Day 30

Q1. Explain Class and Object in java programming?

Ans:

**Purpose:** It is used to create a user defined data type in java programming

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

Class is a set of rules

**Member Data:** Member data can specify what type data can be occur

**Member Function:** Member function can act mediator between user and data.

Syntax:

class ClassName{

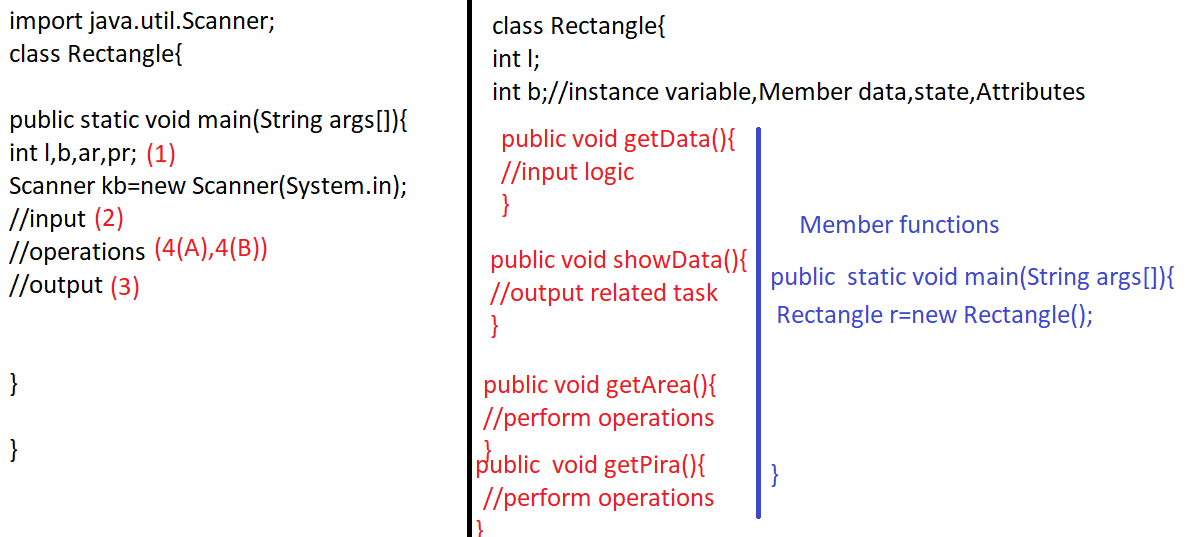
//member data

//Member function

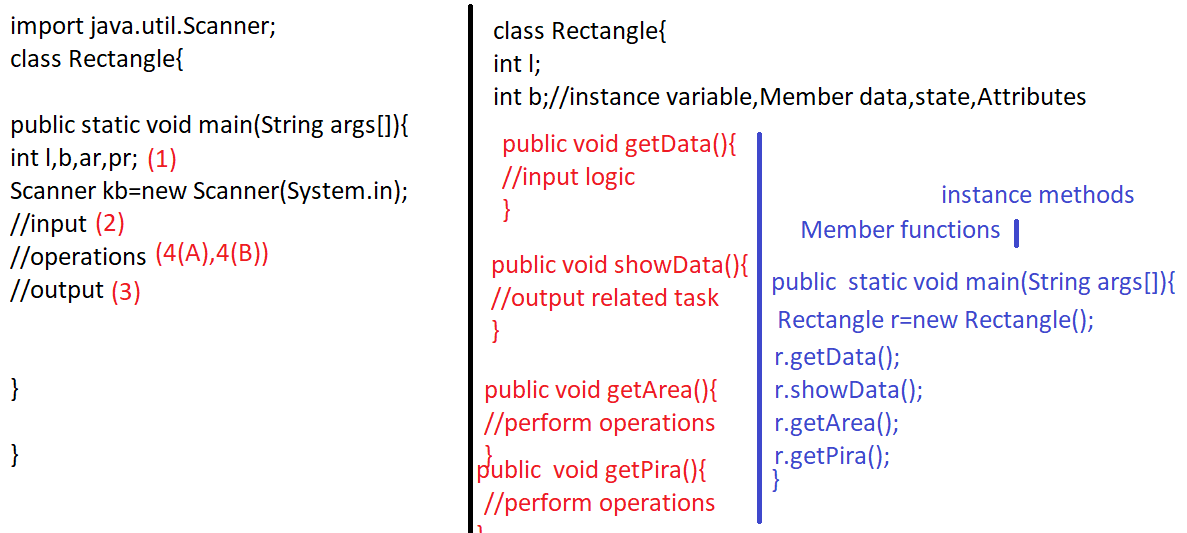
}

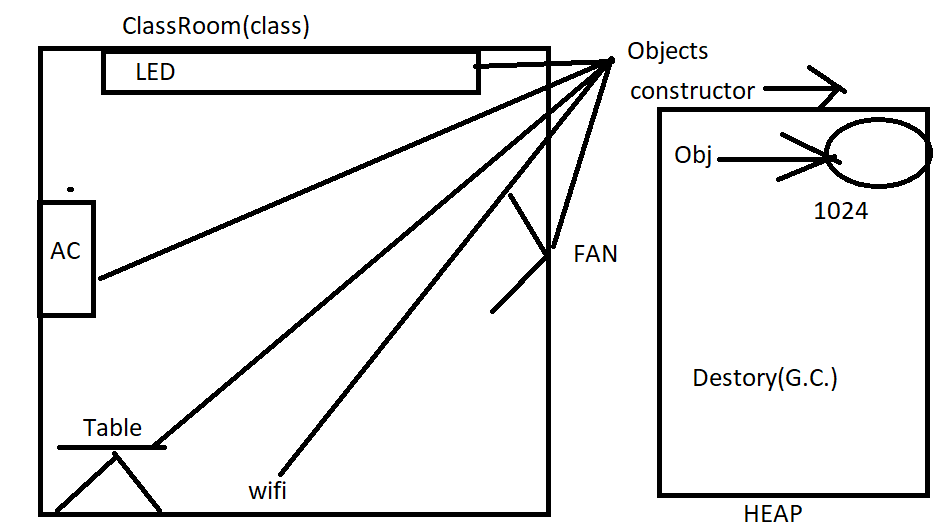
How to create object of a class

ClassName objName=new ClassName();



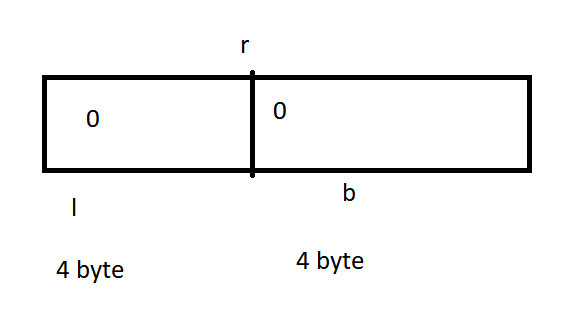
How to call methods of a class objName.methodName();

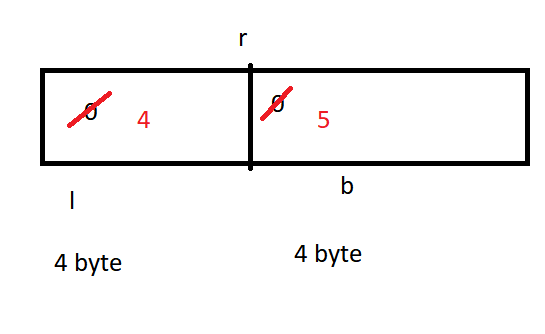




Day 31

Q1. Write a java program to calculate area pirameter of rectangle using class





Q1. Write a java program to create student class and perform following operations?

|  |
| --- |
| import java.util.Scanner; // Importing the Scanner class for user input  // Class definition  class Student {  // Instance variables  String name;  String enroll;  float per;  int rollno;  char section;  int p; // Marks in subject P  int c; // Marks in subject C  int m; // Marks in subject M  int e; // Marks in subject E  int h; // Marks in subject H  // Method to get data from user  public void getData() {  Scanner obj = new Scanner(System.in); // Creating Scanner object for input  System.out.print("Enter Name: ");  name = obj.nextLine(); // Getting name from user  System.out.print("Enter Enroll: ");  enroll = obj.nextLine(); // Getting enrollment number from user  System.out.print("Enter Section: ");  section = obj.next().charAt(0); // Getting section from user  System.out.print("Enter P: ");  p = obj.nextInt(); // Getting marks in subject P from user  System.out.print("Enter C: ");  c = obj.nextInt(); // Getting marks in subject C from user  System.out.print("Enter M: ");  m = obj.nextInt(); // Getting marks in subject M from user  System.out.print("Enter E: ");  e = obj.nextInt(); // Getting marks in subject E from user  System.out.print("Enter H: ");  h = obj.nextInt(); // Getting marks in subject H from user  }  // Method to show student data  public void showData() {  // Printing student information  System.out.println("Name: " + name);  System.out.println("Enroll: " + enroll);  System.out.println("Section: " + section);  System.out.println("P: " + p);  System.out.println("C: " + c);  System.out.println("M: " + m);  System.out.println("E: " + e);  System.out.println("H: " + h);  System.out.println("Per: " + per); // This line shows the percentage, but it isn't calculated in this method  }  // Method to calculate and display total marks  public void getTotalmarks() {  System.out.println("Total Marks: " + (p + c + m + e + h)); // Adding marks of all subjects and displaying the total  }  // Method to calculate and display percentage  public void getPercentage() {  per = (p + c + m + e + h) / 5.0f; // Calculating percentage  System.out.println("Per: " + per + "%"); // Displaying percentage  }  // Main method  public static void main(String args[]) {  Student obj = new Student(); // Creating an object of Student class  obj.getData(); // Calling method to get data from user  obj.showData(); // Calling method to display student data  obj.getTotalmarks(); // Calling method to display total marks  obj.getPercentage(); // Calling method to display percentage  }  } |

Q2. Write a java program to create a employee class and perform some basic operations

|  |
| --- |
| import java.util.Scanner; // Importing Scanner class for user input  // Class definition  class EmpInfo {  // Instance variables  String name;  String job;  double bsal;  int deptno;  int hra, da, ta, gs, tn; // Variables for different components of salary  // Method to get data from user  public void getData() {  Scanner obj = new Scanner(System.in); // Creating Scanner object for input  System.out.print("Enter Name = ");  name = obj.nextLine(); // Getting name from user  System.out.print("Enter job = ");  job = obj.nextLine(); // Getting job from user  System.out.print("Enter bsal = ");  bsal = obj.nextFloat(); // Getting basic salary from user  System.out.print("Enter deptno = ");  deptno = obj.nextInt(); // Getting department number from user  }  // Method to show employee data  public void showData() {  System.out.println("Your Name = " + name); // Displaying name  System.out.println("Your Job = " + job); // Displaying job  System.out.println("Your Bsal = " + bsal); // Displaying basic salary  System.out.println("Your DeptNo = " + deptno); // Displaying department number  }  // Method to calculate HRA (House Rent Allowance)  public double getHRA() {  hra = (int) (bsal \* 0.20); // Calculating HRA as 20% of basic salary  return hra;  }  // Method to calculate DA (Dearness Allowance)  public double getDA() {  da = (int) (bsal \* 0.10); // Calculating DA as 10% of basic salary  return da;  }  // Method to calculate TA (Travel Allowance)  public double getTA() {  ta = (int) (bsal \* 0.10); // Calculating TA as 10% of basic salary  return ta;  }  // Method to calculate total incentives (HRA + DA + TA)  public double totalIncentive() {  tn = hra + da + ta; // Adding HRA, DA, and TA to get total incentives  return tn;  }  // Method to calculate gross salary (Basic Salary + Total Incentives)  public double grossSalary() {  gs = (int) (bsal + tn); // Adding basic salary and total incentives to get gross salary  return gs;  }  // Main method  public static void main(String args[]) {  EmpInfo obj = new EmpInfo(); // Creating an object of EmpInfo class  System.out.println();  obj.getData(); // Calling method to get data from user  obj.showData(); // Calling method to display employee data  System.out.println("Total HRA: " + obj.getHRA()); // Displaying HRA  System.out.println("Total DA: " + obj.getDA()); // Displaying DA  System.out.println("Total TA: " + obj.getTA()); // Displaying TA  System.out.println("Total Incentive: " + obj.totalIncentive()); // Displaying total incentives  System.out.println("Total Gross Salary: " + obj.grossSalary()); // Displaying gross salary  }  } |

Day 32

Text question tha

Day 33

Q1.Write a program in java to compare two strings

Test Data :

Check the length of two strings:

--------------------------------

Input the 1st string : aabbcc

Input the 2nd string : abcdef

String1: aabbcc

String2: abcdef

Expected Output : Strings are not equal.

Check the length of two strings:

--------------------------------

Input the 1st string : aabbcc

Input the 2nd string : aabbcc

String1: aabbcc

String2: aabbcc

Expected Output : Strings are equal.

import java.util.Scanner;

class T1{

public static void cmpString(String s1,String s2){

if(s1.equals(s2)){

System.out.println("String 1 and String 2 are equal");

}

else{

System.out.println("String 1 and String 2 are not equal");

}

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String 1 : ");

String a=obj.nextLine();

System.out.println("Enter String 2 : ");

String b=obj.nextLine();

cmpString(a,b);

}

}

Q3. Write a program in java to count total number of alphabets, digits and

special characters in a string.

Test Data :

Input the string : Welcome to w3resource.com

Expected Output :

Number of Alphabets in the string is : 21

Number of Digits in the string is : 1

Number of Special characters in the string is : 3

import java.util.Scanner;

class T1{

public static void countAds(String s){

int alp=0,dig=0,sp=0;

for(int i=0;i<s.length();i++){

char c=s.charAt(i);

if(c>='A' && c<='Z'){

alp++;

}

if(c>='a' && c<='z'){

alp++;

}

if(c>='0'&& c<='9'){

dig++;

}

if(c==' ' || c=='.'){

sp++;

}

}

System.out.println("Alphabets : "+alp);

System.out.println("Digits : "+dig);

System.out.println("Special characters : "+sp);

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String : ");

String a=obj.nextLine();

countAds(a);

}

}

Q3. Write a program in Java to count total number of vowel or consonant

in a string.

