EJERCICIO 2 – TALLER 5 DE JAVA

Luis Esteban Rincon Jaimes ---- C.C. 1090384822

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).

Para esto, en la clase Moto debemos sobre escribir el método pitar (heredado de la clase padre “Auto”), colocando el siguiente código:

```
@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?

No, porque la nueva clase llamada “Motoneta” sigue teniendo solamente una clase padre (la clase Moto), por ende, no hay herencia múltiple en este caso y no hay ningún problema.

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?

Sí, porque es un método de una clase padre, y se puede sobrescribir sin problema para que actúe de otra manera si se llama desde la clase Moto (clase hija); además, debemos resaltar que esto es posible porque el método arrancar() de la clase padre no está definido como “final”, por lo que permite la sobreescritura sin inconvenientes.

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

Porque lo está heredando de su clase padre (clase Auto), por ende, es como si ese método pitar() estuviera en la misma clase Moto; lo que hace que se pueda llamar sin crear instancias de la clase Auto.

También cabe aclarar que se puede utilizar sin problema porque es un método público, si fuera privado o de paquete no se podría acceder a él, sin importar que fuera de su clase padre, porque está en un paquete diferente.

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?

Agregando en esta clase un nuevo método (de clase) que retorne el valor de este atributo, el cual va a incrementarse de uno en uno por cada instancia creada de Auto o de sus clases hijas (Bus o Moto) porque es un atributo de clase. Entonces creamos un método de clase de la siguiente forma:

```
public static int getNum_Autos(){
        return Auto.num_autos;
}
```

Con esto, el método anterior retornará 2, debido a que se crearon dos instancias de las clases hijas de Auto, ejecutando en esos dos casos el constructor de esta clase padre, en donde se incrementó el valor del contador num_autos.

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

Porque este es un método de paquete y, a pesar de que haga parte de la clase padre (clase Auto), no se hereda a su clase hija (clase Moto) porque estas clases se encuentran en paquetes diferentes.

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

Porque la clase Bus también hereda de la clase Auto, la cual tiene el método arrancar y demás; por ende, desde una instancia de la clase Bus se puede acceder a los métodos y atributos de la clase Auto sin problema (los que no tengan restricciones de acceso), entre ellos el método arrancar().

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

Sí se puede utilizar porque, efectivamente, fue heredado de la clase Auto a la clase Bus, y no tiene ninguna restricción de acceso al ser de tipo público; por

ende, no hay ningún problema en esa línea y el método imprime “Piiiiii” sin inconvenientes.

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

No imprime nada, debido a que ese método lo que hace es retornar el valor del atributo velocidad de la instancia moto, de la clase Moto (es decir, retorna 30), pero no se imprime en consola porque no se usa el comando System.out.printnln().

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?

Una de las formas más apropiadas es agregando un nuevo método en la clase padre (clase Auto) con el objetivo de sobrescribir este método a nuestro beneficio en sus clases hijas, de la siguiente forma:

En la clase Auto:

```
public String getDatos(){
return "Devolver datos del vehículo";
}
```

En la clase Bus:

```
@Override
public String getDatos(){
return "La placa del Bus es: "+ this.placa + ", y su capacidad es de: "+ this.capacidad + " kg.";
}
```

En la clase Moto:

```
@Override
public String getDatos() {
return "La placa de la Moto es: "+ this.placa + ", y es modelo: "+ this.modelo;
}
```

Con esto, usando el polimorfismo, hemos logrado que el método getDatos() retorne diferentes valores según la instancia que lo llame, y para verlo en consola solo debemos colocar en el main: System.***out***.println(moto.getDatos()); o System.***out***.println(bus.getDatos()); y veremos que se imprimen los resultados esperados, que no son los mismos.