Taller Java programación orientada a objetos

Santiago Barrientos Medina

Universidad Nacional de Colombia

Programación orientada a objetos

Jaime Alberto Guzmán Luna

Grupo 2

Facultad de Minas

Ingeniería de sistemas e informática

SOLUCION

A). Para poder modificar el método pitar cuando se crea una moto sin alterar la clase auto, debemos hacer la siguiente, que es posible gracias a la herencia, la clase moto hereda todo de la clase auto y con lo siguiente modificamos el método solo para moto y sus hijos

```
@Override
    public void pitar(){
        System.out.println("las motos no pitan");
    }
```

B) si evidencio algún problema, porque cuando creamos la clase motoneta , no tenemos un constructor por defecto en la clase moto, asi que además de crear la clase motoneta , debemos crear un constructor en motoneta que llame al constructor super y le entregue los parámetros que este esta esperando, esto por el hecho, de que moto no tiene constructor por default, asi que si no hacemos esto, podría saltar un error

C) es posible sobrescribir el método porque el método esta como public en la clase auto, asi que la clase moto realmente si lo heredo. Esto es gracias a que con la herencia , java permite desde las subclases modificar métodos de las superclases, es decir , desde la clase hija yo puedo modificar un método heredado de la clase padre

D) en la línea 13 de la clase moto, puedo utilizar el método pitar porque la clase moto heredo el método pitar de la clase auto, se pudo porque el modificador de acceso del método era publica si que moto lo pudo heredar

E) con un método static que nos permita acceder al atributo de autos , llamándolo desde la clase, asi:

```
public static int getNumAutos(){
        return num_autos;
    }
```

F) en la línea 7 de objtaller no se puede usar el método adelantar desde moto, aunque haya sido heredado por el hecho de que cuando moto hereda las cosas de auto, el método adelantar es de paquete , es decir, solo es visible para los que están en el paquete con el y moto esta en otro paquete, asi que podemos decir que es privado para el , y como sabemos , lo privado no se hereda

G)porque en contraste de la respuesta del punto anterior, esto es gracias al modificador de acceso, el método arrancar es publica si que puede ser accesado por alguien que no este en el mismo paquete que la clase auto, cuando creamos la clase bus y la definimos como una hija de la clase auto, esta hereda todo lo que sea publico de la clase padre y si estuviese en el mismo paquete, todo lo que sea de paquete . como arrancar es public, si lo hereda

H) en la línea 9 de la clase objtaller4 si puedo utilizar el método pitar, me puedo dar cuenta de que si puedo utilizar el método pitar porque cuando me pongo a analizar las clases , pasan las siguientes cosas, bus es hija de auto , asi que va a heredar lo que se pueda de auto y si me voy a mirar que se puede heredar pues efectivamente hereda el método pitar de la clase auto , porque en auto este método es public

I) en la línea 10 de la clase objtaller5 he de admitir que he tenido muchas confusiones respecto a lo que se imprime por pantalla . luego de leer un rato las diapositivas, evidencié que hay espacio para dos velocidades, una de la clase hija y otra de la clase padre , cuando yo ejecuto el método getvelocidad, a pesar de que yo haya creado una velocidad solo para moto, esta no es la que se toma , la que realmente se toma es la de la clase padre, es decir , se imprime el numero 10 por pantalla

M) para obtener las placas, agregué lo siguiente:

```
public String getPlaca(){
        return placa;
```

tanto en bus como en moto

y para modelo hice lo siguiente :

```
private String modelo;
```

le añadí a bus el atributo de modelo

y a moto y a bus les puse este método:

```
public String getModelo(){
        return modelo;

    }
```

Para la capacidad hice lo siguiente :

Le añadí a moto el parámetro de la capacidad

```
 private int capacidad;
```

y luego el método que retorna la capacidad, esto lo puse en moto y en bus :

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