Test Data :

Input the string : Welcome to w3resource.com

Expected Output :

The total number of vowel in the string is : 9

The total number of consonant in the string is : 12

import java.util.Scanner;

class T1{

public static void countVowelAndConsonent(String s){

int v=0,cn=0;

for(int i=0;i<s.length();i++){

char c=s.charAt(i);

if(c=='a'||c=='i'||c=='u'||c=='o'||c=='e'||c=='A'||c=='I'||c=='U'||c=='O'||c=='E'){

v++;

}

else if(c>='0' && c<='9'||c==' '||c=='.'){

continue;

}

else{

cn++;

}

}

System.out.println("Vowels Are : "+v);

System.out.println("Consonent Are : "+cn);

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String : ");

String a=obj.nextLine();

countVowelAndConsonent(a);

}

}

Q4. 4. Write a program in java to find maximum occurring character in a

string.

Test Data :

Input the string : Welcome to w3resource.com.

Expected Output :

The Highest frequency of character 'e'

appears number of times : 4

import java.util.Scanner;

class T1{ //welcome

public static void mostOccuringCharacter(String str1){

int ctr[] = new int[256];

int l = str1.length();

for (int i = 0; i < l; i++)

ctr[str1.charAt(i)]++;

int max = -1;

char result = ' ';

for (int i = 0; i < l; i++) {

if (max < ctr[str1.charAt(i)]) {

max = ctr[str1.charAt(i)];

result = str1.charAt(i);

}

}

System.out.println("Max occuring character : "+result);

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String : ");

String a=obj.nextLine();

mostOccuringCharacter(a);

}

}

Q6. Write a java program to print most occurring digit from the given number.

Enter Any Number : 1221223

Most Occurring Digit is :2

import java.util.Scanner;

class T1{ //welcome

public static void mostOccuringCharacter(String str1){

int ctr[] = new int[256];

int l = str1.length();

for (int i = 0; i < l; i++)

ctr[str1.charAt(i)]++;

int max = -1;

char result = ' ';

for (int i = 0; i < l; i++) {

if (max < ctr[str1.charAt(i)]) {

max = ctr[str1.charAt(i)];

result = str1.charAt(i);

}

}

System.out.println("Max occuring digits : "+result);

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String : ");

String a=obj.nextLine();

mostOccuringCharacter(a);

}

}

Q2. Write a java program to print unique digits from the given number

Enter Any Number : 1232351

Unique Digit is : 1

import java.util.Scanner;

class T1{

public static void printUniqeDigit(int n){//32123

int freq[]=new int[10];

while(n!=0){//0

int r=n%10;//3

freq[r]++;

n=n/10;

}

System.out.println("Unique Digis from the given number");

for(int i=0;i<freq.length;i++){

if(freq[i]==1){

System.out.print("\t"+i);

}

}

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String : ");

int a=obj.nextInt();

printUniqeDigit(a);

}

}

Q2. Write a java program to print duplicate digits from the given number

Enter Any Number : 1232351

Duplicates Digit is : 2 3 1

import java.util.Scanner;

class T1{

public static void printUniqeDigit(int n){//32123

int freq[]=new int[10];

while(n!=0){//0

int r=n%10;//3

freq[r]++;

n=n/10;

}

System.out.println("duplicates Digits from the given number");

for(int i=0;i<freq.length;i++){

if(freq[i]>1){

System.out.print("\t"+i);

}

}

}

public static void main(String args[]){

Scanner obj=new Scanner(System.in);

System.out.println("Enter String : ");

int a=obj.nextInt();

printUniqeDigit(a);

}

}

Q1. Explain Access specifier in java programming?

Ans: Access specifier can specify the scope of member data, member function, class and interfaces

There are 4 access specifier in java programming

private: It can access only inside the class

Recommended for Member data

We cannot use private access specifier as a class level

public: It can access fro any where

Recommended for methods, constructors, classes and interfaces

protected: It can access inside the class or its child class

Recommended for Member data.

We can not uses protected access specifier as a class level

Default: It can accessible only inside the package

Example public and default access specifer with package

package indore;

import bhopal.A;

class B{

public static void main(String args[]){

A obj=new A();

System.out.println(obj.x);

}

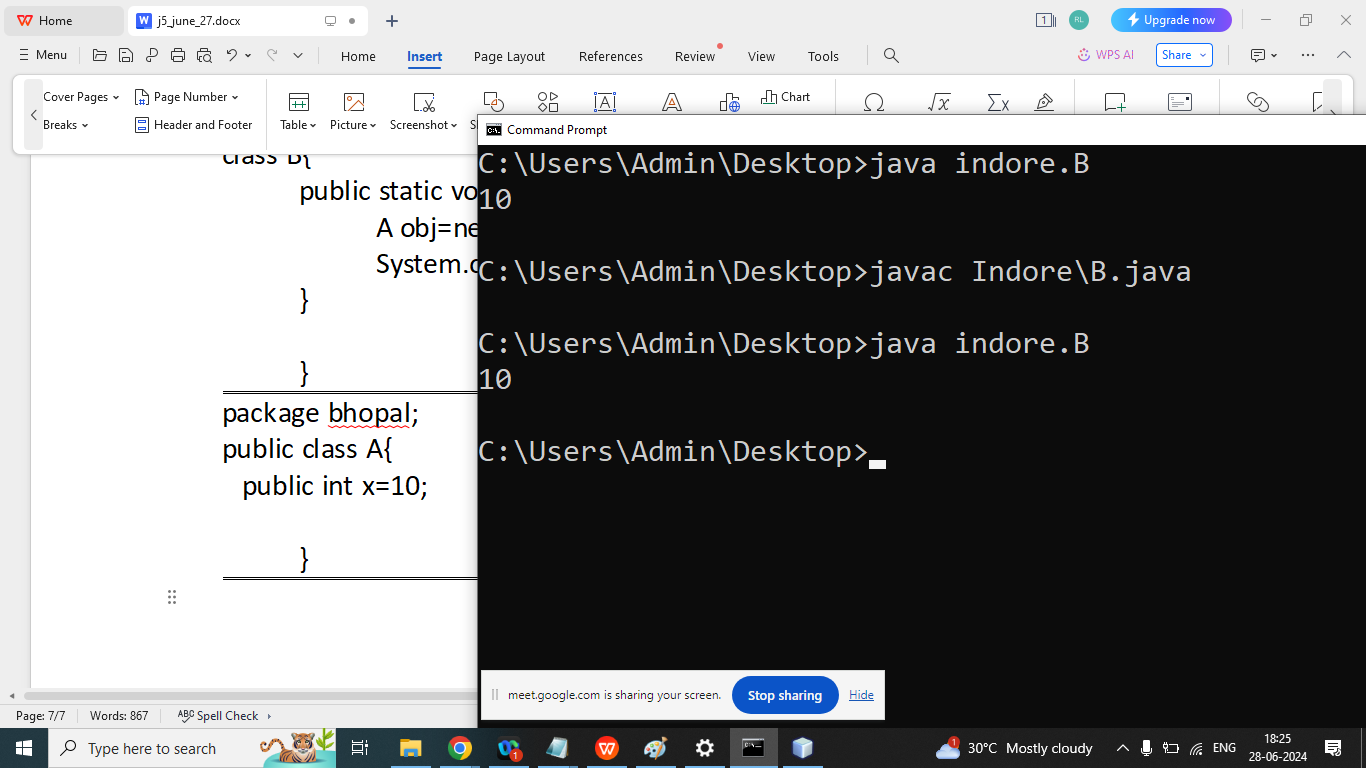
}

package bhopal;

public class A{

public int x=10;

}



package indore;

public class A {

protected int x = 10;

}

Day 34

Q1. Explain Constructor in java programming?

Ans:

A constructor is a special member function of a class it is used to initialize the member data of a class and also done memory allocation inside the heap

Rules of Constructors

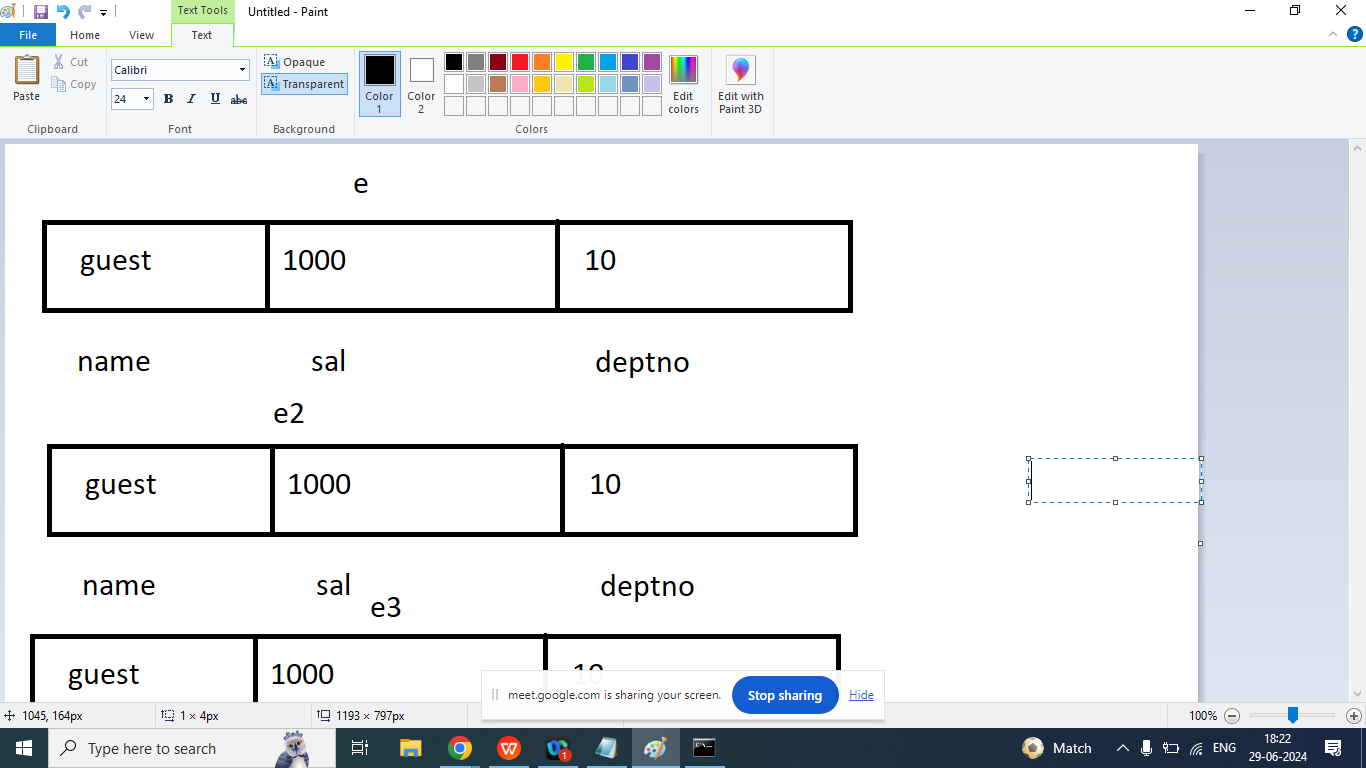
1. A class name and constructor name must be same.
2. Constructor does not have any return type even void
3. Constructor is automatically called when object is created.

**Types of constructors:**

* Default Constructor(Parameter less constructor)
* Parameterized constructor(with parameters)

If we don’t write any constructor in our program then compiler automatically add default constructor

If we write any type of constructor then compiler does not add any constructor in our program



Syntax:

class Emp{

String name;

float sal;

int deptno;

public Emp(){

name=”guest”;

Sal=1000;

deptno=10;

}

}

Example: Default Constructor

class Emp{

String name;

float sal;

int deptno;

public Emp(){

System.out.println("Default constructor is called");

name="guest";

sal=1000;

deptno=10;

}

public void showData(){

System.out.println("Name : "+name);

System.out.println("Salary : "+sal);

System.out.println("Deptno : "+deptno);

}

public static void main(String args[]){

Emp e=new Emp();

e.showData();

Emp e1=new Emp();

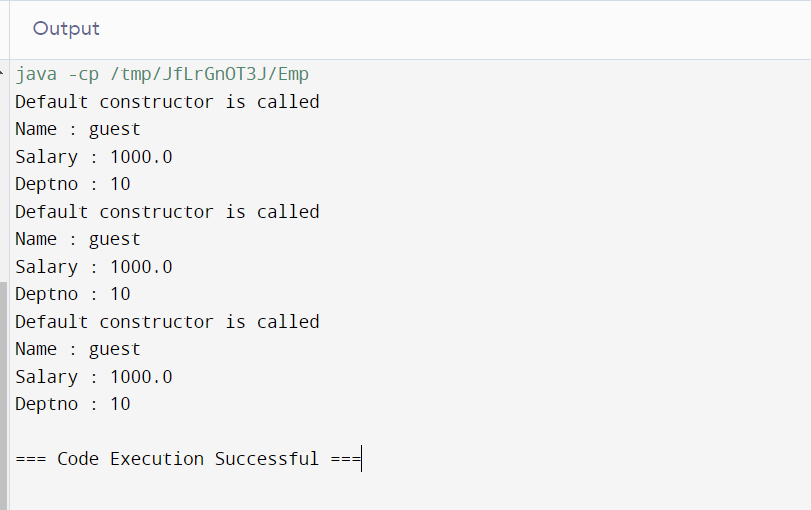
e.showData();

Emp e2=new Emp();

