Irfan Ali Muzammail

OOP PAST PAPER LONG QUESTIONS

Subjective Part (2013)

- **Define a class "clock" with three data members hour, minute and second and two methods "increment Seconds ()" that increments the seconds by one and "display Time()" that displays the time in this manner "Time" is 12 hours :30 minutes :10 seconds?**

```
package javaapplication146;

public class JavaApplication146 {

   public static void main(String[] args) {

      Clock c=new Clock();

      c.incrementSec();

      c.display();

   }

  }

class Clock{

int hour=12;

int min=30;

int sec=9;

void incrementSec(){

sec++;

}

void display(){

System.out.print("The time is:"+hour+"hours:"+min+"minute:"+sec+"second");

}
```


Scanned with CamScanner

```
}
```

- **Write a program to illustrate how to handle division by ZERO Exception?**

```
package javaapplication115;

import java.util.Scanner;

public class JavaApplication115 {

 public static void main(String[] args) {

double a,b,c;

 Scanner input = new Scanner(System.in);

System.out.println("Enter two numbers:");

  a=input.nextDouble();

  b=input.nextDouble();

try{c=a/b;

System.out.println(c);

}

 catch(ArithmeticException e){

    System.out.println(e);

}

  }

  }

}
```

Subjective Part (2015)

- **Create a class *Student* with instance variable roll no., name, age, and decide proper data types and access modifiers for these variables. Define overloaded constructors, getter/setter methods and also override to String (). Demonstrate this class in your program?**

Scanned with CamScanner

```
package javaapplication143;
import java .util.Scanner;
public class JavaApplication143 {
public static void main(String[] args) {
   Student s =new Student(5,"irfan",12);
   s.getter();
   s.setter(12, "ali", 20);
   }
}
class Student{
public int rollno;
public String name;
public int age ;
Student(int r, String n,int a){
rollno=r;
System.out.println(r);
name=n;
System.out.println(n);
age=a;
System.out.println(a);
}
void getter(){
Scanner input =new Scanner(System.in);
rollno=input.nextInt();
System.out.println(rollno);
name=input.next();
System.out.println(name);
age=input.nextInt();
System.out.println(age);
}
void setter(int r, String n,int a){
rollno=r;
System.out.println(r);
name=n;
System.out.println(n);
age=a;
System.out.println(a);
}

public String tostring()
```


Scanned with CamScanner

```
{return rollno+""+name+""+age; }
}
```

- **Explain Exception Handling in java and why we use Nested Try Blocks. Write a program to Explain Exception Handling by Nested try/catch Blocks.**

Solution:

Why we use Exceptional Handling in Java:

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

Why we use Nested Try Block:

Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error. In such cases, exception handlers have to be nested.

EXAMPLE:

```
//Creating Nested class package nested;
import java.util.InputMismatchException;
public class Nested {
 public static void main(String[] args) {
 try {
System.out.println("Outer try block starts");
try {
System.out.println("Inner try block starts");
 int res = 5 / 0; }
catch (InputMismatchException e) {
System.out.println("InputMismatchException caught"); }
 finally {
 System.out.println("Inner final");
} }
 catch (ArithmeticException e) {
 System.out.println("ArithmeticException caught");
 }
 finally { System.out.println("Outer finally");
} } }
```


Scanned with CamScanner

- **Write a java program using *Swing Package* to create a simple four function arithmetic calculator. Select layout of your own choice?**

```
package javaapplication127;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class JavaApplication127 {
public static void main(String[] args) {

    Cal c1=new Cal();

   }

} class Cal extends JFrame implements ActionListener{
JTextField t1,t2;
JButton b1,b2,b3,b4;
JLabel l1,l2,l3,l4;

Cal(){
   JFrame jf1=new JFrame();

l1=new JLabel("first number");
jf1.add(l1);
t1= new JTextField(10);
jf1.add(t1);
l2=new JLabel("second number");
jf1.add(l2);
t2= new JTextField(10);
jf1.add(t2);
b1=new JButton("+");
jf1.add(b1);
b1.addActionListener(this);
b2=new JButton("-");
jf1.add(b2);
b2.addActionListener(this);
b3=new JButton("/");
jf1.add(b3);
b3.addActionListener(this);
b4=new JButton("*");
jf1.add(b4);
```


