**Question day3**

Questions on Abstract Classes & Interfaces

1. Create an abstract class pen with methods write () and refill () as abstract methods

Ans

**package** com.org.gen.assement3;

**abstract** **class** Pen{

**abstract** **void** write();

**abstract** **void** refill();

}

**class** Fill **extends** Pen{

**void** write()

{

System.***out***.println("Write only");

}

**void** refill()

{

System.***out***.println("refill only");

}

}

**public** **class** Question1 {

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

Pen obj = **new** Fill();

obj.write();

obj.refill();

}

}

1. Use the pen class from Q1 to create a concrete class fountain pen with additional method change Nib ()

Ans

**package** com.org.gen.assement3;

**abstract** **class** Pen{

**abstract** **void** write();

**abstract** **void** refill();

}

**class** Fountainpen **extends** Pen

{

**public** **void** write()

{

System.***out***.println("this is write");

}

**public** **void** refill()

{

System.***out***.println("this is refill");

}

**public** **void** changenib()

{

System.***out***.println("This is change Nib");

}

}

**public** **class** Question2 {

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

Fountainpen obj=**new** Fountainpen();

obj.write();

obj.refill();

obj.changenib();

}

}

1. Create a class monkey with jump ( ) and bite ( ) methods Create a class human whichinherits this monkey class and implements basicanimal interface with eat ( ) and sleep methods

Ans

**package** com.org.gen.assement3;

**class** Monkey

{

**void** jump()

{

System.***out***.println("This is jump");

}

**void** bite()

{

System.***out***.println("This is bite");

}

}

**interface** Basicanimal

{

**public** **void** eat();

**public** **void** sleep();

}

**class** Human **extends** Monkey **implements** Basicanimal

{

**void** jump()

{

System.***out***.println("Human can jump");

}

**void** bite()

{

System.***out***.println("Human can bite");

}

**public** **void** eat()

{

System.***out***.println("Human Can eat");

}

**public** **void** sleep()

{

System.***out***.println("Human Can sleep");

}

}

**public** **class** Question3 {

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

Human obj=**new** Human();

obj.bite();

obj.jump();

obj.eat();

obj.sleep();

}

}

1. Create a class telephone with ( ) , lift ( ) and disconnected ( ) methods as abstract methods create another class smart telephone and demonstrate polymorphism

Ans

**package** com.org.gen.assement3;

**abstract** **class** Telephone

{

**abstract** **void** lift();

**abstract** **void** disconnected();

}

**class** Smarttelephonem **extends** Telephone

{

**public** **void** lift()

{

System.***out***.println("this is lift");

}

**public** **void** disconnected()

{

System.***out***.println("this is disconnected");

}

}

**public** **class** Question4 {

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

Telephone obj=**new** Smarttelephonem();

obj.lift();

obj.disconnected();

}

}

1. Demonstrate polymorphism using using monkey  class from Q3

Ans

**package** com.org.gen.assement3;

**class** Monkey{

**public** **void** jump() {

System.***out***.println("This is jump");

}

**public** **void** bite() {

System.***out***.println("This is bite ");

}

}

**interface** BasicAnimal{

**public** **void** eat();

**public** **void** sleep();

}

**class** Human **extends** Monkey **implements** BasicAnimal{

**public** **void** eat() {

System.***out***.println("you can eat");

}

**public** **void** sleep() {

System.***out***.println("you can sleep");

}

}

**class** poly **extends** Monkey{

**public** **void** jump() {

System.***out***.println("Jump");

}

**public** **void** bite() {

System.***out***.println("bite");

}

}

**public** **class** Question5 {

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

Human obj = **new** Human();

obj.eat();

obj.sleep();

obj.jump();

obj.bite();

}

}

1. Create an interface TVremote and use it to inherit another interface smart TVremote

Ans

**package** com.org.gen.assement3;

**interface** Tvremote

{

**void** on();

}

**interface** SmartTvremort

{

**void** onwithvoice();

}

**public** **class** Question6 {

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

}

}

1. Create a class TV which implements TVremote interface from Q6

Ans

**package** com.org.gen.assement3;

**interface** Tvremote

{

**void** on();

}

**interface** SmartTvremort

{

**void** onwithvoice();

}

**class** Tv

{

**void** on()

{

System.***out***.println("Tv Remote is ON");

}

**void** onwithvoice()

{

System.***out***.println("Smart Tv On on Voice");

}

}

**public** **class** Question7 {

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

Tv obj=**new** Tv();

obj.on();

obj.onwithvoice();

}

}

Given an inteface in1 which includes a method display which takes an integer as input .

interface in1

{

void display(int p);

}

Task is to write a class testClass which implements interface in1 and has a method named display which takes an integer as input and total number of prime numbers between 2 and integer k (including it). Main function in written for you in the editor.

**Input:**  
The first line will contain an integer **T**(number of test cases). Each test case consists of an integer n.

**Output:**  
Print total number of prime numbers between 2 and the given number ( including them ).

**Constraints:**  
1 < = **T** < = 1000  
2<= n <= 104

**Example:  
Input:**  
2  
13

19

**Output:**  
6  
8

Ans

**package** com.org.gen.assement3;

**interface** prime{

**void** display(**int** a);

}

**public** **class** Question8 **implements** prime{

**public** **void** display(**int** a) {

**int** count=0;

**int** total = 0;

**for**(**int** i=2;i<=a;i++)

{

**for**(**int** j=1;j<=i;j++)

{

**if**(i%j==0)

count++;

}

**if**(count==2)

total++;

count=0;

}

System.***out***.println("Total Prime no "+ total);

}

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

prime obj = **new** Question8();

obj.display(13);

}

}