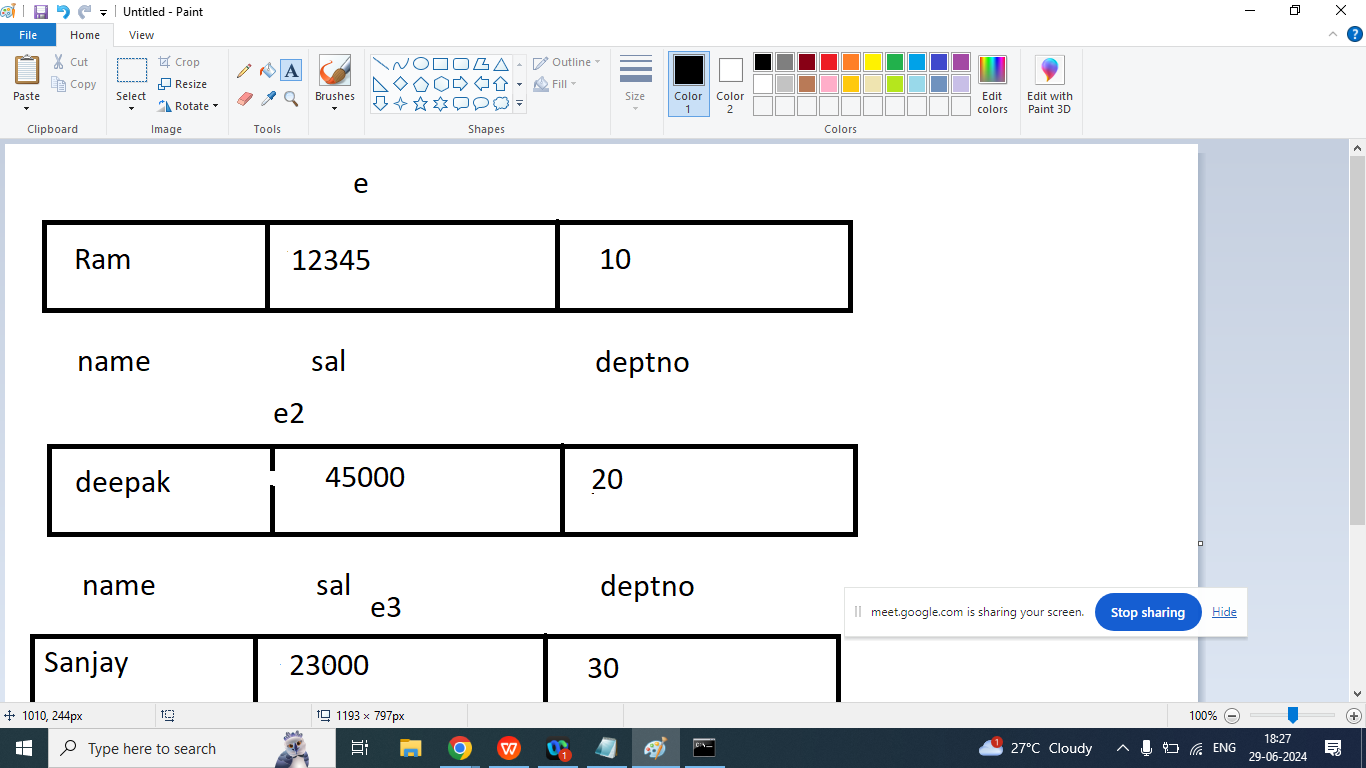
e2.showData();

}

}



Example Parameterized Constructor



class Emp{

String name;

float sal;

int deptno;

public Emp(){

System.out.println("Default constructor is called");

name="guest";

sal=1000;

deptno=10;

}

public Emp(String n,float s, int d){

System.out.println("Parameterized constructor is called");

name=n;

sal=s;

deptno=d;

}

public void showData(){

System.out.println("Name : "+name);

System.out.println("Salary : "+sal);

System.out.println("Deptno : "+deptno);

}

public static void main(String args[]){

Emp e=new Emp("Ram",12345,10);

e.showData();

Emp e1=new Emp("Deepak",45000,20);

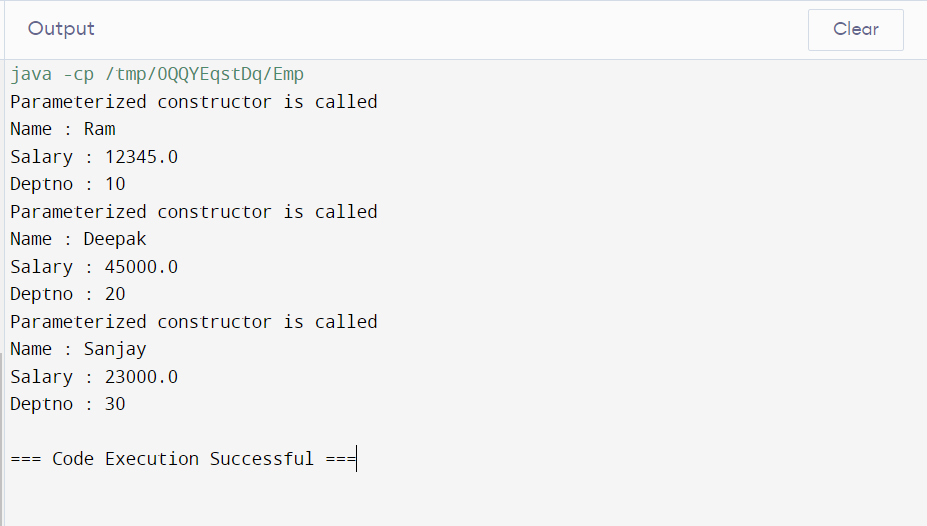
e1.showData();

Emp e2=new Emp("Sanjay",23000,30);

e2.showData();

}

}



class Emp{

String name;

float sal;

int deptno;

public Emp(){

System.out.println("Default constructor is called");

name="guest";

sal=1000;

deptno=10;

}

public void showData(){

System.out.println("Name : "+name);

System.out.println("Salary : "+sal);

System.out.println("Deptno : "+deptno);

}

public Emp(String n,float s, int d){

System.out.println("Parameterized constructor is called");

name=n;

sal=s;

deptno=d;

}

public static void main(String args[]){

Emp e=new Emp("Ram",12345,10);

e.showData();

Emp e1=new Emp("Deepak",45000,20);

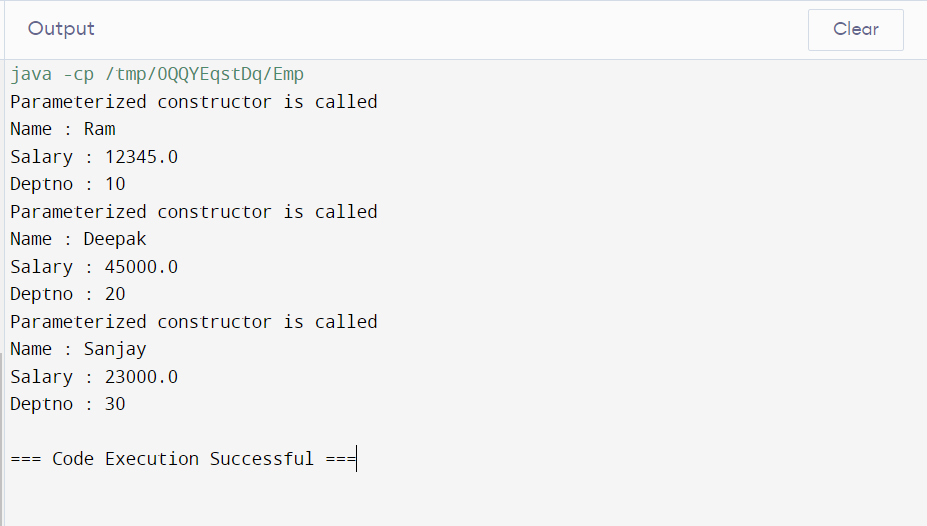
e1.showData();

Emp e2=new Emp("Sanjay",23000,30);

e2.showData();

}

}



Day 35

Q1. Explain Encapsulation in java programming?

Ans:

Hiding non essential information from the User.

The meaning of Encapsulation is to make sure that “sensitive” data is hidden from the users.

To achieve this

1. To declare member data as private
2. Create getter and setter method for access and change

Rules of Getter

1. Return type of getter method same as its member data
2. It does not take any argument but it must be return a value(same as its member data)
3. Method name should be prefix as a get and suffix its memberData Name

Syntax:

public return type getXXX(){

return XXX;

}

Example:

public int getRollNo(){

return rollno;

}

public float getPer(){

return per;

}

public String getName(){

return name;

}

Rules of setter

1. Return type of setter method always void
2. It does not any value but it must be take an argument(same as its member data)
3. Method name should be prefix as a set and suffix its memberData Name

Syntax:

public void setName(String name){

this.name=name; }

public void setRollNo(int rollno){

this.rollno=rollno;

}

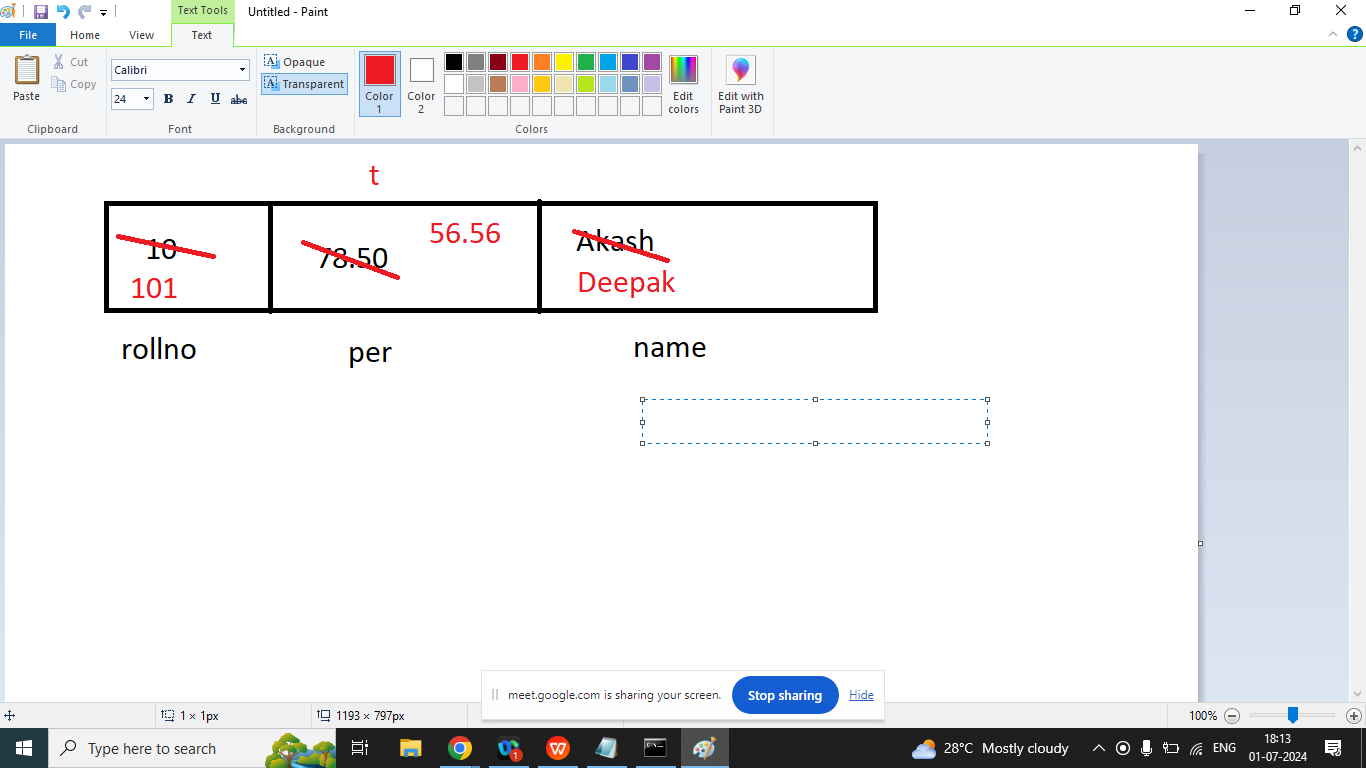
public void setPer(float per){

this.per=per;

}

Getter method

|  |
| --- |
| class A{  private int rollno=10;  private String name="Shani";  private float sal=101010.50f;  public int getRollNo(){  return rollno;  }  public String getName(){  return name;  }  public float getSalary(){  return sal;  }  public static void main(String args[]){  A obj=new A();  System.out.println("RollNumber = " + obj.getRollNo());  System.out.println("Name = " + obj.getName());  System.out.println("Salary = " + obj.getSalary());  }  } |
|  |
|  |



class Test{

private int rollno=10;//instance variable

private float per=78.50f;//instance variable

private String name="akash";//instance variable

public int getRollNo(){

return rollno;

}

public float getPer(){

return per;

}

public String getName(){

return name;

}

public void setName(String name){

this.name=name;

}

public void setRollNo(int rollno){

this.rollno=rollno;

}

public void setPer(float per){

this.per=per;

}

}

class T1{

public static void main(String args[]){

Test t=new Test();

t.setRollNo(101);

t.setPer(56.56f);

t.setName("deepak");

System.out.println(t.getRollNo());

System.out.println(t.getPer());

System.out.println(t.getName());

}

}

Q2. Explain this keyword in java programming?

