PROGRAMACIÓN ORIENTADA A OBJETOS - UNALMED 2024 -2

Preguntas de análisis

1. Si deseo modificar el mensaje del método pitar al crear un objeto moto sin alterar la clase Auto, ¿qué debo agregarle al código? (Por ejemplo, al llamar el método pitar imprima: Las motos no pitan).

R//=
```
@Override
public void pitar() {
    System.out.println("Las motos no pitan");
}
```

2. Suponga que se agrega una nueva clase al código, class Motoneta, y esta hereda de la clase Moto, ¿evidencia algún problema? ¿Por qué?

R//= No habría un problema inherente con la herencia desde Moto. Sin embargo, si en Moto existieran métodos final (Que en este caso no existe) o constructores privados (Que tampoco es el caso), podría haber restricciones.

3. Suponga que se definió el método:

```
public void arrancar() {
    System.out.println("Arrancando");
}
```

en la clase Moto, ¿es posible sobrescribir el método? ¿Por qué?

R//= Sí, es posible sobrescribirlo, porque el método arrancar en la clase Auto no está declarado como final. Además, está definido como public, lo que permite modificar su implementación en una subclase.

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

R//= Se puede utilizar el método pitar porque Moto hereda de Auto, y este método está declarado como public en la clase base. Los métodos public son accesibles desde cualquier clase.

5. Haciendo una pequeña modificación a la clase Auto y utilizando la variable num_autos, sin modificar su modificador de acceso, ¿cómo puedo obtener el número de autos creados desde la clase ObjTaller5H?

R//= Como num_autos es un atributo static de la clase Auto, puedes acceder a él directamente desde la clase sin necesidad de un objeto. Simplemente hacemos lo siguiente:
System.out.println("Número de autos creados: " + Auto.num_autos);

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

R//= El método adelantar tiene un modificador de acceso predeterminado (package-private), lo que significa que sólo es accesible dentro del paquete paquete1. Dado que ObjTaller5H pertenece al paquete paquete2, no puede acceder al método.

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

R//= Es posible porque la clase Bus hereda de Auto, y arrancar es un método public definido en la clase Auto. Por lo tanto, todos los objetos Bus tienen acceso a este método a través de la herencia.

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

R//= No se puede utilizar el método pitar porque en la clase Auto ya existe otro miembro con el mismo nombre, pero este es un atributo (String pitar). Java no permite la sobrecarga entre atributos

y métodos con el mismo nombre en un contexto de herencia, y esto genera ambigüedad. Cambiar el nombre del atributo o del método resolvería el problema.

9. En la línea 10 de la clase ObjTaller5H, ¿qué imprime el método getVelocidad()? ¿Por qué?

R//= Imprime 10. Esto ocurre porque el método getVelocidad accede al atributo velocidad definido en la clase Auto (que es 10) y no al atributo velocidad de la clase Moto. Esto se debe a que getVelocidad no es sobrescrito en la clase Moto.

10. Si quisiera obtener el valor de la placa de las clases Moto y Bus, además de su modelo y capacidad respectivamente, ¿Que debo agregar al código?

R//= **En la clase Moto:**

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

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

En la clase Bus:

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

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