rectangle{
int l;
int b;
// instance variable,member data, state
void acceptData(){
Scanner sk=new Scanner(System.in);
System.out.println("Enter Length : ");
l=sk.nextInt();
System.out.println("Enter Breadth : ");
b=sk.nextInt();
}

void showData(){
System.out.println("Length is : "+l);
System.out.println("Breadth is : "+b);
}

void getArea(){
```

```
System.out.println("Area : "+(l*b));
}

void getPira(){
System.out.println("Pirameter of Rectangle : "+2*(l+b));
}
//Member function, Instance method

public static void main(String args[]){
Rectangle r=new  Rectangle();//r is an object of Rectangle class
r.acceptData();
r.showData();
r.getArea();
r.getPira();
}
}
```

Q3. Write a java Program to calculate area and circumference of Circle Using class?

```
class Circle{

   float r;

void acceptData(){}

void showData(){}

void getArea(){}

void getCirc(){}



}
```

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

   Circle c1=new Circle();
   c1.acceptData();
   c1.showData();
   c1.getArea();
   c1.getCirc();



}
```

```
import java.util.Scanner;
class Circle{
float r;
// instance variable,member data, state
void acceptData(){
Scanner sk=new Scanner(System.in);
System.out.println("Enter Radius: ");
r=sk.nextFloat();
}

void showData(){
System.out.println("Radius is : "+r);
}

void getArea(){
System.out.println("Area : "+Math.PI*Math.pow(r,2));
}

void getCirc(){
System.out.println("Circumference : "+2*Math.PI*r);
}
//Member function, Instance method

public static void main(String args[]){
Circle r=new Circle();//r is an object of Circle class
r.acceptData();
r.showData();
r.getArea();
r.getCirc();
}
}
```

Q3. Write a java Program to create a student class and perform following operations?

Enter Name: Akash
Enter Enrollment : 0103CS
Enter P : 67
Enter C :78
Enter M : 87
Enter H: 67
Enter E: 55

Total Marks:
Percentage:

```
class Student{
String name;
String enroll;
int p;
int c;
int m;
int h;
int e;
//Member Data

void getData(){}

void showData(){}

int getTotalMarks(){}
float getPercentage(int totalmarks){}
```

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

 Student st=new Student();

 st.getData();
 st.showData();
 int tm=st.getTotalMarks();
 System.out.println("Total Marks : "+tm);
 float p=st.getPercentage(tm);
 System.out.println("Percentage  is : "+p);

}

}
```

Q1. Write a java Program to create a Employee class and perform following Operations

```
class Employee{
String name;
int empno;
float sal;
//Member Data
void acceptData(){}
void showData(){}
double getHRA(){}
double getDA(){}
double getTA(){}

double grossSalary(){}

double getTotalIncentive(){}
```

```
public static void main(String args[]){
Employee e=new Employee();
e.acceptData();
e.showData();
sop("HRA : "+e.getHRA());
sop("DA : "+e.getDA());
sop("TA : "+e.getTA());
sop("All Incentive : "+e.getTotalIncentive());
sop("Gross Salary : "+e.getgrossSalary());