Ans:

This keyword can be used in three places

1. this keyword refers to the current object in method or constructor. The most common use of this keywords is to eliminate the confusion between member data and arguments
2. This keyword can be used to call current class constructor.

class Test{

private int rollno=10;//instance variable

private float per=78.50f;//instance variable

private String name="akash";//instance variable

public Test(){

System.out.println("This is Default Constructor");

}

public Test(String name,int rollno,float per){

this();//to call current class default constructor

this.name=name;

this.rollno=rollno;

this.per=per;

System.out.println("This is Parameterized Constructor");

}

public int getRollNo(){

return rollno;

}

public float getPer(){

return per;

}

public String getName(){

return name;

}

public void setName(String name){

this.name=name;

}

public void setRollNo(int rollno){

this.rollno=rollno;

}

public void setPer(float per){

this.per=per;

}

}

class T1{

public static void main(String args[]){

Test t=new Test("Akash",103,67.45f);

System.out.println(t.getRollNo());

System.out.println(t.getPer());

System.out.println(t.getName());

}

}

|  |
| --- |
| This is Default Constructor  This is Parameterized Constructor  103  67.45  Akash |

1. To call current class method using this

class Test{

private int rollno=10;//instance variable

private float per=78.50f;//instance variable

private String name="akash";//instance variable

public Test(){

System.out.println("This is Default Constructor");

}

public Test(String name,int rollno,float per){

this();//to call current class default constructor

this.name=name;

this.rollno=rollno;

this.per=per;

System.out.println("This is Parameterized Constructor");

}

public int getRollNo(){

return rollno;

}

public float getPer(){

return per;

}

public String getName(){

return name;

}

public void setName(String name){

this.name=name;

}

public void setRollNo(int rollno){

this.rollno=rollno;

}

public void setPer(float per){

this.per=per;

}

public void showData(){

System.out.println("Name : "+name);

System.out.println("RollNumber : "+rollno);

System.out.println("Percentage : "+per);

}

public void display(){

this.showData();//to call current class method using this

}

}

class T1{

public static void main(String args[]){

Test t=new Test("Akash",103,67.45f);

t.display();

}

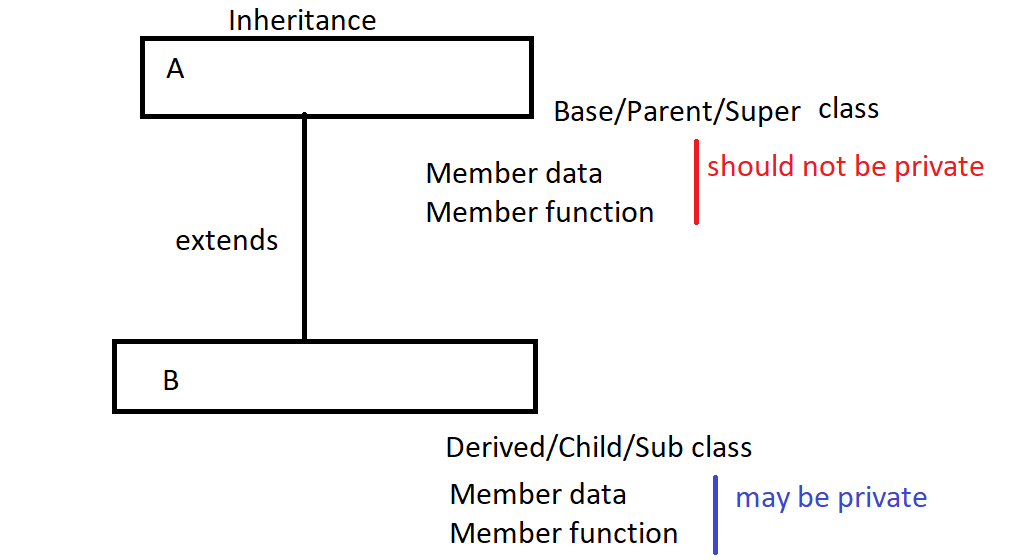
}

Day 36

Q1. Explain Inheritance in java programming?

Ans: Passing properties from one class into another class

1. Properties may be member data and member function
2. In case of inheritance member data should not be private
3. Java Does not support multiple inheritance through the class
4. Java Does not support Operator overloading
5. Java Does not support inline function
6. Java Does not support friend function
7. Java does not support virtual or pure virtual function



Syntax:

class Parent{

Member data

Member function

}

class Child extends Parent{

Member data

Member function

}

//Example of single inheritance extends keyword for single inheritance

class AB{

static{

System.out.println("This is parent class");

}

}

class B extends AB{

static{

System.out.println("This is child class ");

}

}

class Bc extends B{

static{

System.out.println("This is subchild class ");

}

}

class MainB extends Bc{

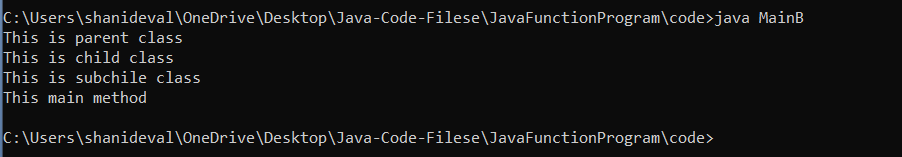
public static void main(String args[]){

AB obj=new AB();

System.out.println("This main method");

}

}



**Day 37**

Q1. Example of Inheritance

class A{

public static void main(String args[]){

System.out.println("This is A class Main");

}

}

class B extends A{

public static void main(String args[]){

System.out.println("This is B class Main");

}

}

Q2. Explain super keyword in java programming?

Ans: super keyword always represent parent class.There are three use of super keyword in java

1. To access Parent class member data from the child class

Syntax:

super.memberDataName

class A{

int y=10;

public A(){

System.out.println("This is A class Default Constructor");

}

public static void main(String args[]){

System.out.println("This is A class Main");

}

}

class B extends A{

int x=20;

public B(){

System.out.println("This is B class Default Constructor");

}

}

class C extends B{

int x=30;

public C(){

System.out.println("This is C class Default Constructor");

}

public void show(){

System.out.println(" This is A Parent X: "+super.y);

System.out.println(" This is B Child x: "+super.x);

System.out.println(" This is C Child x: "+x);

}

public static void main(String args[]){

C obj=new C();

obj.show();

}

}

|  |
| --- |
| This is A class Default Constructor  This is B class Default Constructor  This is C class Default Constructor  This is A Parent X: 10  This is B Child x: 20  This is C Child x: 30 |

1. Using super we call Parent class constructor from the child class

Note: super is a first statement of any constructor or methods

super();//call default constructor of Parent class

super(a,b)

class A{

int x=10;

public A(){

System.out.println("This is A class Default Constructor");

}

public A(int x){

this.x=x;

System.out.println("This is A class Parameterized Constructor");

}

}

class B extends A{

int y=20;

public B(){

System.out.println("This is B class Default Constructor");

}

public B(int x,int y){

super(x);

this.y=y;

System.out.println("This is B class Parameterized Constructor");

}

}

class C extends B{

int z=30;

public C(){

System.out.println("This is C class Default Constructor");

}

public C(int x,int y,int z){

super(x,y);

this.z=z;

System.out.println("This is C class Parameterized Constructor");

}

public void show(){

System.out.println(" This is A Parent X: "+super.x);

System.out.println(" This is B Child x: "+super.y);

System.out.println(" This is C Child x: "+z);

}

public static void main(String args[]){

C obj=new C(1,2,3);

obj.show();

}

}

|  |
| --- |
| This is A class Parameterized Constructor  This is B class Parameterized Constructor  This is C class Parameterized Constructor  This is A Parent X: 1  This is B Child x: 2  This is C Child x: 3 |

1. Super is also used to call Parent class method from the child class

Syntax:

super.methodName();

class A{

int x=10;

public A(){

System.out.println("This is A class Default Constructor");

}

public A(int x){

this.x=x;

System.out.println("This is A class Parameterized Constructor");

}

}

class B extends A{

int y=20;

public B(){

System.out.println("This is B class Default Constructor");

}

public B(int x,int y){

super(x);

this.y=y;

System.out.println("This is B class Parameterized Constructor");

}

public void show(){

System.out.println(" This is A Parent X: "+super.x);

System.out.println(" This is B Child x: "+y);

}

}

class C extends B{

int z=30;

public C(){

System.out.println("This is C class Default Constructor");

}

public C(int x,int y,int z){

super(x,y);

this.z=z;

System.out.println("This is C class Parameterized Constructor");

}

public void show(){

super.show();//to call Parent class method

System.out.println(" This is C Child "+z);

}

public static void main(String args[]){

C obj=new C(1,2,3);

obj.show();

}

}

|  |
| --- |
| This is A class Parameterized Constructor  This is B class Parameterized Constructor  This is C class Parameterized Constructor  This is A Parent X: 1  This is B Child x: 2  This is C Child x: 3 |

Day 38

Q1. Explain Polymorphism in java programming?

Ans: Poly morphism means one name many forms

Examples of Polymorphism

MySelf for You as a Trainer

Myself for Student in college as a Professor

Myself for Home I am son

Myself for My child I am Father

Myself for Wife I am Husband

Myself for Parents I am son

Types of Polymorphism

1. Compile time Polymorphism

Method Overloading

1. Run Time Polymorphism

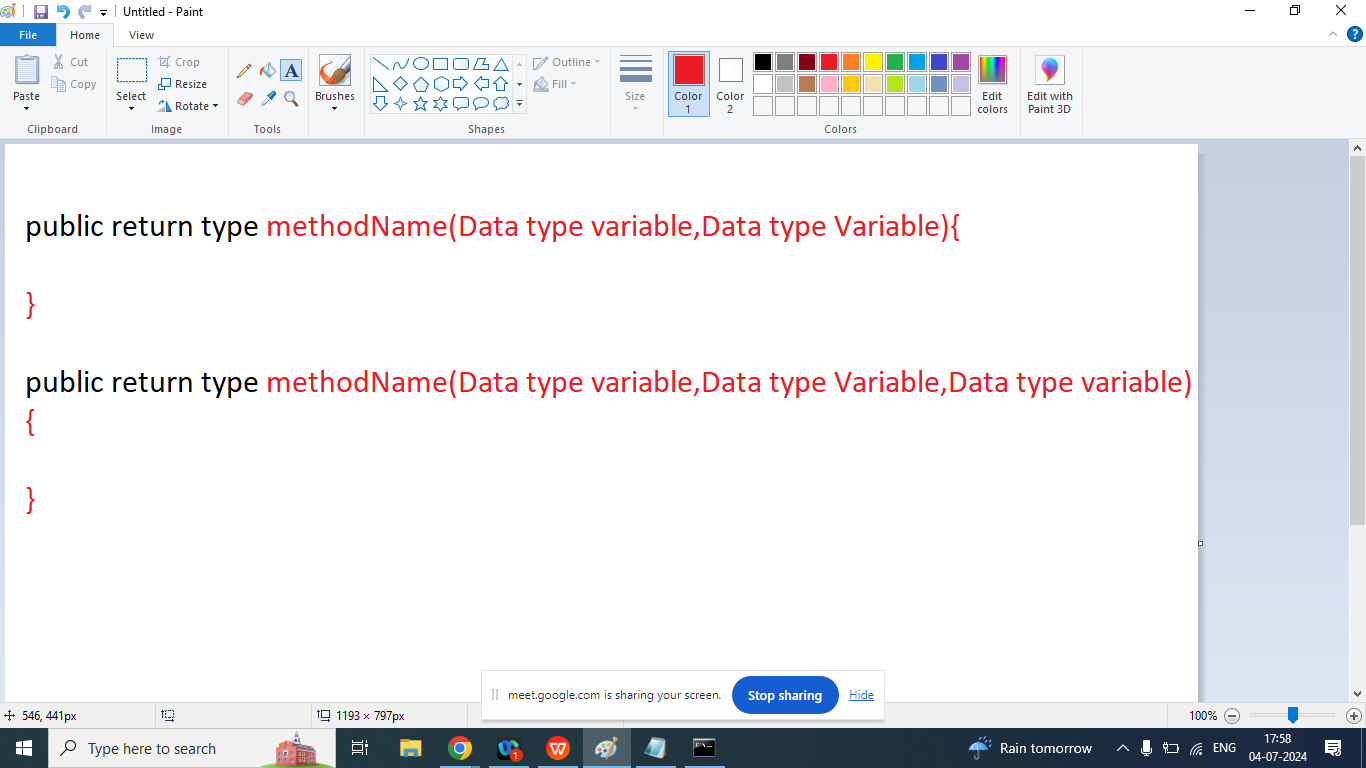
Method Overriding

Q2. Explain Method overloading in Java programming?

Ans:

Method Overloading means having two or more methods in a class with the same name and different types of Parameters or different number of Parameters.

