Programacion Orientada a Objetos

Santiago Abelardo Salcedo Rodriguez

Taller Java 8 parte 2

Ejercicio de Analisis.

1. Implemente una nueva clase llamada Aerodeslizador Esta nueva clase debe implementar las interface de VehiculoTierra y VehiculoAgua con las siguientes condiciones:

a. Defina los atributos necesarios para implementar la clase de acuerdo con la lógica de los métodos que hereda esta clase. Estos atributos deben quedar con el mayor nivel de privacidad para la clase. Tome como ejemplo la clase Sedan. Defina un constructor por defecto que inicialice los atributos con valores predefinidos por usted.

b. Defina otro constructor que inicialice los atributos de esta clase con los valores pasados como parámetros a este constructor.

c. Implemente el cuerpo de los métodos que hereda esta clase teniendo en cuenta que no se deben cambiar los métodos abstractos declarados en las interfaces. Ninguna lógica de estos métodos debe ejecutar la instrucción System.out.println. Esta instrucción será exclusiva del método main de la clase ObjTaller8.

Respuesta=

```
public class Aerodeslizador implements VehiculoTierra, VehiculoAgua {
    private String nombre;
    private short maxPasajeros;
    private int maxVelocidad;
    private int numeroRuedas;

    public Aerodeslizador() {
        maxPasajeros = 2;
        maxVelocidad = 120;
        nombre = "Aerodeslizador Generico";
        numeroRuedas = 0;
    }
```

```
    public Aerodeslizador(short maxPasajeros, int maxVelocidad, String nombre, int numeroRuedas)
{
        this.maxPasajeros = maxPasajeros;
        this.maxVelocidad = maxVelocidad;
        this.nombre = nombre;
        this.numeroRuedas = numeroRuedas;
    }

    @Override
    public String soltarAmarras() {
        return "Se Soltaron las Amarras de "+this.getNombre();
    }
    @Override
    public int getNumeroRuedas() {
        return numeroRuedas;
    }

    @Override
    public void agregarRuedas(int numeroRuedas) {
        this.numeroRuedas = numeroRuedas;
    }

    @Override
    public String conducir() {
        return this.getNombre()+" esta siendo conducido";
    }
    @Override
    public String getNombre() {
        return nombre;
    }

    @Override
```

```
    public int getMaxPasajeros() {
        return maxPasajeros;
    }

    @Override
    public int getMaxVelocidad() {
        return maxVelocidad;
    }
}
```

2. Implemente una nueva clase llamada Fragata Esta nueva clase debe heredar de VehiculoAgua, con las mismas condiciones indicadas en el numeral anterior.

Respuesta==

```
public class Fragata implements VehiculoAgua{

    private String nombre;
    private short maxPasajeros;
    private int maxVelocidad;

    public Fragata() {
        nombre= "Fragata Generica";
        maxPasajeros = 300;
        maxVelocidad = 60;
    }

    public Fragata(short maxPasajeros, int maxVelocidad, String nombre) {
        this.maxPasajeros = maxPasajeros;
        this.maxVelocidad = maxVelocidad;
        this.nombre = nombre;
    }

    @Override
```

```
    public String getNombre() {
        return nombre;
    }

    @Override
    public int getMaxPasajeros() {
        return maxPasajeros;
    }

    @Override
    public int getMaxVelocidad() {
        return maxVelocidad;
    }

    @Override
    public String soltarAmarras() {
        return "Se Soltaron las Amarras de "+ this.getNombre();
    }
}
```

3. Implemente una nueva clase llamada PatrullaPolicia. Esta nueva clase debe heredar de Sedan y Emergencia. Con las siguientes condiciones:

a. Implemente un constructor que inicialice los atributos heredados por la superclase.

b. Implemente los atributos de acuerdo a la lógica de los métodos que está heredando esta clase.

c. Implemente el cuerpo de los métodos abstractos que está heredando esta clase. Tenga en cuenta que el método getVolumen() deben retornar el valor definido por el atributo VOLUMEN en la interface Emergencia. Tenga en cuenta las demás aclaraciones en la condición (c) del numeral (1).

Respuesta=

```
public class PatrullaPolicia extends Sedan implements Emergencia{

    public PatrullaPolicia() {
```

```
    }

    public PatrullaPolicia(String nombre, short maxPasajeros, int maxVelocidad) {
        super(nombre, maxPasajeros, maxVelocidad);
    }

    @Override
    public String sonarSirena() {
        return this.getNombre()+" esta sonando su sirena";
    }

    @Override
    public int getVolumen() {
        return Emergencia.VOLUMEN;
    }

    @Override
    public String toString() {
        return "El nombre de la patrulla es "+getNombre()+" tiene un maximo de pasajeros de
"+this.getMaxPasajeros()+
                " una velocidad maxima de "+this.getMaxVelocidad()+ " y un volumen de la sirena de
"+Emergencia.VOLUMEN;
    }
}
```

d. Implemente el siguiente método dentro de la clase PatrullaPolicia:

```
public void setVolumen() {
Emergencia.VOLUMEN++;}
```

Explique, ¿por qué esta implementación genera error?

Respuesta= Porque este es un método de tipo Final, es decir es una constante y no puede ser modificada una vez se le asigno su valor.

e. Defina el método toString() dentro de la clase PatrullaPolicia y responda:

¿Se está sobrescribiendo el método? ¿De quién se está sobrescribiendo el método toString() en este caso?

