**2. 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.

[i.e based on the random  number genererated, the first compartment can be Luggage, the second one could be Ladies and so on..]

import java.util.Random;  
  
abstract class Compartment {  
 public abstract String notice();  
}  
class FirstClass extends Compartment {  
 public String notice(){  
 return "You are in First Class";  
 }  
}  
class Ladies extends Compartment {  
 public String notice(){  
 return "You are in Ladies Class";  
 }  
}  
class General extends Compartment {  
 public String notice(){  
 return "You are in General Class";  
 }  
}  
class Luggage extends Compartment {  
 public String notice(){  
 return "You are in Luggage Class";  
 }  
}  
class TestCompartment {  
 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();  
 else if(objNum==2)  
 obj[i] = new General();  
 else if(objNum==3)  
 obj[i] = new Ladies();  
 else if(objNum == 4)  
 obj[i] = new FirstClass();  
  
 System.*out*.println(obj[i].notice());  
 }  
  
 }  
}

**3. 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.

import java.util.Random;  
  
abstract class Instrument {  
 public abstract String play();  
}  
class Piano extends Instrument {  
 public String play(){  
 return "Piano is playing tan tan tan tan ";  
 }  
}  
class Flute extends Instrument {  
 public String play(){  
 return "Flute is playing toot toot toot toot";  
 }  
}  
class Guitar extends Instrument {  
 public String play(){  
 return "Guitar is playing tin tin tin ";  
 }  
}  
class TestInstrument {  
 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();  
  
 System.*out*.println(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);  
  
 }  
 }  
}

4.What is the output of the pgm

**interface** A

{

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

}

**Private is not allowed in interface**

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- My Method**

6.What is the output here

**interface** X

{

**void** methodX();

}

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

{

**void** methodX()

    {

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

    }

}

**methodX() in Y cannot implement methodX() in X because method() in Y is not public**

7.Will this program execute if no why

**interface** A

{

**int** i = 111;

}

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

{

**void** methodB()

    {

        i = 222;

    }

}

**This program will not execute as i is final.**

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-**

**QQQQPPPP**

**PPPPQQQQ**

**9.Can interfaces have constructors?**

**No, you cannot have a constructor within an interface.**

**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));

    }

}

**OUTPUT- 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");

    }

}

**No return type given and no method name declared.**

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

**class** P

{

**interface** Q

    {

**int** i = 111;

    }

}

interface Q  
{  
 int *i* = 111;  
}  
  
class P implements Q  
{  
 public static void main(String[] args) {  
 P obj = new P();  
 System.*out*.println(obj.*i*);  
 }  
}