Respuestas:

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/- Para modificar el mensaje del método pitar al crear un objeto moto sin alterar la clase auto debo agregar lo siguiente al 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?

R/- No hay problemas en la herencia de Motoneta a partir de la clase Moto, pero si Motoneta sobrescribe métodos de Moto, hay que usar su respectivo @Override, y si los métodos o atributos en Moto o Auto son private, entonces no serán accesibles desde Motoneta.

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/- Si es posible sobrescribir el método arrancar porque está en la clase padre Auto con al modificador public, lo que lo hace accesible a cualquier clase hija, simplemente antes del método para que se sobrescriba correctamente hay que agregar @Override.

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

R/- En la línea 13 de la clase Moto, se puede utilizar el método pitar porque la clase (hija) Moto hereda de la clase (padre) Auto, y el método pitar está definido como public en la clase padre. Esto hace que sea accesible para todas las clases que heredan de Auto.

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/- La variable num_autos es public y está declarada como static. Esto significa que pertenece a la clase y no a instancias específicas. Desde la clase ObjTaller5H, se puede acceder a ella directamente, 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 visibilidad package-private porque no está declarado como public, protected, ni private. Esto significa que solo puede ser accedido dentro del mismo paquete.

Como Moto y Bus están en el paquete paquete2, y adelantar pertenece a Auto en paquete1, no puedes acceder al método desde la clase ObjTaller5H.

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

R/- El método arrancar está definido como public en la clase base Auto. Esto permite que sea accesible desde cualquier clase que herede de Auto, incluso si no está definido específicamente en Bus.

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

R/- El método pitar está correctamente heredado, pero hay confusión debido al uso incorrecto del nombre de la variable y método pitar en la clase Auto, esto genera conflictos dependiendo de cómo se accede.

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

R/- El método getVelocidad() en Moto devuelve el valor de velocidad de la clase Auto que es igual a 10. Esto ocurre porque getVelocidad() no está sobrescrito en la clase Moto. La variable velocidad=30 de Moto no afecta a este método, ya que es una nueva variable que no sobrescribe la de la clase padre.

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/- Se debe agregar los métodos get en las clases Moto y Bus:

En la clase Moto:

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

En la clase Bus:

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