 }
```

Q2. Explain Access Specifier in Java Programming?

Ans: Access Specifier can specify the scope of member data, member function, class and interface

There are 4 access specifier available in java

1. **private:** It can access only inside the class

Recommended for : member data

2. **public:** It can access anywhere

Recommended: Member Function,constructor, Class, interface

3. **protected:** It can access inside the package and its child class

Recommended: Member data

4. **default**(No Access Specifier is by default =default) It can access only inside particular package/folder/directory

Access specifier	Inside	Inside package	Child Class	Outside world
private	YES	NO	NO	NO
public	YES	YES	YES	YES
protected	YES	YES	YES	NO
default	YES	YES	YES	NO

Q2. Explain Constructor in java Programming?
Ans:
Constructor is a special member function in a class it is used to initialize a user defined data type

Rules
1. A class Name and Constructor Name must be same
2. A constructor does not have any return type even void
3. A class can have more than one constructor it means constructor can be overloaded
4. A constructor cannot be override
5. A constructor cannot be static
6. If we does not write any constructor then compiler automatically add a default constructor
7. If we write any constructor in a class then compiler will not add any type(default or parameterized) constructor
8. Constructor is automatically called when object created by new

Task of Constructor
1. Allocate memory of all member data (HEAP)
2. Assign default value to the member data based on default value

Note:
1. Java Does not support Destructor
2. Garbage collector always destroy the un allocated memory from the heap

Types of Constructor
In java 2 types constructor is available
1. Default Constructor (Parameter less)
2. Parameterized Constructor

Q1. Explain Inheritance in java Programming?
Ans : Passing Properties from one class to another class is known as inheritance

Properties may be member data or member function

A class who gives the properties known as Parent / Super / Base class

A class who receives the properties are known as Sub / derived / Child class

In case of Inheritance Member Data or member function of Parent class should not be private

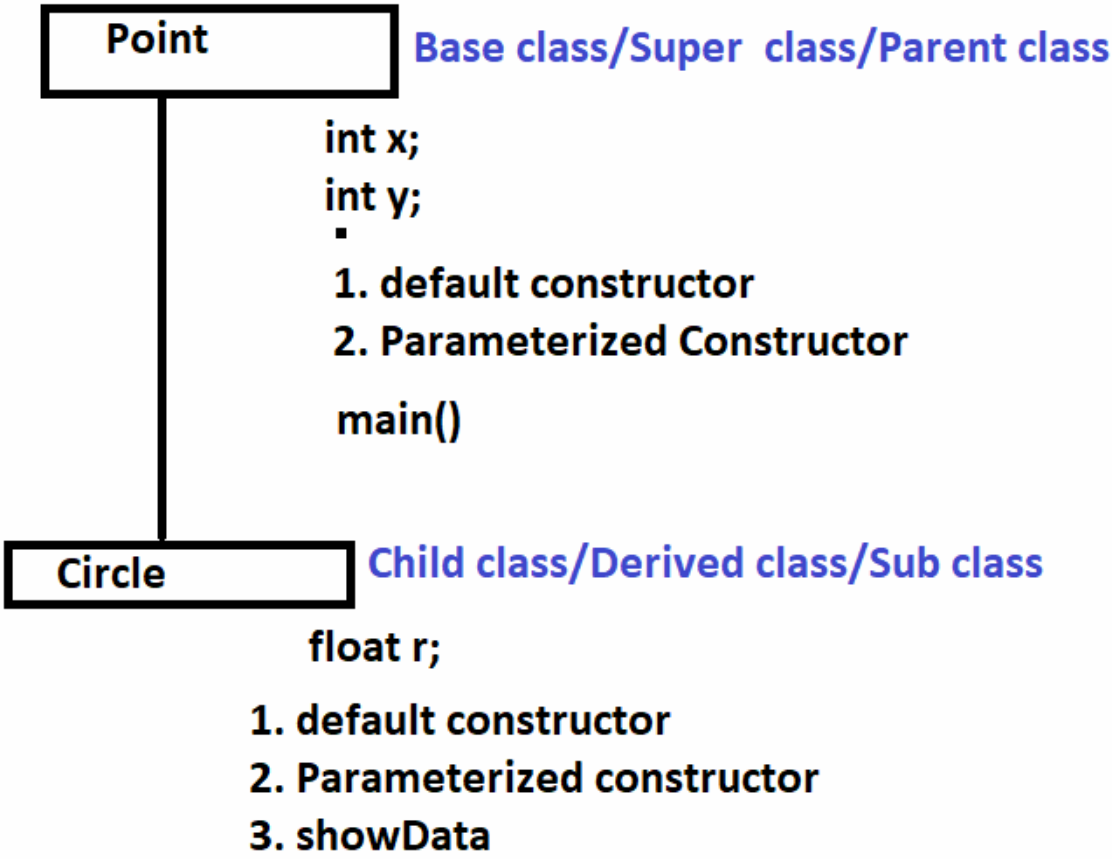
extends keyword to use inherit the class

Java does not support multiple inheritance using class

Syntax:

```
class Parent{
//member data
//member function
}
```

```
class Child extends Parent{
//Member data
//Member Function

}
```



Q2. Explain this keyword in java Programming?
Ans:
This keyword represent current class objects.

When an instance variable name is same as local variable (formal parameter). this can be used to refer to the instance variable of a class explicitly

Example:

```
class Point{
public int x;
public int y;//instance variable

public Point(){
System.out.println("Point class Default Constructor is
called");
}

public Point(int x,int y){
this.x=x;
this.y=y;
System.out.printf("x=%d  Y=%d ",x,y);
System.out.println("Point class Parameterized Constructor
is called");
}

void showData(){
System.out.println("X_CO : "+x);
System.out.println("Y_CO : "+y);

}
public static void main(String args[]){
System.out.println("Point class Main Method");
Point p2=new Point(10,20);
p2.showData();
}
```

```
}
class Circle extends Point{



}
```

1. This keyword can be used to call current class method

Syntax:
this.methodName()

2. This keyword can be used to call current class Constructor

Syntax:
this()
this(10,20)

```
class Point{
public int x;
public int y;//instance variable

public Point(){
System.out.println("Point class Default Constructor is called");
```

```
}

public Point(int x,int y){
this();//to call current class default constructor
this.x=x;
this.y=y;

System.out.printf("\nx=%d  Y=%d ",x,y);
System.out.println("Point class Parameterized Constructor
is called");
}

void showData(){
System.out.println("X_CO : "+x);
System.out.println("Y_CO : "+y);

}
void hi(){
System.out.println("Hi... Method is Called");
this.showData();
int x=111;
int y=222;
System.out.printf("\nx=%d  Y=%d ",this.x,this.y);


}
public static void main(String args[]){
System.out.println("Point class Main Method");
```

```
Point p2=new Point(10,20);
p2.hi();
}

}
class Circle extends Point{



}
```
