JAVA ASSIGNMENT 4 SOLUTIONS

By: Vishal Gupta

1) 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..]

Code:-

Compartment.java

```java
import java.util.Random;

6 usages  4 inheritors
public abstract class Compartment {
    1 usage  4 implementations
    public abstract String notice();

}

1 usage
class FirstClass extends Compartment {
    1 usage
    @Override
    public String notice() {
        return "This is FirstClass Compartment";
    }
}

1 usage
class Ladies extends Compartment {

    1 usage
    @Override
    public String notice() {
        return "This is Ladies Compartment";
    }
}

```

```
1 usage
class General extends Compartment{
    1 usage
    @Override
    public String notice() {
        return "This is General Compartment";
    }
}

1 usage
class Luggage extends Compartment{
    1 usage
    @Override
    public String notice() {
        return "This is Luggage COmpartment";
    }
}

class testCompartment{
    public static void main(String[] args) {
        Compartment[] compartments = new Compartment[10];

        Random random = new Random();

        for (int i = 0; i < 10; i++) {
            int randomNum = random.nextInt( bound: (4 - 1) + 1) + 1;

            if (randomNum == 1)
                compartments[i] = new Luggage();
            else if (randomNum == 2)
                compartments[i] = new Ladies();
            else if (randomNum == 3)
                compartments[i] = new General();
            else if (randomNum == 4)
                compartments[i] = new FirstClass();

            System.out.println(compartments[i].notice());
        }

    }
}
```

Output:-

```
testCompartment
"C:\Program Files\Amazon Corretto\jdk11.0.15_9\bin\java.exe"
This is Luggage COmpartment
This is General Compartment
This is Luggage COmpartment
This is Ladies Compartment
This is FirstClass Compartment
This is General Compartment
This is Luggage COmpartment
This is General Compartment
This is General Compartment
This is Ladies Compartment

Process finished with exit code 0
```

2) 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.

Code:-

Instrument.java

```java
import java.util.Random;

5 usages  3 inheritors
public abstract class Instrument {
    1 usage  3 implementations
    public abstract String play();
}

2 usages
class Piano extends Instrument{
    1 usage
    @Override
    public String play() {
        return "Piano is playing  tan tan tan tan";
    }
}

2 usages
class Flute extends Instrument{
    1 usage
    @Override
    public String play() {
        return "Flute is playing  toot toot toot toot";
    }
}

2 usages
class Guitar extends Instrument{
    1 usage
    @Override
    public String play() {
        return "Guitar is playing  tin  tin  tin";
    }
}

class TestInstrument{
    public static void main(String[] args) {
        Instrument[] instruments = new Instrument[10];

        Random rand = new Random();

        for (int i = 0; i < 10; i++) {
            int randomNum = rand.nextInt( bound: (3 - 1) + 1) + 1;

```

```
            if (randomNum == 1)
                instruments[i] = new Piano();
            else if (randomNum == 2)
                instruments[i] = new Flute();
            else if (randomNum == 3)
                instruments[i] = new Guitar();

            instruments[i].play();
        }

        for (int i = 0; i < 10; i++) {
            if (instruments[i] instanceof Piano)
                System.out.println("Piano is stored at index " + i);
            else if (instruments[i] instanceof Flute)
                System.out.println("Flute is stored at index " + i);
            else if (instruments[i] instanceof Guitar)
                System.out.println("Guitar is stored at index " + i);
        }
    }
}
```

Output:-

```
TestInstrument
"C:\Program Files\Amazon Corretto\jdk11.0.15_9\bin\java.exe"
Piano is stored at index 0
Piano is stored at index 1
Guitar is stored at index 2
Guitar is stored at index 3
Piano is stored at index 4
Guitar is stored at index 5
Guitar is stored at index 6
Guitar is stored at index 7
Piano is stored at index 8
Piano is stored at index 9

Process finished with exit code 0
```

3) What is the output of the pgm

```
interface A
{
   private int i;
}
```

Output:-

It will through error, because in interface modifiers can only be public, static or final. Here it modifiers is private which is not allowed.

4) 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

5) What is the output here

```
interface X
{
    void methodX();
}

class Y implements X
{
    void methodX()
    {
        System.out.println("Method X");
    }
}
```

Output:-

Here methodX in class Y should be declared “public”. It will show error at compile time

6) Will this program execute if no why

```
interface A
{
    int i = 111;
}
class B implements A
{
    void methodB()
    {
        i = 222;
```

```
    }
}
```

Output:-

No it will show compile time error, because interface fields are static and final by defaults.
We can not change their values in methodB of class B.

7) 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

8) 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

9) Can you find out the errors in the following code?

```
interface A
{
    {
        System.out.println("Interface A");
    }

    static
    {
        System.out.println("Interface A");
    }
}
```

Output:-

Interface can't have initializer

10) How do you access interface field 'i' in the below code?

```
class P
{
    interface Q
    {
        int i = 111;
    }
}
```

Output:- **P.Q.i**