PROGRAMACIÓN ORIENTADA A OBJETOS - UNALMED 2024 -2

NOTA: Lo que no esta en este archivo respondido, esta implementado en el código

1. Implemente una nueva clase llamada `Aerodeslizador` Esta nueva clase debe implementar las interfaces 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.`

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

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

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

    Explique, ¿por qué esta implementación genera error?

    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?

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?

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?

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());
.
.
.
//Información del resto de métodos definidos para cada clase.
```

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?

RTA: Si se quisiera subir el volumen de la sirena de la patrulla de policía usando un método setVolumen(), se debería eliminar el modificador final del atributo VOLUMEN en la interface Emergencia. Sin embargo, esto violaría las buenas prácticas de diseño al modificar un valor constante definido en una interfaz. En su lugar, sería más adecuado definir una variable de volumen dentro de la clase PatrullaPolicia.

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

RTA: La línea 47 de la clase Sedan está llamando al método toString() definido en la clase Object, la cual es la superclase de todas las clases en Java. Por lo tanto, cuando se llama a super.toString(), se está invocando el método toString() de la clase Object.

PROGRAMACIÓN ORIENTADA A OBJETOS - UNALMED 2024 -2