Respuesta== Si, este método esta siendo sobreescrito este método esta siendo heredado de la clase padre Sedan, pero en la implementación de Sedan, llama al método toString de su padre pero este no hereda de nadie, esto se debe a que Este método esta siendo heredado de forma implícita de la clase Object, todas las clases heredan de dicha clase aunque en el código no se escriba la instrucción.

4. En el método main de la clase ObjTaller8 realice lo siguiente:

a. Crear las instancias (objetos) de las clases definidas anteriormente de la siguiente manera.

Vehiculo s = new Sedan();

Vehiculo a = newAerodeslizador();

Vehiculo f = new Fragata();

Vehiculo p = new PatrullaPolicia("Patrulla 001", 5, 200);

b. Agregue 4 ruedas a los vehículos de tierra (al sedan y a la patrulla de policía) y 2 ruedas al aerodeslizador, utilizando el método agregarRuedas(). Debe conservar Vehiculo como el tipo de los objetos creados en el literal anterior. NO PUEDE. Cambiar el tipo de los objetos creados anteriormente. Para este punto tenga en cuenta el casting de objetos.

c. Cree el siguiente arreglo:

ArrayList<???????> listaVehiculos = new ArrayList<???????>();

Responda: ¿Qué debe ir en el operador diamante del ArrayList?

Respuesta== Vehiculos, para que pueda Aceptar a todos los objetos de las clases creadas

d. Agregue TODOS los objetos creados en el literal (a) de este numeral al ArrayList con el método add() de la clase ArrayList. P. Ej.: listaVehiculos.add(s);

e. Recorra el arreglo creado con el siguiente ciclo:

```
for (int i = 0; i < listaVehiculos.size();
i++) { //Obtiene cada elemento en el
arreglo y lo almacena //en la variable v.
v = listaVehiculos.get(i);
```

//Implemente nuevo código a partir de esta línea

}

Responda: ¿De qué tipo debe ser la variable v en el código anterior?

Respuesta== De tipo Vehiculo

f. Dentro del ciclo del literal anterior debe mostrar la información que devuelven TODOS los métodos que retornan algún valor de cada objeto. Debe conservar Vehiculo como el tipo de los objetos creados en el literal (a) del numeral (4). NO PUEDE cambiar el tipo del objeto creado anteriormente. Para este punto tenga en cuenta el operador instanceof y los respectivos cast. Por ejemplo:

```
System.out.println("Nombre = " + v.getNombre());
System.out.println("Max Pasajeros = " + v.getMaxPasajeros());
System.out.println("Max Velocidad = " + v.getMaxVelocidad());
.
.
.
```

Respuesta=

```
import java.util.*;
   public class ObjTaller8 {
      public static void main (String[] args) {

         Vehiculo s = new Sedan();
         Vehiculo a = new Aerodeslizador();
         Vehiculo f = new Fragata();
         Vehiculo p = new PatrullaPolicia("Patrulla 001", (short) 5, 200);

         ((Sedan) s).agregarRuedas(4);
         ((PatrullaPolicia)p).agregarRuedas(4);
         ((Aerodeslizador)a).agregarRuedas(2);

         //Defina el operador diamante del siguiente arreglo.
      ArrayList<Vehiculo> listaVehiculos = new ArrayList<Vehiculo>();
```

```
        for (int i = 0; i < listaVehiculos.size(); i++) {
            //Obtiene cada elemento en el arreglo y lo almacena
            //en la variable v.
            Vehiculo v = listaVehiculos.get(i);
            if (v instanceof Sedan){
                Sedan o= (Sedan)v;
                System.out.println( "Nombre ="+o.getNombre()+" maxPasajeros= "+
o.getMaxPasajeros()+" maxVelocidad= "+o.getMaxVelocidad() +"numeroRuedas= "+
o.getNumeroRuedas());
            } else if(v instanceof Aerodeslizador){
                Aerodeslizador o= (Aerodeslizador)v;
                System.out.println( "Nombre ="+o.getNombre()+" maxPasajeros= "+
o.getMaxPasajeros()+" maxVelocidad= "+o.getMaxVelocidad() +"numeroRuedas= "+
o.getNumeroRuedas());
            } else if(v instanceof Fragata){
                Fragata o= (Fragata)v;
                System.out.println( "Nombre ="+o.getNombre()+" maxPasajeros= "+
o.getMaxPasajeros()+" maxVelocidad= "+o.getMaxVelocidad());
            }else if(v instanceof PatrullaPolicia){
                PatrullaPolicia o= (PatrullaPolicia)v;
                System.out.println( "Nombre ="+o.getNombre()+" maxPasajeros= "+
o.getMaxPasajeros()+" maxVelocidad= "+o.getMaxVelocidad() +"numeroRuedas= "+
o.getNumeroRuedas());


            }
    }
}
     }
```

g. Si se quisiera subir el volumen de la sirena de la patrulla de policía usando un método setVolumen(). Responda: ¿Qué debería hacerse y por qué?

R= La interfaz no debería declarar VOLUMEN como public static final. En su lugar, este atributo debe ser un atributo de instancia en la clase PatrullaPolicia. La clase PatrullaPolicia debería tener su propio atributo volumen con un valor inicial y métodos getVolumen() y setVolumen() para modificarlo.

h. La línea 47 de la clase sedan está haciendo un llamado al método toString(). ¿En dónde está definido este método para su invocación.

R= En la clase Object que es heredada por todas las clases, no hay ningún problema con esta invocación.