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;

}

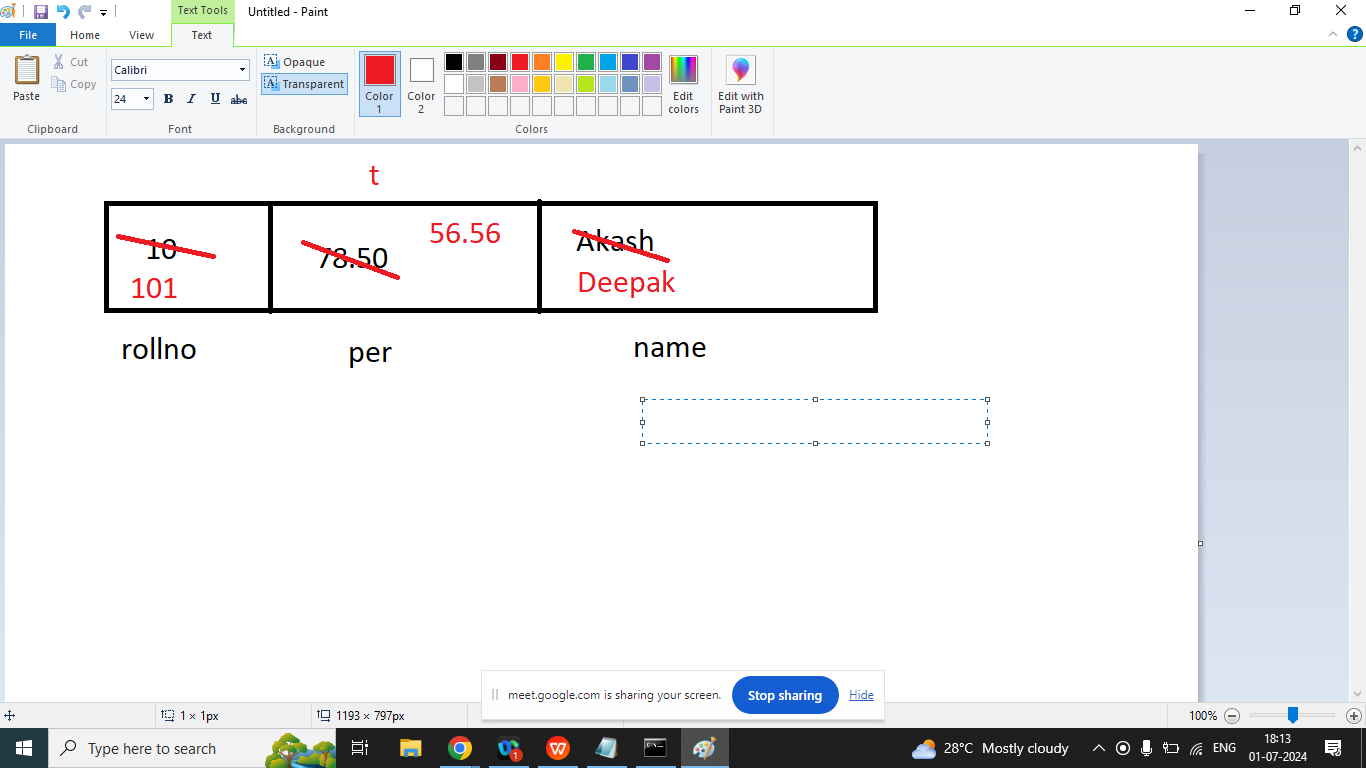
public void setPer(float per){

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();

}

}