Scanned with CamScanner

```
b4.addActionListener(this);
l3=new JLabel("Answer:");
jf1.add(l3);
l4=new JLabel("      ");
jf1.add(l4);
jf1.setVisible(true);
jf1.setSize(350,500);
jf1.setLayout(new FlowLayout());
//jf1.setLayout(new FlowLayout(FlowLayout.LEFT));
jf1.setTitle("calculator");
jf1.setDefaultCloseOperation(3);
}
public void actionPerformed(ActionEvent ae){
   int a = 0,b=0,s;
if (ae.getSource()==b1){
      a=Integer.parseInt(t1.getText());
      b=Integer.parseInt(t2.getText());
       s=a+b;
      l4.setText("sum is "+s);}
else if (ae.getSource()==b2){

      a=Integer.parseInt(t1.getText());
      b=Integer.parseInt(t2.getText());
       s=a-b;
      l4.setText("subtraction is: "+s);}
else if (ae.getSource()==b3){

      a=Integer.parseInt(t1.getText());
      b=Integer.parseInt(t2.getText());
       s=a/b;
      l4.setText("division is: "+s);}
else{
   a=Integer.parseInt(t1.getText());
      b=Integer.parseInt(t2.getText());
      s=a*b;
      l4.setText("multipliation is: "+s);
}
}
}
```


Scanned with CamScanner

Explain the advantage of using *Interfaces* in java? How they are different from *Abstract classes*. Support your answer with an example of java code?

- **Interface:**

An interface in Java is a blueprint of a class.

It has static constants and abstract methods

The interface in Java is a mechanism to achieve abstraction.

There can be only abstract methods in the Java interface, not method body.

It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables.

It cannot have a method body.

It cannot be instantiated just like the abstract class.

Advantages of Using Interface:

An interface in java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

- **achieve full abstraction**
- **achieve multiple inheritance**
- **achieve loose coupling**

Scanned with CamScanner

- **to break up the complex designs and clear the dependencies between objects.**

EXAMPLE:

```
interface printable{
 void print_text();
 }
 class One implements printable{
public void print_text(){
 System.out.println("Hello World");
}
 public static void main(String args[]){
One o= new One();
 o.print_text();
 } }
```

Subjective Part (2016)

- **create a class publication(title ,price. create two classes named books(no of pages)and tape(playing-time) from it. write appropriate constructors print()and get() function for each class.**

Write an application that demonstrates using object of each class.

```
package javaapplication113;
import java.util.Scanner;
public class JavaApplication113 {
   public static void main(String[] args) {
      Publication p1 = new Publication();
      p1.get();
      p1.print();
      Book b1 = new Book();
      b1.get_data();
```


Scanned with CamScanner

```
        b1.print_data();
        Tape t1 = new Tape();
    }

}
class Publication
{
    String title;
    int price;
    void get()
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the title:");
        title = input.next();
        System.out.println("Enter the price:");
        price = input.nextInt();
    }
    void print()
    {
         System.out.println("The title is :"+title);
         System.out.println("The price is :"+price);
    }
}
class Book extends Publication
{
    int no_of_pages;
    void get_data()
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter number of pages :");
        no_of_pages = input.nextInt();
    }
    void print_data()
    {
        System.out.println("The no. of pages are :"+no_of_pages);
    }
}
```


Scanned with CamScanner

```
class Tape extends Publication
{
   Tape()
   {
      System.out.println("It's Playing time:");
   }
}
```

