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). Se debe hacer una sobreescritura del método pitar dentro de la clase Moto, de forma que no se modifica el comportamiento del método en la clase padre.

```
@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? En este caso, al haber hecho override al método pitar, esa modficiación se heredaría también en la clase Motoneta, retornando que "Las motos no pitan".

3. Suponga que se definió el método en la clase Moto, ¿es posible sobrescribir el método? ¿Por qué?

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

Sí sobreescribe el método, porque tienen la misma firma, así que se tomará el método que esté definido a un nivel más cercano.

4. En la línea 13 de la clase moto, ¿Por qué puedo utilizar el método pitar? En la línea se ejecuta `this.pitar();`, dondes this es una palabra clave de Java para interactuar con la clase en la que está definida (en este caso, dentro de la clase Moto), por lo que `this.pitar()` invoca al método pitar mediante la clase Moto (recordemos de la clase Moto hereda de la clase Auto, que tiene definido el método pitar).

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? En la clase Auto se puede crear un método getter que acceda al atributo estático, de forma que desde cualquiera de las instancias se pueda llamar:

```
//en public class Auto...
public int getNum(){
    return num_autos;
}

//en main...
System.out.println(moto.getNum());
```

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

Porque adelantar no fue definido como un método público, sino con privacidad por defecto, lo que significa que solo puede leerse dentro del mismo paquete, pero Moto se definió en otro archivo, por lo que es innaccesible.

7. En la línea 8, ¿por qué es posible utilizar el método arrancar si la clase Bus no lo define? Porque arrancar fue definido como método público, que es accessible dentro y fuera del paquete por las clases que lo hereden.

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

9. En la línea 10 de la clase ObjTaller5H, ¿qué imprime el método getVelocidad()? ¿Por qué? `moto.getVelocidad()` no imprime nada en pantalla, porque no está especificado que deba imprimir, solo retorna. Sin embargo, si se coloca que imprima, devuelve 10 en vez de 30, esto es porque no está sobreescrito el método y está accediendo a la velocidad default de la clase Auto.

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? Como placa, modelo y capacidad son atributos privados, se debe crear un método que los retorne desde la misma instancia: un getter.