Syntax:



class Op1{

public void add(){

int a,b,c;

a=10;

b=20;

c=a+b;

System.out.println("Addition without argument"+c);

}

public int add(int a,int b){

int c;

a=10;

b=20;

c=a+b;

System.out.println("Addition with two int argument"+c);

return c;

}

public void add(float a,float b){

float c=a+b;

System.out.println("Add two float : "+c);

}

public void add(double a,double b){

double c=a+b;

System.out.println("Add two double : "+c);

}

public void add(String a,String b){

int c=Integer.parseInt(a)+Integer.parseInt(b);

System.out.println("Add two String : "+c);

}

public static void main(String args[]){

Op1 obj=new Op1();

obj.add();

obj.add(5,7);

obj.add(12.5,2.5);

obj.add(2.5f,2.5f);

obj.add("1","1");

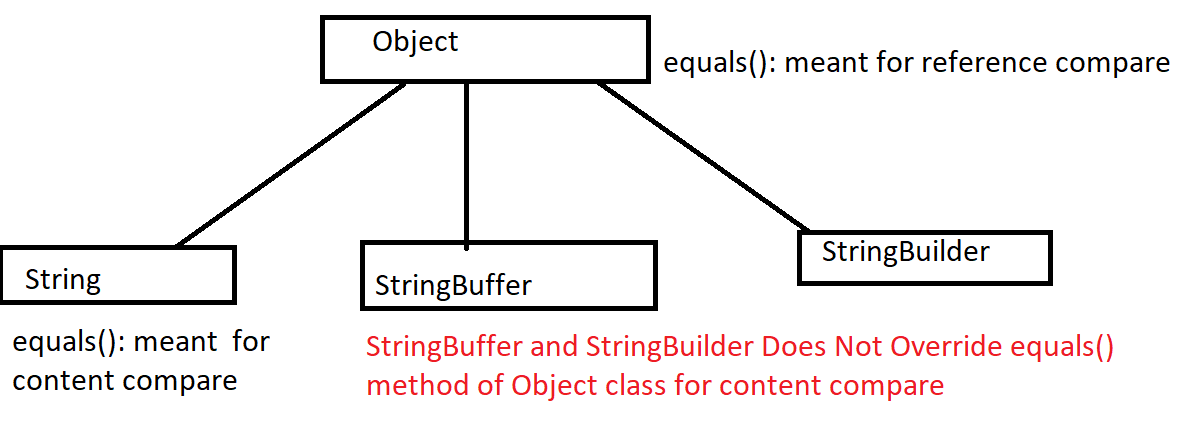
}

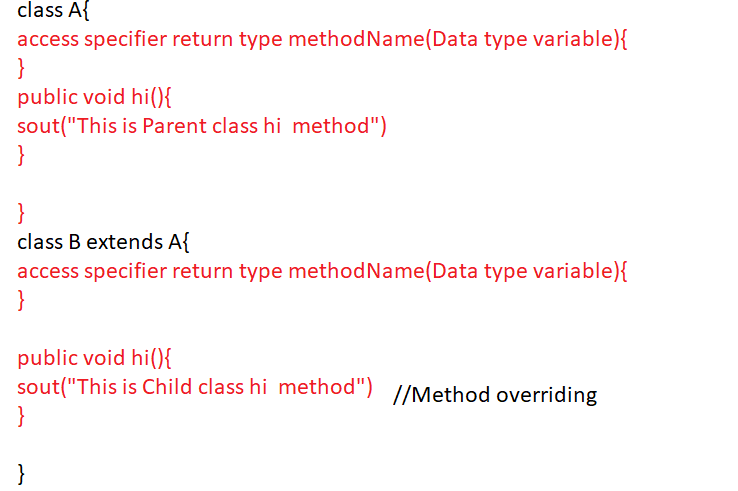
}

|  |
| --- |
| Addition without argument: 30  Addition with two int arguments: 12  Add two double : 15.0  Add two float : 5.0  Add two String : 2 |

Q2. Explain Method overriding in java programming?

Ans: In java method overriding is a features that allows a sub class or child class to provide a specific implementation of a method that is already defined in Parent class. When a method in a sub class has the same name, same return type, same argument (type)





class A{

public void hi(){

System.out.println("This is Parent class hi... method()");

}

}

class B extends A{

public void hi(){

System.out.println("This is Child class hi... method()");

}

public static void main(String args[]){

A obj=new B();

obj.hi();

}

}

|  |
| --- |
| This is Child class hi... method() |

|  |
| --- |
| class AB{  public void hi(){  System.out.println("AB class ");  }  }  class BC extends AB{  public void hi(){  System.out.println("Bc Main method");  }  public static void main(String args[]){  AB obj=new BC();  obj.hi();  }  } |
| Bc Main method |

Day 39

Q1. Explain Abstraction in java programming?

Ans: Give Essential Details to the user in summarized Form

Abstraction in java refers to hiding implementation details of a code and exposed only the necessary information to the user.

There are two ways to achieve abstraction in java

1. Using abstract class (abstraction we can not find 100%)
2. Using Interface (100% abstraction we can find)

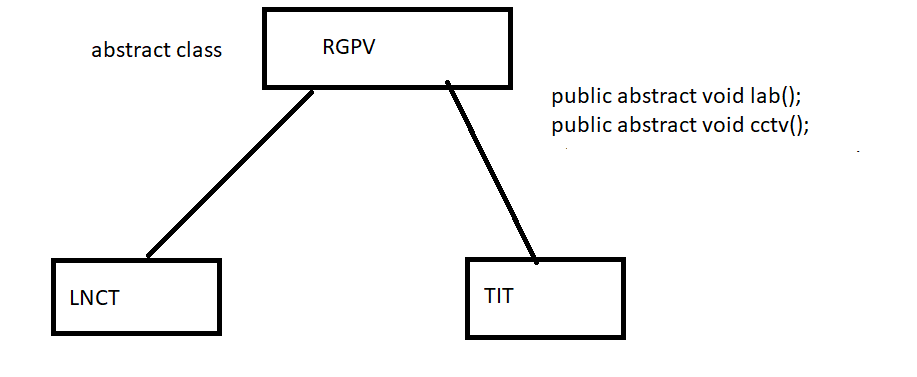
Abstract class:

1. Abstract class is the collection of abstract method or non abstract method
2. We cannot create an instance / object of Abstract class
3. Abstract class can have a constructor
4. Abstract class can inherit another abstract or non abstract class
5. An abstract class can implement any interfaces
6. The abstract keywords in an non-access modifier, used for classes and methods

**Abstract method**: can only be used in an abstract class, and it does not have a body /definition. The definition / body /implementation provided by its child classes.

**Abstract class:** Ia a restricted class that cannot be used to create objects. It must be inherited from another class

We cannot create an object of abstract class but it can store reference of its child class.



Apart from abstract class method a child class can have its own method

abstract class RGPV{

public abstract void lab();

public abstract void cctv();

//abstract

public void examCopyCheck(){

System.out.println("This Is RGPV non abstract method");

}

public RGPV(){

System.out.println("This is RGPV CONStrUCTOR");

}

}

class College extends RGPV{

public void cctv(){

System.out.println("This is CCTV Method Rule by RGPV");

}

public void lab(){

System.out.println("This is lab Method Rule by RGPV");

}

public void staff(){

System.out.println("This is College class Staff Method");

}

public College(){

System.out.println("This is College CONStrUCTOR");

}

public static void main(String args[]){

//RGPV obj=new RGPV();//C.E.

RGPV obj=new College();

obj.cctv();

obj.lab();

obj.examCopyCheck();

//obj.staff();

College c=new College();

c.cctv();

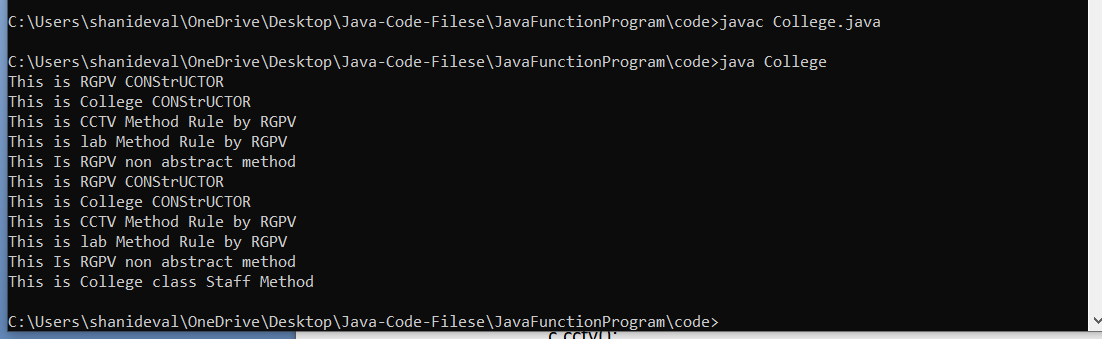
c.lab();

c.examCopyCheck();

c.staff();

}

}



Day 40

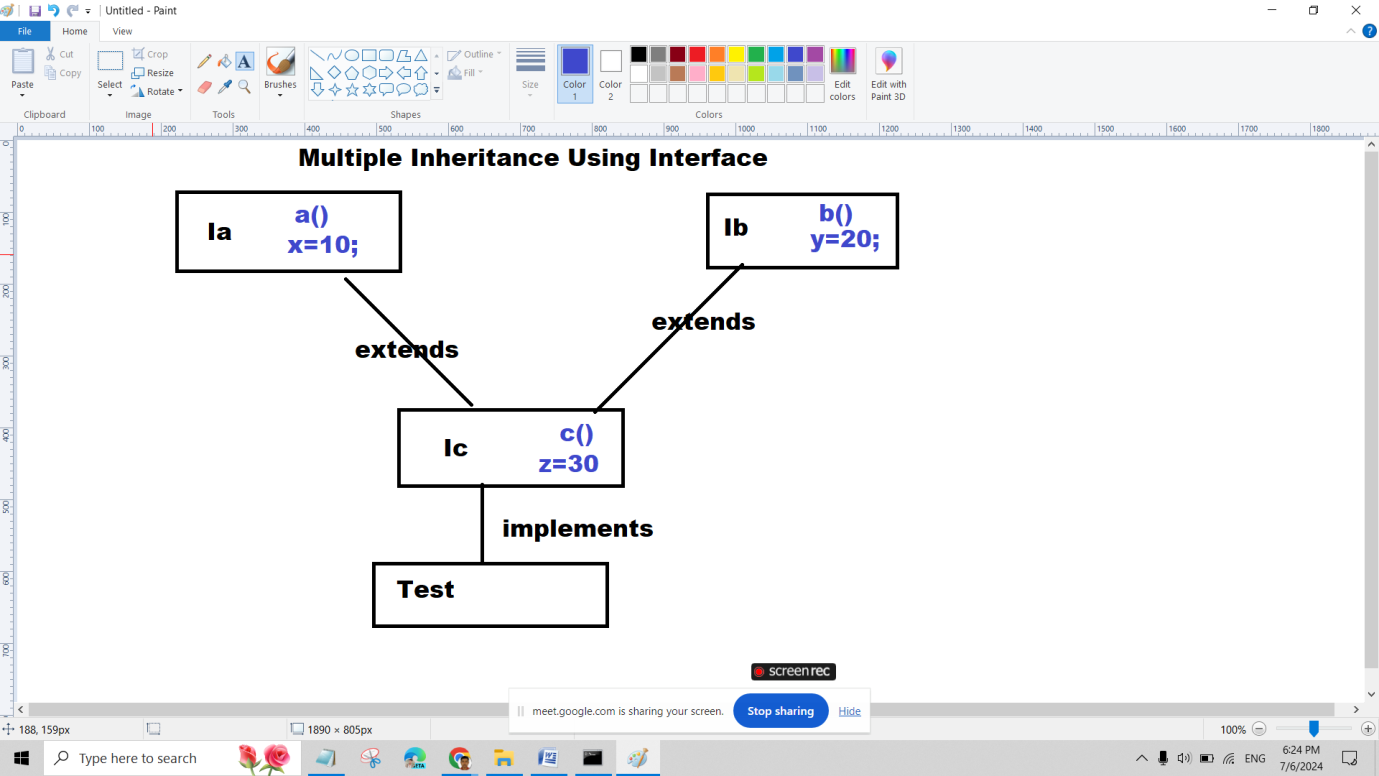
Q1.Explain interface in java programming?

Ans: we can also achieve abstraction in java programming with interface

1. Using interface we achieve 100% abstraction in java because by default every method in interface is public and abstract
2. No need to use abstract keyword in interface
3. Interface is a collection of method declaration and final variable
4. Interface keyword is used to declare interface
5. A class can implements any number of interfaces at a time
6. A interface can extends more than one interface at a time(Multiple inheritance)
7. We cannot create an instance / object of interface, but it can store the reference of its implemented class.

Apart from interface method a implemented class can have its own method

|  |
| --- |
| // Interface Ia with a constant integer x declared as 10 and an abstract method a()  interface Ia {  int x = 10; // Constant integer x  void a(); // Abstract method a()  }  // Interface Ib with a constant integer y declared as 20 and an abstract method b()  interface Ib {  int y = 20; // Constant integer y  void b(); // Abstract method b()  }  // Class Test implementing interfaces Ia and Ib  class Test implements Ia, Ib {  // Implementation of method a() from interface Ia  public void a() {  System.out.println("Ia Interface method : " + x);  }  // Implementation of method b() from interface Ib  public void b() {  System.out.println("Ib Interface method : " + y);  }  // Additional method defined in Test class  public void hello() {  System.out.println("This is Test class Method");  }  // Main method to demonstrate usage of interfaces and Test class  public static void main(String args[]) {  // Creating an object of type Ia, which references an instance of Test (polymorphism)  Ia obj = new Test();  obj.a(); // Calls method a() through interface reference  // Creating an object of type Test  Test t = new Test();  t.hello(); // Calls method hello() from Test class  t.a(); // Calls method a() from Test class  t.b(); // Calls method b() from Test class  }  } |



interface Ia{

int x=10;

void a();

}

interface Ib{

int y=20;

void b();

}

interface IC extends Ia,Ib{

int z=30;

void c();

}

class Test implements IC{

public void a(){

System.out.println("Ia INterface method : "+x);

}

public void b(){

System.out.println("Ib INterface method : "+y);

}

public void c(){

System.out.println("Ic INterface method : "+z);

}

public void hello(){

System.out.println("This is Test class Method");

}

public static void main(String args[]){

//Ia obj=new Ia();//C.E.

Ia obj=new Test();

obj.a();

//obj.hello();

Test t=new Test();

t.hello();

t.a();

t.b();

t.c();

//t.x=22;

}

}

Day 41

Q1. Explain Static keyword in java programming?

Ans:

1. Variables:

When a variable declared static, then a Single copy of the variable is created and shared among all objects of at the class level. Static variable are essentially , global variables.

All Instance of the class share the same static variables.