- **Create an abstract class with field for the car make and price. Include get and set methods for these field; the setPrice() methods is abstract. Create two subclasses for individual automobiles marker (for example Mehran, Toyota) and include appropriate setPrice() methods in each class(i.e. Rs600000 or Rs1800000).Finally write an application that uses auto class and sub classes to display information about different car?**

```
package javaapplication144;
import java.util.Scanner;
public class JavaApplication144 {
 public static void main(String[] args) {
  Car obj=new AutoMobilesMehran();
   obj.getPrice();
   System.out.println("car price is:"+obj.price);
   obj.SetPrice();
   System.out.println("Mehran price is:"+obj.price);
   Car obj1=new AutoMobilesToyota();
   obj1.SetPrice();
   System.out.println("Toyota price is:"+obj1.price);
    }

}
abstract class Car{
int price;
void getPrice(){
Scanner input=new Scanner(System.in);
System.out.println("Enter price:");
```


Scanned with CamScanner

```
price=input.nextInt();
}
abstract void SetPrice();
}
 class AutoMobilesMehran extends Car{
void SetPrice(){
price=600000;
}
}
 class AutoMobilesToyota extends Car{
 void SetPrice(){
price=1900000;
}
}
```

- Write down a class time having fields (hrs, min, sec), containing no argument constructor, three argument constructor and copy constructor. Also write down a method inc_time() that increment seconds by 1 if sec are equal to 60 then increment minute by 1 and set seconds to 0. If minutes become equal to 60 then increment hrs by 1 and set min to 0. Write down a method that compare two time objects and return true if they are equal else return false?

```
package javaapplication145;
public class JavaApplication145 {
public static void main(String[] args) {
    Time time1=new Time();
 Time time2=new Time(2,30,59);
 //Coping time1 OBJECT to the time3 OBJECT
 Time time3=new Time(time1);
 time1.inc_time();
 time2.inc_time();
 time1.display_time();
```


Scanned with CamScanner

```
 if(time1.hour==time2.hour && time1.min==time2.min &&
time1.sec==time2.sec)
 System.out.println("TRUE");
 else
 System.out.println("FALSE");

 //Object Copied
 System.out.println("AFTER COPING OBJECT");
 time3.display_time();

    }

}
class Time{
int hour;
int min;
int sec;
Time(){
 hour=2;
 min=30;
 sec=59;
}
Time(int h,int m,int s ){
 this.hour= h;
 this.min=m;
 this.sec=s;
}
Time(Time t){
this.hour=t.hour;
this.min=t.min;
this.sec=t.sec;
}
void inc_time(){
sec++;
if(sec==60){
    sec=0;
    min++;
```


Scanned with CamScanner

```
}
if(min==60){
    min=0;
  hour++;
}
}
 void display_time(){
 System.out.println("Time is");
 System.out.println(hour +":"+min+":"+sec);
 }
}
```

- **What is composition? Explain your answer with example?**

COMPOSITION:

Composition is the design technique to implement has-a relationship in classes. We can use java inheritance or Object composition for code reuse. Java composition is achieved by using instance variables that refers to other objects

Example:

```
package com.journaldev.composition;
public class Job {
private String role;
private long salary;
private int id;
public String getRole() {
return role; }
public void setRole(String role) {
this.role = role; }
public long getSalary() {
return salary; }
public void setSalary(long salary) {
this.salary = salary;
} public int getId() {
return id;
}
public void setId(int id) {
```


Scanned with CamScanner

```
this.id = id;
} }
package com.journaldev.composition;
 public class Person {
 //composition has-a relationship private Job job;
 public Person(){
 this.job=new Job();
job.setSalary(1000L);
}
public long getSalary() {
 return job.getSalary();
 } }
 package com.journaldev.composition;
 public class TestPerson {
 public static void main(String[] args) {
Person person = new Person();
 long salary = person.getSalary();
}
 }
```

- **What is this keyword and its purpose? Give an example?**

What is this keyword and its Purpose:

