PROGRAMACIÓN ORIENTADA A OBJETOS - UNALMED 2024 -2

Respuestas

1. Modificar el mensaje del método pitar al crear un objeto Moto sin alterar la clase Auto.

Debes sobrescribir el método pitar en la clase Moto. El código sería:

```
@Overridepublic void pitar() {
   System.out.println("Las motos no pitan");
}
```

Al sobrescribir el método, cualquier llamada a pitar() desde un objeto de tipo Moto usará esta nueva implementación.

2. Agregar una clase Motoneta que hereda de Moto: ¿Evidencia algún problema?

No hay problemas directos al agregar una clase Motoneta que hereda de Moto. Sin embargo:

- Si Moto tiene atributos o métodos marcados como private, no serán accesibles directamente desde Motoneta.
- En un diseño jerárquico más complejo, puede haber problemas de coherencia si no se maneja correctamente la lógica heredada.

Código de ejemplo para Motoneta:

```
class Motoneta extends Moto {
   public Motoneta(String placa, String modelo) {
      super(placa, modelo);
   }
}
```

3. ¿Es posible sobrescribir el método arrancar en la clase Moto? ¿Por qué?

Sí, es posible sobrescribirlo porque el método arrancar en Auto tiene el modificador public. Este modificador permite que el método sea accesible y sobrescrito en cualquier subclase.

Código para sobrescribir:

```
@Overridepublic void arrancar() {
   System.out.println("La moto está arrancando");
}
```

4. En la línea 13 de la clase Moto, ¿por qué puedo utilizar el método pitar?

Puedes usarlo porque pitar está definido como public en la clase Auto, lo que permite que las subclases (y otras clases) lo invoquen directamente.

5. Obtener el número de autos creados desde ObjTaller5H sin modificar el modificador de acceso.

La variable num_autos es public static. Esto permite acceder directamente desde cualquier clase usando el nombre de la clase Auto.

```
System.out.println("Número de autos creados: " + Auto.num_autos);
```

6. En la línea 7 de ObjTaller5H, ¿por qué no puedo utilizar el método adelantar, si este fue heredado?

El método adelantar tiene acceso por defecto (package-private). Esto significa que solo es accesible dentro del mismo paquete (paquete1), pero ObjTaller5H está en paquete2.

Para solucionarlo, puedes cambiar el modificador de acceso de adelantar a protected o public en la clase Auto.

7. En la línea 8, ¿por qué es posible utilizar el método arrancar si la clase Bus no lo define?

La clase Bus hereda de Auto, y el método arrancar está definido como public en Auto. Por tanto, todas las subclases de Auto, incluidas Bus y Moto, pueden usar este método directamente.

8. En la línea 9 de ObjTaller5H, ¿por qué no puedo utilizar el método pitar, si este fue heredado?

El problema radica en que el método pitar está siendo redefinido como un atributo en la clase Auto. Esto causa conflictos con la invocación directa del método desde bus.

Solución: Cambia el nombre del atributo pitar en la clase Auto para evitar la colisión con el método. Por ejemplo:

private String sonidoPitar = "Piiiiii";

9. En la línea 10, ¿qué imprime el método getVelocidad()? ¿Por qué?

Imprime 10, que es el valor de la variable velocidad de la clase Auto. Esto ocurre porque el método getVelocidad() no accede al atributo redefinido en la subclase, sino al atributo original de la clase Auto.

---

10. Obtener el valor de placa, modelo y capacidad:

Debes agregar métodos getter en las clases Moto y Bus. Ejemplo para Moto:

```
public String getPlaca() {
   return this.placa;
}
public String getModelo() {
```

```
    return this.modelo;
}
```

Y en Bus:

```
public String getPlaca() {
    return this.placa;
}
public int getCapacidad() {
    return this.capacidad;
}
```

Llamada desde ObjTaller5H:

```
System.out.println("Placa de la moto: " + moto.getPlaca());
System.out.println("Modelo de la moto: " + moto.getModelo());
System.out.println("Placa del bus: " + bus.getPlaca());
System.out.println("Capacidad del bus: " + bus.getCapacidad());
```

4o