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

**package** Assignment;

**import** java.util.Random;

**abstract** **class** A

{

String str;

**public** **abstract** String notice();

}

**class** FirstClass **extends** A

{

**public** String notice()

{

str="Firstclass";

**return** str;

}

}

**class** LadiesClass **extends** A

{

**public** String notice()

{

str="Ladies Class";

**return** str;

}

}

**class** GeneralClass **extends** A

{

**public** String notice()

{

str="Second Class";

**return** str;

}

}

**class** Luggage **extends** A

{

**public** String notice()

{

str="Luggage";

**return** str;

}

}

**public** **class** TestCompartment

{

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

{

String com[]=**new** String[10];

**int** i=1,x;

Random random=**new** Random();

Luggage d=**new** Luggage();

GeneralClass c=**new** GeneralClass();

LadiesClass b=**new** LadiesClass();

FirstClass a=**new** FirstClass();

**while**(i<=4)

{

x=random.nextInt(1,5);

**switch**(x)

{

**case** 1:

{

com[i]=a.notice();

System.***out***.println("Compartment no."+i+" is "+com[i]);

i++;

}

**break**;

**case** 2:

{

com[i]=b.notice();

System.***out***.println("Compartment no."+i+" is "+com[i]);

i++;

}

**break**;

**case** 3:

{

com[i]=c.notice();

System.***out***.println("Compartment no."+i+" is "+com[i]);

i++;

}

**break**;

**case** 4:

{

com[i]=d.notice();

System.***out***.println("Compartment no."+i+" is "+com[i]);

i++;

}

**break**;

**default**:System.***out***.println("invaid compartment");**break**;

}

}

}

}

O/P:

Compartment no.1 is Firstclass

Compartment no.2 is Second Class

Compartment no.3 is Second Class

Compartment no.4 is Firstclass

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

**package** Assignment;

**abstract** **class** instruments

{

String name="";

**abstract** **void** play();

}

**class** Piano **extends** instruments

{

**void** play()

{

System.***out***.println("Piano is playing tan tan tan tan ");

}

}

**class** Flute **extends** Piano

{

**void** play()

{

System.***out***.println("Flute is playing toot toot toot toot ");

}

}

**class** Guitar **extends** Flute

{

**void** play()

{

System.***out***.println("Guitar is playing tin tin tin ");

}

}

**public** **class** MusicalInstruments **extends** Guitar

{

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

{

Piano p=**new** Piano();

Flute f=**new** Flute();

Guitar g=**new** Guitar();

MusicalInstruments a[]=**new** MusicalInstruments[10];

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

{

a[i]=**new** MusicalInstruments();

}

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

{

**if**(i%5==0)

{

a[i].name="Guitar";

}

**else** **if**(i%2!=0)

{

a[i].name="piano";

}

**else**

{

a[i].name="flute";

}

}

**int** j=a.length-1;

**while**(j!=0)

{

**if**(a[j].name.equals("Guitar"))

{

g.play();

}

**else** **if**(a[j].name.equals("flute"))

{

f.play();

}

**else**

{

p.play();

}

j--;

}

}

}

O/P:

Piano is playing tan tan tan tan

Flute is playing toot toot toot toot

Piano is playing tan tan tan tan

Flute is playing toot toot toot toot

Guitar is playing tin tin tin

Flute is playing toot toot toot toot

Piano is playing tan tan tan tan

Flute is playing toot toot toot toot

Piano is playing tan tan tan tan

4.What is the output of the pgm

Ans: No output

5.What is the output of the program

Ans: My Method

6.What is the output here

Ans: throws an error (about abstract inherited class visibility)

7.Will this program execute if no why

Ans :No, the variable defined in an interface cannot be modified by the class that implements the interface

8.What is the output

Ans: QQQQPPPP

PPPPQQQQ

**9.Can interfaces have constructors?**

Ans: No,An interfaces cannot have constructors

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

Ans: Yes,the program written correctly and the output is four(4)

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

Ans: A interface contains only name of the methods

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

Ans: By using its its interface name ,dot operator and variable.

In this case. nested interface is used so ,we have declare a object for the class and

Using object, dot operator and variable is used to access “i”