Keyword THIS is a reference variable in Java that refers to the current object. It can be used to refer instance variable of current class. It can be used to invoke or initiate current class constructor. It can be passed as an argument in the method call.

```
package pkgthis;
 public class This {
 int a,b,sum;
 public This(int a , int b) {
this.a=a;
 this.b=b;
}
 public static void main(String[] args) {
This thi=new This(5,6);
thi.sum=thi.a+thi.b;
 System.out.println("Sum of Integers is :" + thi.sum);
 } }
```


Scanned with CamScanner

Subjective Part (2017)

- **Write a class rectangle with attributes length and width, each of which defaults to 1. Provide member function that calculates the area of a rectangle also provide set and get functions for the length and width attributes. The set function should verify that length and width are each floating point number larger than 0.0 and less-than 20.0?**

```
package javaapplication147;
import java.util.Scanner;
public class JavaApplication147 {
    public static void main(String[] args) {
        Rectangle rec=new Rectangle ();

 rec.set_length();
 rec.set_width();

 System.out.println("Area of Rectangle is : "+ rec.area());
    }

}

 class Rectangle {
 Scanner input = new Scanner(System.in);
 float length,width;
 Rectangle (){
 length=1;
 width=1;
}
 float area(){
 float area=length*width;
 return area;
 }

 void set_length(){
 System.out.println("Enter length");
 float a=input.nextFloat();
if(a>0 && a<20){
```


Scanned with CamScanner

```
length=a;
}
else
System.out.println("Condition Not Satify");
}

void set_width(){
System.out.println("Enter Width");

float a=input.nextFloat();
if(a>0 && a<20){
width=a;
}
else
System.out.println("Condition Not Satify");
}

float get_length(){
return length;
}

float get_width(){
return width;
}
}
```

- Write a class name operator that have only one data member count the class has the following member functions.

- A constructor to initial the count
- Show function to show the count
- Overload ++ operator to increase the count by 1

```
//Creating Operator class
 package operator;
 public class Operator {
int count;
 Operator(){
 count=2;
 }
void show(){
 System.out.println(count);
```


Scanned with CamScanner

```
}
void Overload(){
count++;
System.out.println(count);
}
 public static void main(String[] args) {
 Operator op = new Operator();
 op.show();
 op.Overload();
 } }
```

- **Define a class for bank account that include the following data members.**

Name of the depositor, account number, type of account
-- Balance amount in the count, the class also contain the following member of function
-- A constructor to initial value
--Deposit function to deposit some amount. It should accept the amount as parameter. --With draw function to withdraw an amount after checking the balance. It should accept the amount as parameter .Display function to display name, and balance?

```
//Creating Bank class

package bank_account;
import java.util.Scanner;
public class Bank_Account {
 final String name,acc_type;
 String d_name;
final int account_num;
int bank_bal;
public Bank_Account() {
name="Ubaid";
acc_type="Saving";
d_name="Adeel";
account_num=3415976;
bank_bal=150000;
}
void deposit(int amount){
```


Scanned with CamScanner

```
bank_bal+=amount;
shiningstudy.com
}
void withdraw(int amount){
bank_bal-=amount;
}
void Display(){
System.out.println("Account Name is :: "+ name);
System.out.println("Your Name is :: "+ d_name);
System.out.println("Your Bank Balance is :: "+ bank_bal);
}
public static void main(String[] args) {
Bank_Account bank= new Bank_Account();
Scanner input= new Scanner(System.in);
System.out.println("Please Enter the Depositor Name");
bank.d_name=input.next();
System.out.println("Please Enter an Amount U want to Deposit");
bank.deposit(input.nextInt());
bank.Display();
System.out.println("Please Enter an Amount U want to Withdraw");
bank.withdraw(input.nextInt());
bank.Display();
}
}
```

- **Explain constructor overloading with the help of one example?**

```
public class Name{

String name;

Name(){

name=”Ubaid”;

} //Constructor Overloading

Name(String name){

this.name=name;
```