We can declare static variable at class level

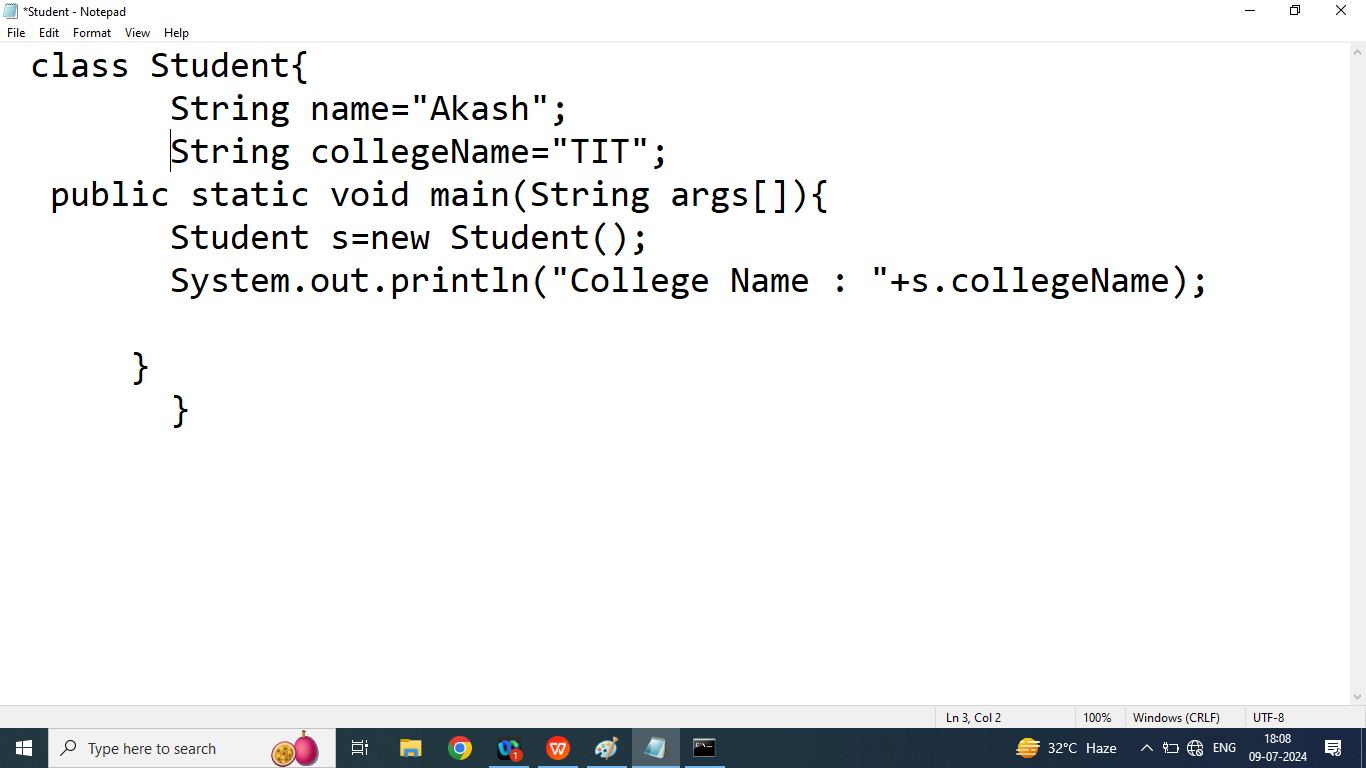
Static variable can access by class name without using of object

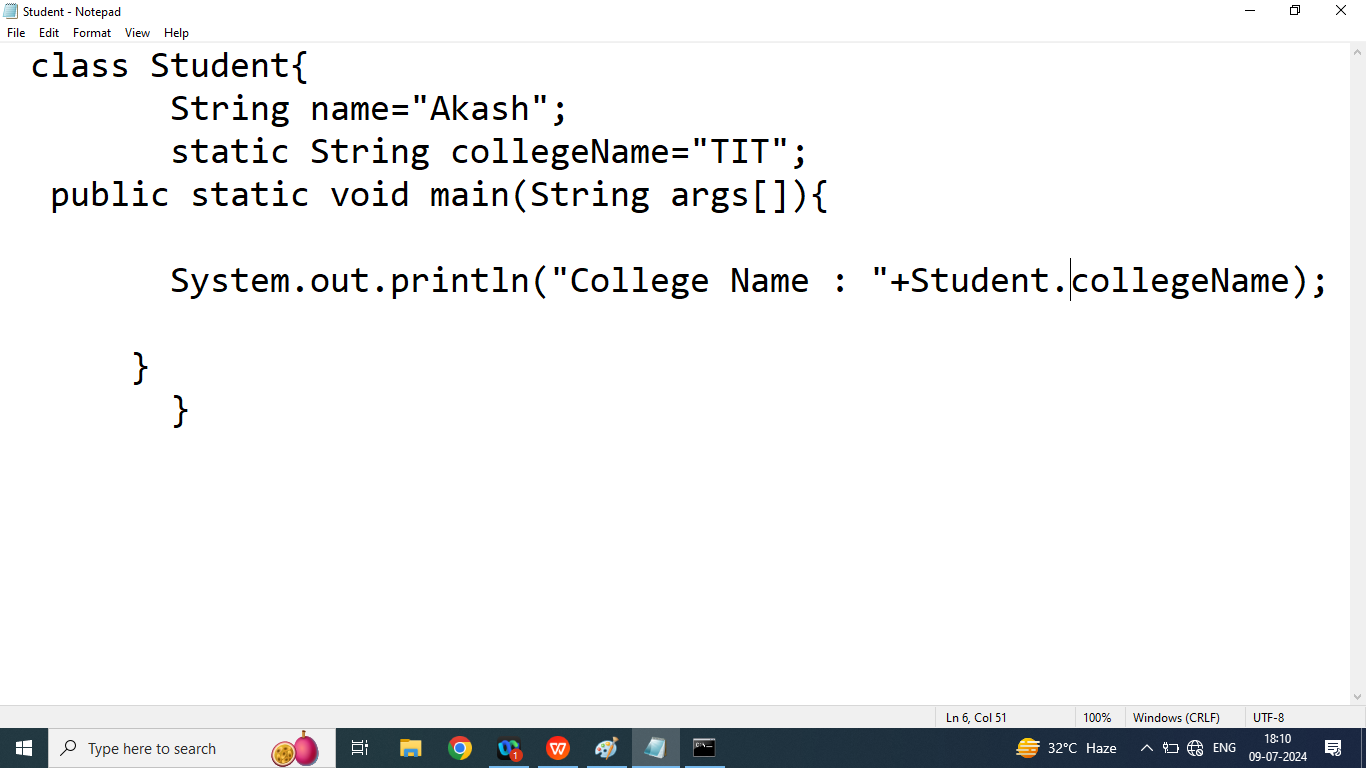
If static member in a same class we used without using class Name or Object name.

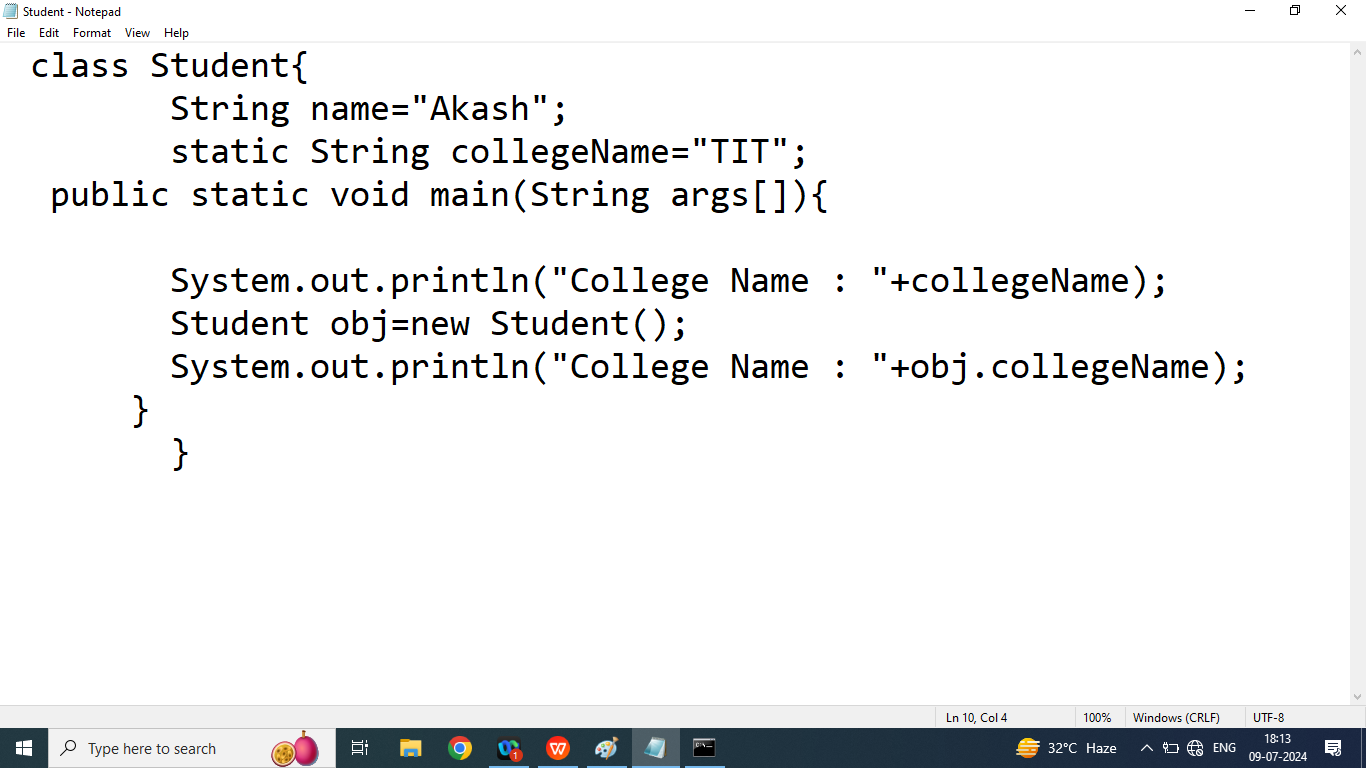
Static variable recommended to access via className

Syntax:

ClassName.VariableName







1. Static Block:

Static block is used to initialized a static variable or class level variable

A class can have multiple static block but it can be executed according order they are declared

In java static block are executed before main method

Syntax:

static{

//static block

}

Static method : A method that can be worked only static member

A static method can access using class name(no need to create an object of a class)

Q1. Write a java program to count number of object created

class Student{

String name="Akash";

static String collegeName;

private static int count=0; //static variable

public static int getCount(){

return count;

}

public Student(){

count++;

}

public Student(String name){

this.name=name;

count++;

}

// static block

static{

System.out.println("This is Static Block 2");

}

static{

System.out.println("This is Static Block 1");

collegeName="LNCT";

}

public static void main(String args[]){

System.out.println("This is Main Method");

Student s1=new Student();

Student s2=new Student("Akash");

Student s3=new Student();

System.out.println("Number of Object created : "+Student.getCount());

System.out.println("Number of Object created : "+getCount());

System.out.println("Number of Object created : "+s1.getCount());

System.out.println("College Name : "+collegeName);

}

}

|  |
| --- |
| This is Static Block 2  This is Static Block 1 |
|  |

Day 42

Q1. Explain final keyword in java programming?

Ans:

If we want to define some constant then we should go for final variable(we cannot change value of final variable)

If we want to define some methods and we want to not overridden then we should go for final methods

if we want to create our own immutable class then we should go for final class

Final variable: We cannot change value of final variable

class TestFinal{

final int X;

public TestFinal(){

X=20;

}

public void show(){

System.out.println("X : "+X);

}

public static void main(String args[]){

TestFinal t=new TestFinal();

t.X=111;

t.show();

}

}

Final methods:We cannot override final methods

Final class: we cannot inherit final class

Note: we also declare local variable as a final.final variable declared in UPPERCASE

Example:

final class TestFinal{//final class

final int X;//final variable

public TestFinal(){

X=20;

}

public final void show(){

System.out.println("This is Parent class Show : "+X);

}//final methods

}

class Hello extends TestFinal{

public static void main(String args[]){

TestFinal t=new TestFinal();

t.show();

}

}

class TestFinal{//final class

final int X;//final variable

public TestFinal(){

X=20;

}

public final void show(){

System.out.println("This is Parent class Show : "+X);

}//final methods

}

class Hello extends TestFinal{

public static void main(String args[]){

final int b=10;

TestFinal t=new TestFinal();

System.out.println("Final Local Variable : "+b);

t.show();

}

}

class TestFinal{//final class

final int X;//final variable

public TestFinal(){

X=20;

}

}

class Hello extends TestFinal{

public void show(){

System.out.println("This is Parent class Show : "+X);

}

public static void main(String args[]){

final int b=10;

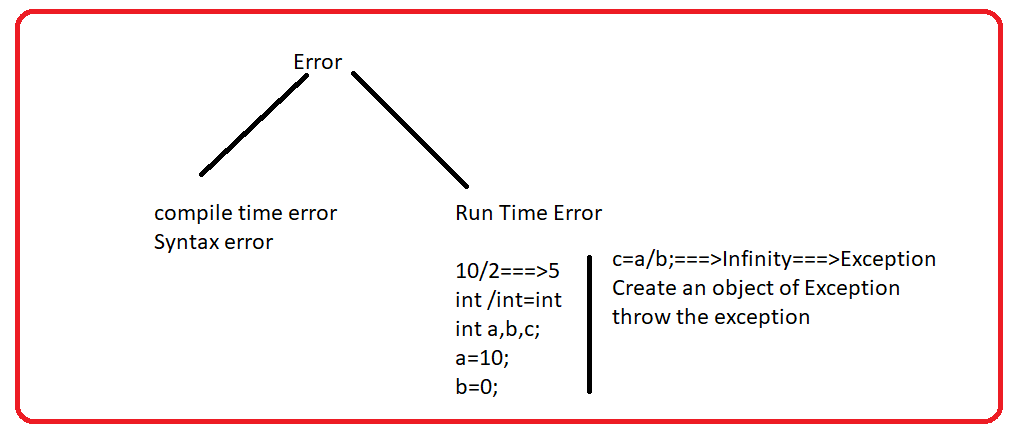
Hello t=new Hello();

