Axis Assignment(Ashish Gupta)

**. Create an abstract class Compartment to represent a rail coach. Provide an abstract function notice in this class.**

public abstract String notice();

Derive FirstClass, Ladies, General, Luggage classes from the compartment class.

Override the notice function in each of them to print notice message that is suitable to the specific type of  compartment.

Create a class TestCompartment.Write main function to do the following:

Declare an array of Compartment of size 10.

Create a compartment of a type as decided by a randomly generated integer in the range 1 to 4.

Check the polymorphic behavior of the notice method.

**package** abstracts;

**public** **abstract** **class** Compartment {

**public** **abstract** String details();

}

**public** **class** First\_class **extends** Compartment {

**public** String details() {

**return** "This is First\_class AC compartment";

}

}

**public** **class** Genral **extends** Compartment{

**public** String details() {

**return** "This is Genral ";

}

}

**public** **class** Ladies **extends** Compartment{

**public** String details() {

**return** "This ladies Compartment";

}

}

**public** **class** Luggage\_class **extends** Compartment{

**public** String details() {

**return** "this is only for luggage";

}

}

**import** java.util.\*;

**public** **class** Test\_Compartment {

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

Compartment[] obj = **new** Compartment[10];

Random obj1 = **new** Random();

**for**(**int** i = 0 ; i<10 ; i++) {

**int** objNum = obj1.nextInt((4-1)+1)+1;

**if** (objNum ==1)

obj[i] = **new** Luggage\_class();

**else** **if**(objNum==2)

obj[i] = **new** Genral();

**else** **if**(objNum==3)

obj[i] = **new** Ladies();

**else** **if**(objNum == 4)

obj[i] = **new** First\_class();

System.***out***.println(obj[i].details());

// obj[i].details();

}

**for** (**int** i = 0; i < 10; i++) {

**if** (obj[i] **instanceof** First\_class)

System.***out***.println("This is First Class" + i);

**else** **if** (obj[i] **instanceof** Ladies)

System.***out***.println("For Ladies " + i);

**else** **if** (obj[i] **instanceof** Genral)

System.***out***.println("For Genral " + i);

**else** **if** (obj[i] **instanceof** Luggage\_class)

System.***out***.println("Only for Luggage " + i);

}

}

}

Output is::

For Genral 0

For Genral 1

For Ladies 2

This is First Class3

For Ladies 4

Only for Luggage 5

Only for Luggage 6

This is First Class7

This is First Class8

For Genral 9

**. Create an abstract class Instrument which is having the abstract function play.**

Create three more sub classes from Instrument which is Piano, Flute, Guitar.

Override the play method inside all three classes printing a message

“Piano is playing  tan tan tan tan  ”  for Piano class

“Flute is playing  toot toot toot toot”  for Flute class

“Guitar is playing  tin  tin  tin ”  for Guitar class

Create an array of 10 Instruments.

Assign different type of instrument to Instrument reference.

Check for the polymorphic behavior of  play method.

Use the instanceof operator to print which object is stored at which index of instrument array.

**package** abstracts;

**public** **abstract** **class** Instrument {

**public** **abstract** String play();

}

**public** **class** Piano **extends** Instrument {

**public** String play() {

**return** "Piano is playing tan tan tan tan";

}

}

**public** **class** Flute **extends** Instrument {

**public** String play() {

**return** "Flute is playing toot toot toot toot";

}

}

**public** **class** Guitar **extends** Instrument{

**public** String play() {

**return** "Guitar is playing tin tin tin";

}

}

**import** java.util.\*;

**public** **class** TestClass {

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

Instrument[] obj = **new** Instrument[10];

Random rand = **new** Random();

**for**(**int** i =0; i<10; i++) {

**int** randomNum = rand.nextInt((3-1)+1)+1;

**if** (randomNum == 1 )

obj[i] = **new** Piano();

**else** **if**(randomNum == 2)

obj[i] = **new** Flute();

**else** **if**(randomNum == 3)

obj[i] = **new** Guitar();

obj[i].play();

}

**for** (**int** i = 0; i < 10; i++) {

**if** (obj[i] **instanceof** Piano)

System.***out***.println("Piano is stored at index " + i);

**else** **if** (obj[i] **instanceof** Flute)

System.***out***.println("Flute is stored at index " + i);

**else** **if** (obj[i] **instanceof** Guitar)

System.***out***.println("Guitar is stored at index " + i);

}

}

}

Output

Piano is stored at index 0

Flute is stored at index 1

Flute is stored at index 2

Guitar is stored at index 3

Flute is stored at index 4

Guitar is stored at index 5

Piano is stored at index 6

Flute is stored at index 7

Guitar is stored at index 8

Piano is stored at index 9

4.What is the output of the pgm

**interface** A

{

**private** **int** i;

}

Out put :: Gives Error.

5.What is the output of the program

**interface** A

{

**void** myMethod();

}

**class** B

{

**public** **void** myMethod()

    {

        System.out.println("My Method");

    }

}

**class** C **extends** B **implements** A

{

}

**class** MainClass

{

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

    {

        A a = **new** C();

        a.myMethod();

    }

}

Output is : My Method.

6.What is the output here

**interface** X

{

**void** methodX();

}

**class** Y **implements** X

{

**void** methodX()

    {

        System.out.println("Method X");

    }

}

Ans= it gives error because interface method must be implemented as public.

7.Will this program execute if no why

**interface** A

{

**int** i = 111;

}

**class** B **implements** A

{

**void** methodB()

    {

        i = 222;

    }

}

Ans = no because interface field static and final by default so cant change after initialized .

8.What is the output

**interface** P

{

    String p = "PPPP";

    String methodP();

}

**interface** Q **extends** P

{

    String q = "QQQQ";

    String methodQ();

}

**class** R **implements** P, Q

{

**public** String methodP()

    {

**return** q+p;

    }

**public** String methodQ()

    {

**return** p+q;

    }

}

**public** **class** MainClass

{

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

    {

        R r = **new** R();

        System.out.println(r.methodP());

        System.out.println(r.methodQ());

    }

}

Output is:: QQQQPPPP

PPPPQQQQ

**9.Can interfaces have constructors?**

**Answer :**

No. Interfaces can’t have constructors.

**10.Is the below program written correctly? If yes, what will be the output?**

**class** A **implements** B

{

**public** **int** methodB(**int** i)

    {

**return** i =+ i \* i;

    }

}

**interface** B

{

**int** methodB(**int** i);

}

**public** **class** MainClass

{

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

    {

        B b = **new** A();

        System.out.println(b.methodB(2));

    }

}

Out put is:: 4.

**11.Can you find out the errors in the following code?**

**interface** A

{

    {

        System.out.println("Interface A");

    }

**static**

    {

        System.out.println("Interface A");

    }

}

Ans = because interface is not initializers

**12.How do you access interface field ‘i’ in the below code?**

**class** P

{

**interface** Q

    {

**int** i = 111;

    }

}

Out put is:: P,Q,i.