Scanned with CamScanner

```
}
 public static void main(String[] args){
 Name obj = new Name();
//It will Call Default Constructor
Name obj1 = new Name("Adeel");
 //It will Call Overloaded Constructor
 System.out.println(obj.name);
System.out.println(obj1.name);
}
 }
```

- Write a class local phone that contains an attribute phone to store a local phone number. The class contain member functions to input and display phone number. Write a child class Nat-Phone for national phone numbers that inherits local phone class. It additionally contains an attribute to store city code .It also contain member function to input and show the city code. Write another class IntPhone for international phone numbers that inherit Nat-Phone class. It additionally contain an attribute to store country code .it also contains member functions to input and show the country code?

```
//Creating Localphone class
package localphone;
import java.util.Scanner;
public class Localphone {
 int phone_num;
 Scanner input = new Scanner(System.in);


 void setNum(){
```


Scanned with CamScanner

```
System.out.println("Please enter Your Local Phone Number : ");
phone_num=input.nextInt();
}
void Display(){
System.out.println("Your Phone Number is : " + phone_num);
}
void input(){
}
}
//Creating NatPhone as sub class
package localphone;
shiningstudy.com
public class Natphone extends Localphone {
 int ci_code=0;


 void setCi_code(){
 System.out.println("Please enter Your City Code : ");
 ci_code=input.nextInt();
 System.out.println("Your City Code is : 0"+ ci_code);
 }
}
//Creating IntPhone as sub class
package localphone;
public class Intphone extends Natphone{
```


Scanned with CamScanner

```
int co_code;


void setCo_code(){
System.out.println("Please enter Your Country Code : ");
co_code=input.nextInt();
System.out.println("Your City Code is : 00"+ co_code);
}


public static void main(String[] args) {
Intphone inter= new Intphone();
Natphone nat= new Natphone();
inter.setCo_code();
inter.setCi_code();


nat.setNum();
nat.Display();
}
}
```

Subjective Part (2019)

- create a class Book with fields book name .author and price .Include no argument ,three argument constructors and two methods namely getData() and setData() to set and get three data members. Finally, write an application that instantiate an object of Book class and call both methods?

Scanned with CamScanner

```
package javaapplication148;
import java.util.Scanner;
public class JavaApplication148 {

   public static void main(String[] args) {
     Book b=new Book("the secrets of the
self",950,"Muhammad iqbal");
     b.getdata();
     b.setdata("bal e jibreel", 350,"Allama Iqbal");
   }

}
class Book{
String bookname;
int price;
String author;
Book(){
bookname="the secrets of the self";
price=950;
author="Muhammad iqbal";
}
Book(String b,int p,String a){
this.bookname=b;
this.price=p;
this.author=a;
}
void setdata(String b,int p,String a){
bookname=b;
System.out.print("bookname is:"+bookname);
price=p;
System.out.print("price is:"+price);
author=a;
System.out.print("author is:"+author);
}
void getdata(){
Scanner input=new Scanner(System.in);
```


Scanned with CamScanner

```
bookname=input.next();
System.out.print("bookname is:"+bookname);
price=input.nextInt();
System.out.print("price is:"+price);
author=input.next();
System.out.print("author is:"+author);
}
}
```

- **what is the purpose of toString() method .From which class it is inherited?**

The **method** is used to get a String object representing the value of the Number Object. If the **method** takes a primitive data type as an argument, then the String object representing the primitive data type value is returned.

- **Create an Auto class that consist of two fields wheels and price Include get() and set() method for these fields; the setPrice() method is abstract . Create two subclasses for individual automobiles marker (for example Mehran, Toyota) and include appropriate setPrice() methods in each class(i.e. Rs600000 or Rs1800000).Finally write an application that uses auto class and sub classes to display information about different car?**

REPEATED QUESTION

Scanned with CamScanner