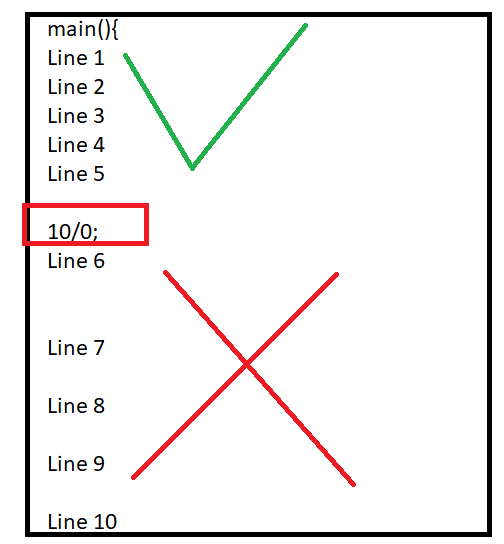
System.out.println("Final Local Variable : "+b);

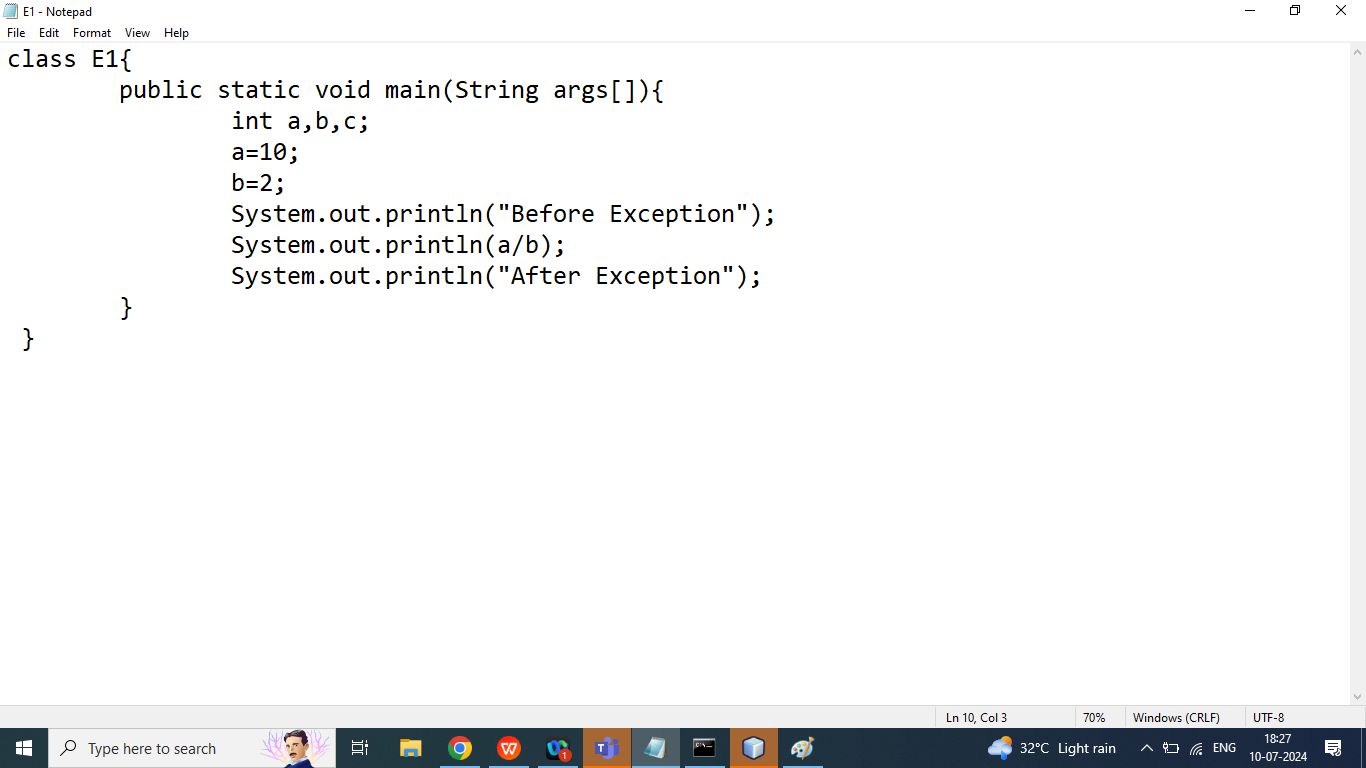
t.show();

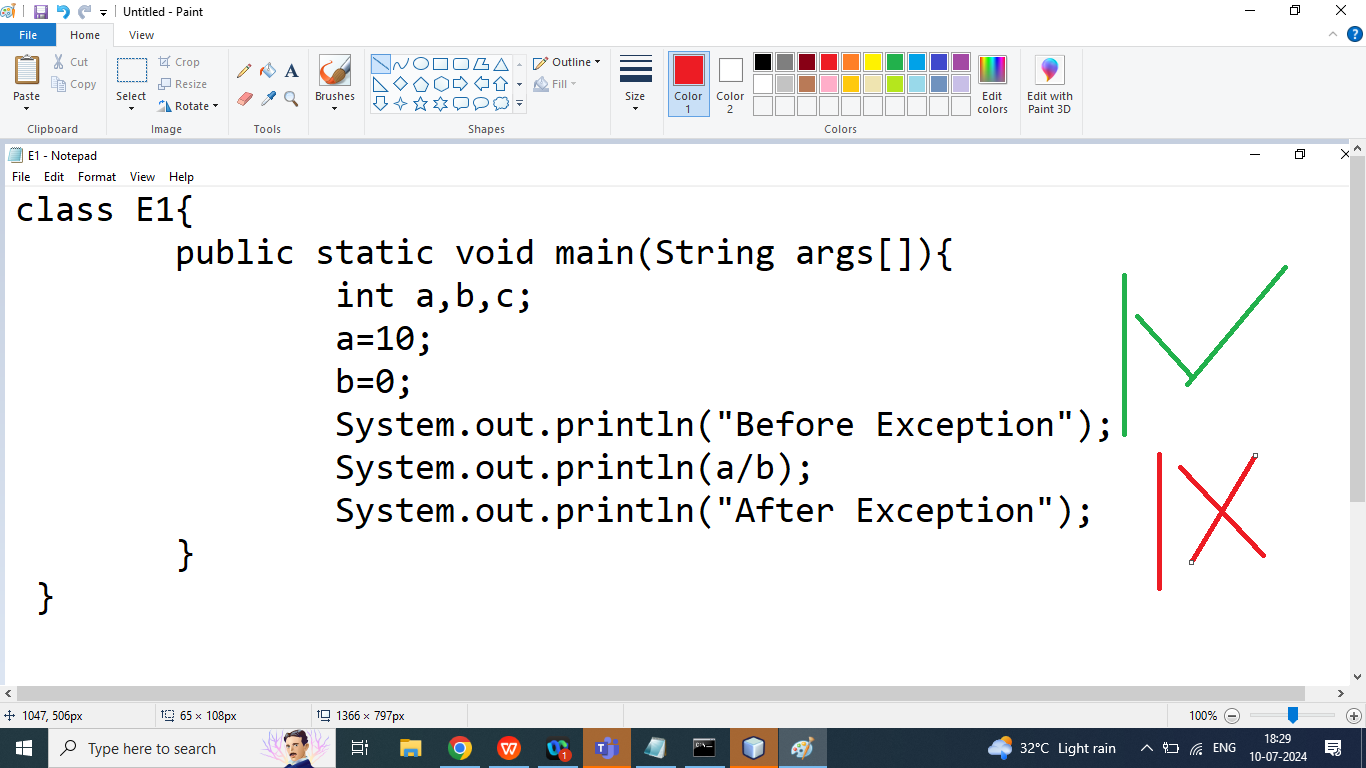
}

}









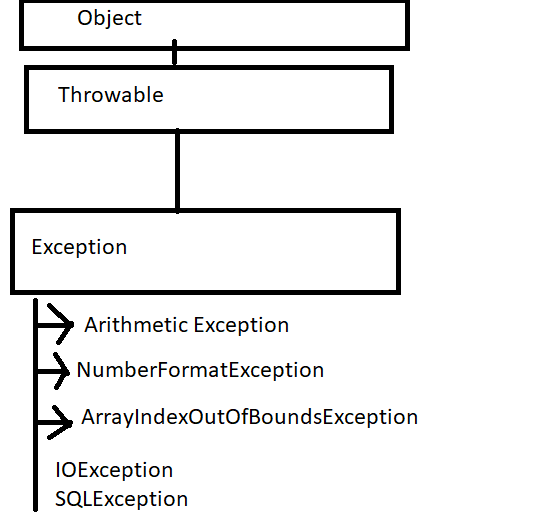
Day 43

Q1. Explain Exception Handling in java programming?

Ans:

When executing java code, different errors can occur, coding errors made by the programmer, errors due to wrong input

When an errors occurs, java will normally stop the program and generate an error message. The technical terms for this, in java will throw an exception.



In java programming we use 5 keyword to handle the exception

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

The try statement allows you to define a block of code to be tested for errors while it is being executed .

1. catch:The catch block statement allows you to define a block of code to be executed if an exception occurs in the try block

Catch block is always take an argument Exception class or Its Derived class

A single try bock followed by multiple catch block.

A catch block is conditional , if Exception occurs then catch block will executed.

Inside catch block we write handling code.

Syntax:

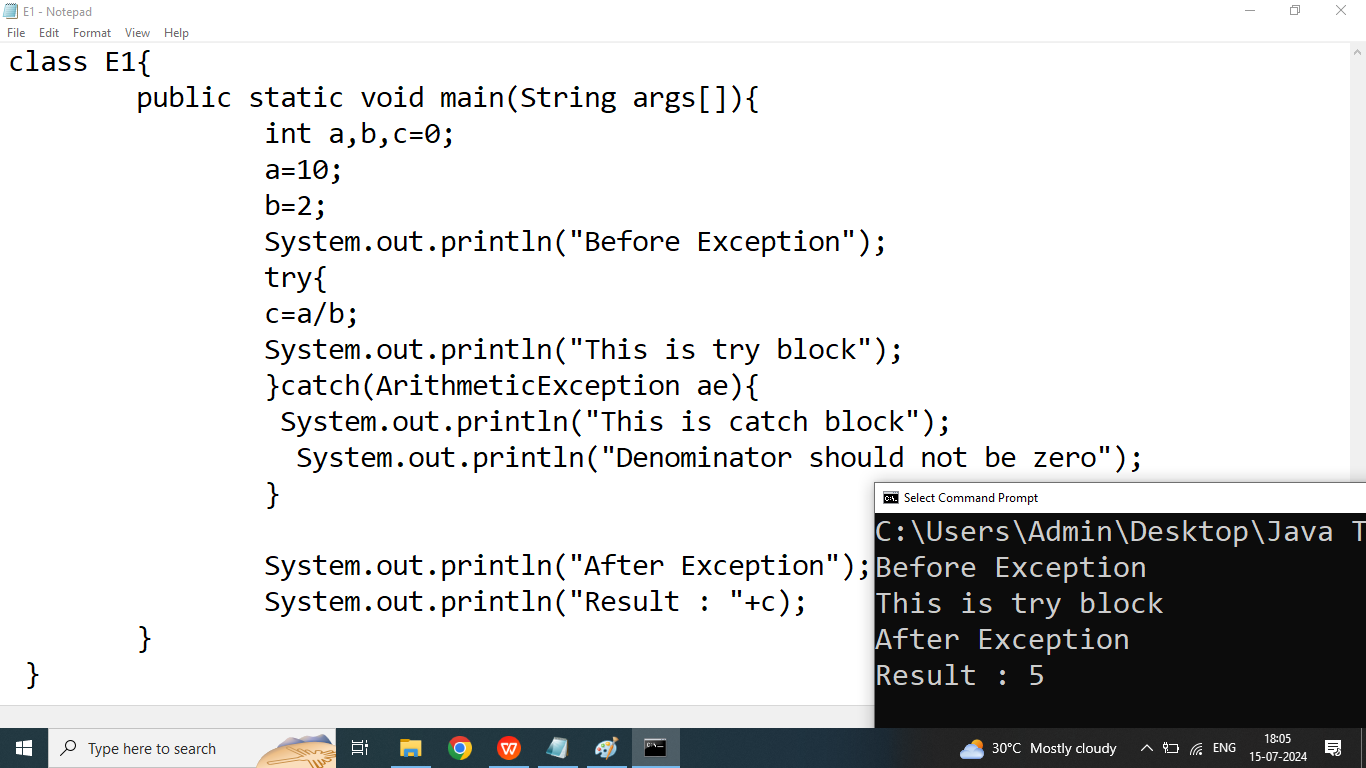
try{

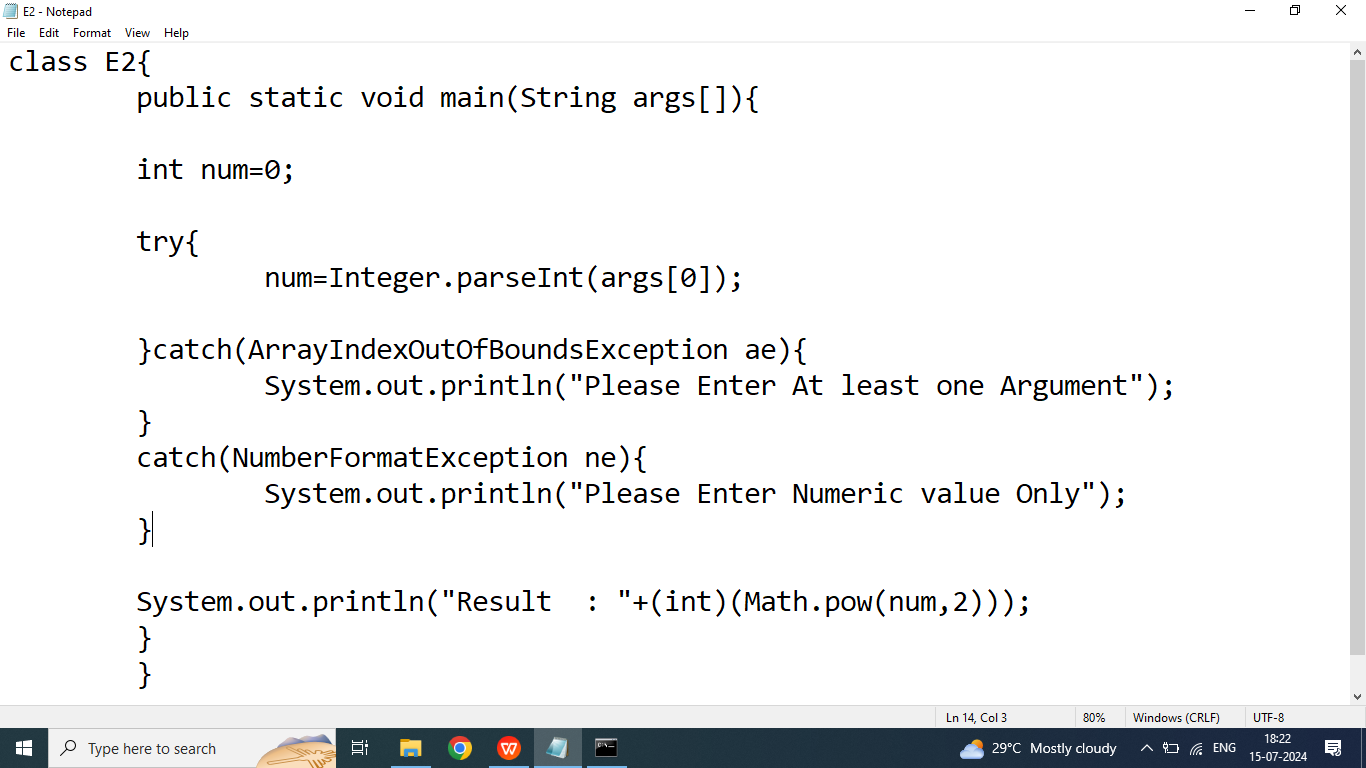
//code that result may be exception

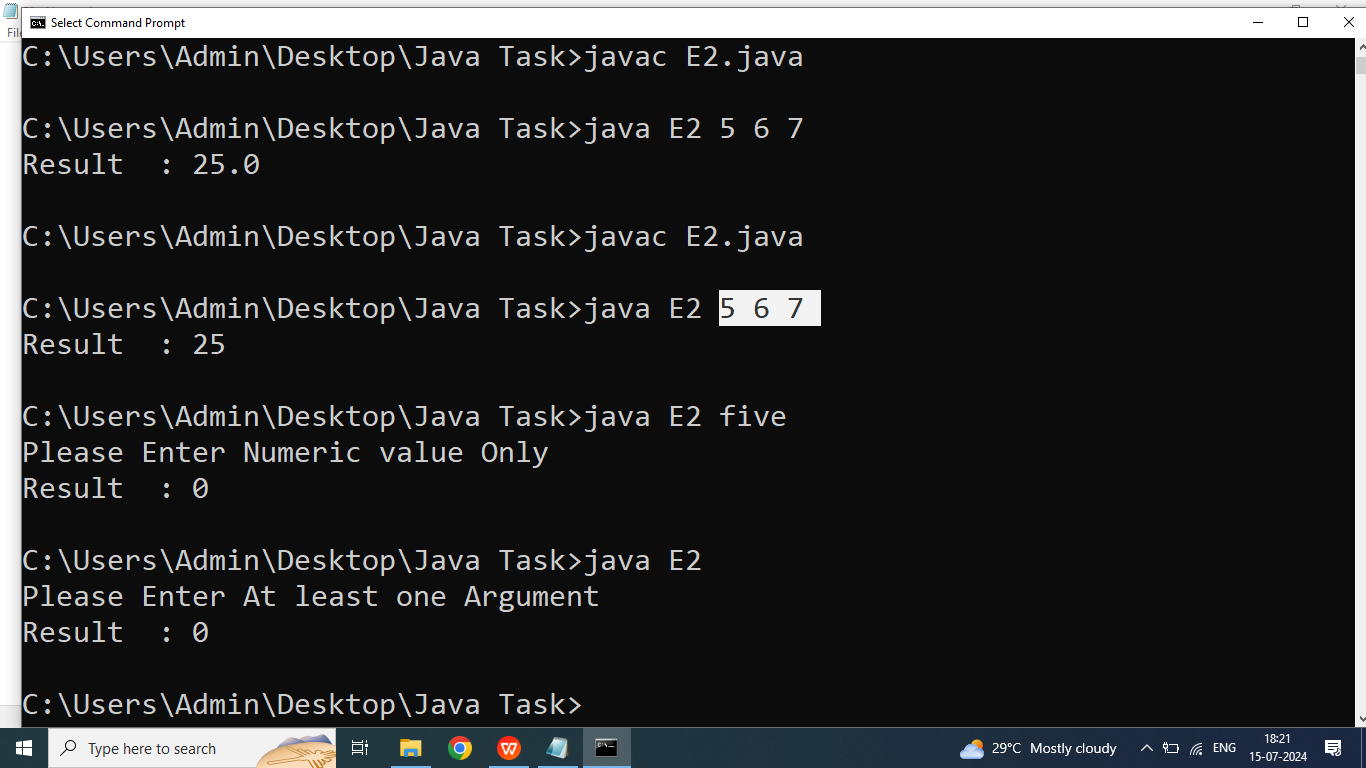
}catch(Exception){

//handling code

}







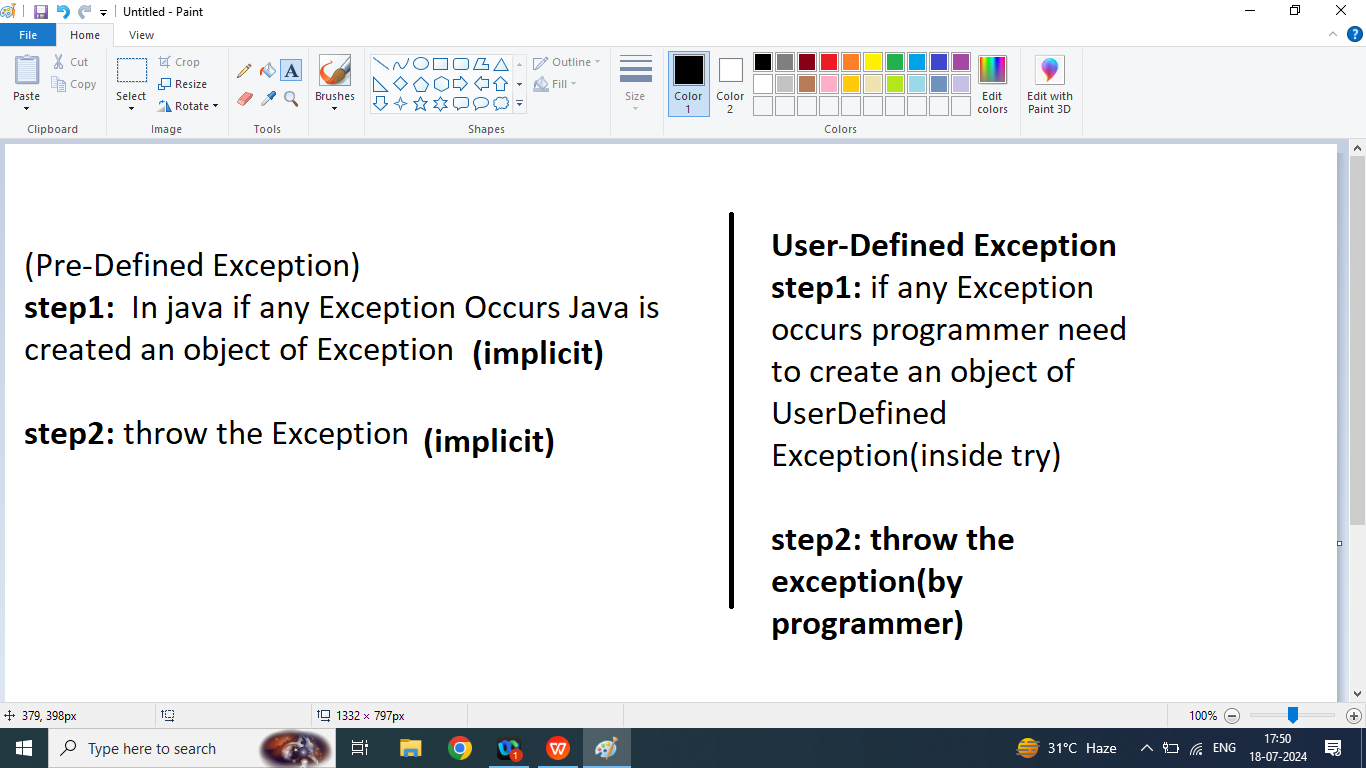
Day 44

Q1. How to create a user defined Exception in java programming?

Ans: The process of creating a user defined exception

Involves creating a new class that extends Throwable / Exception

Syntax:



Q1. Write a java program to read age of emloyee from the command line argument and check given age is valid or not using user defined Exception Handling

class AgeException extends Throwable{

String msg;

public AgeException(String msg){

this.msg=msg;

}

public String getMsg(){

return msg;

}

}

class E3{

public static void main(String args[]){

int a=-1;

try{

a=Integer.parseInt(args[0]);

if(a<0 || a>100){

AgeException ae=new AgeException("Invalid Age");

throw ae;

}

System.out.println("Valid Age");

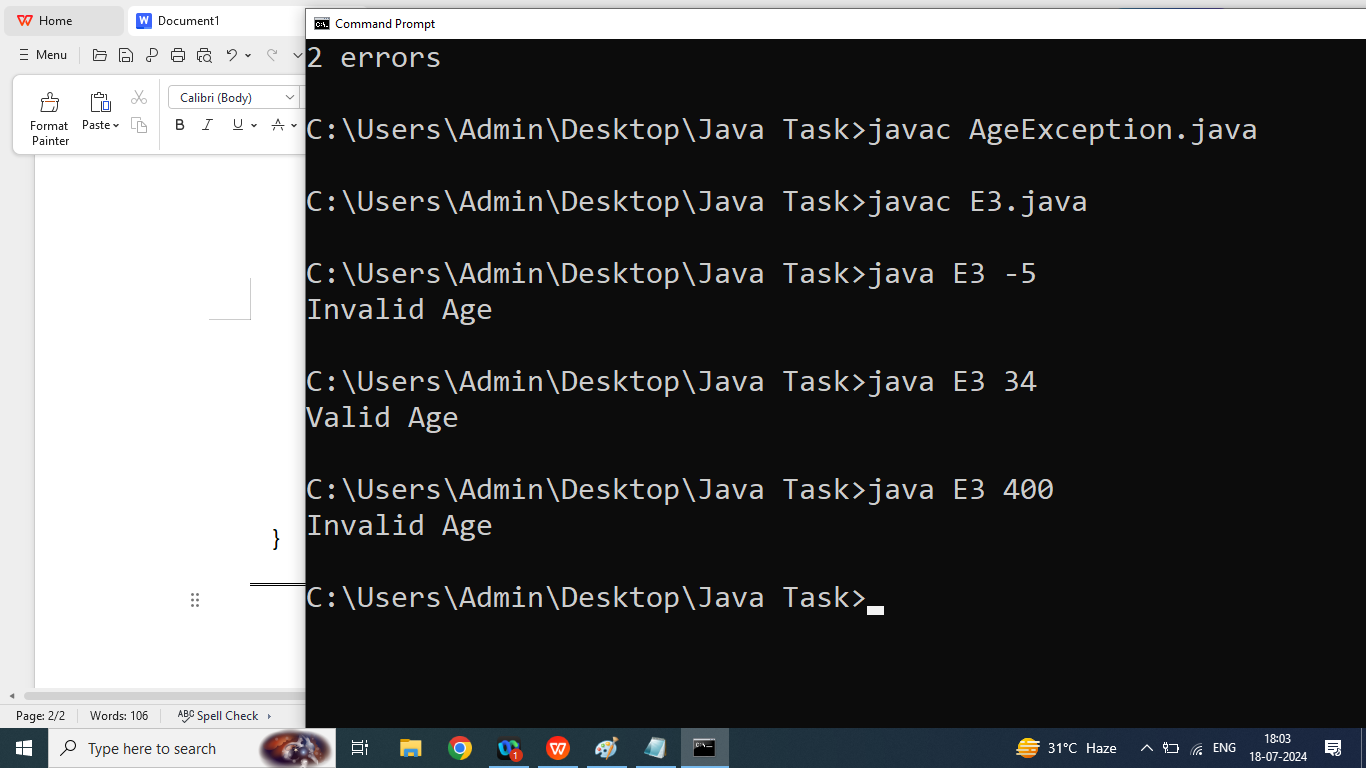
}catch(AgeException x){

System.out.println(x.getMsg());

}

}

}



Q2. Types of Exception in java?

1. Pre-Defined
2. User-Defined
3. Checked Exception(Example: SQLException, FileNotFoundException)
4. Un Checked Exception(Example ArithmaticException,NumberFormatException,ArrayIndexOutOfBoundsException)

Q1. Explain Linear Data Structure in java programming?

DBMS: DBMS Deals with Data Storage

DS: Data Structure Deals With How to store and retrieve efficient manner

Linear Data Structure: if we want store data in a particular sequence then we should go for Linear data structure

We discuss following Linear Data Structure

1. Array
2. Stack(FILO/LIFO)
3. QUEUE(FIFO)
4. Linked List

