Respuestas Taller 5 Java

1. Para modificar el mensaje sin necesidad de modificar el método de la clase Auto (que es la clase padre) lo que debes hacer es redefinir o sobrescribir es método dentro de la clase Moto de una manera más o menos así:

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

   De esta forma, cuando llamas al método pitar desde una instancia de Moto, se llamará al método redefinido en Moto y no al método de Auto.

2. El problema que se evidencia cuando simplemente creas una clase Motoneta que hereda de Moto es que Java te va a pedir que debes de tener un constructor explícito, esto se debe a que Moto que vendría a ser la clase padre de Motoneta tiene definido un constructor, por lo cual es necesario y obligatorio definir un constructor dentro de Motoneta que tiene que inicializar los atributos de Moto usando la palabra reservada “super” de esta manera:

```
public class Motoneta extends Moto {
    public Motoneta(String placa, String modelo){
        super(placa, modelo);
    }
```

   Esto se debe a que Moto necesita atributos para inicializarse, por lo cual Motoneta no podrá heredar de Moto si Moto no está inicializado. En caso de que Moto tuviera un constructor por defecto, entonces no sería necesario definir un constructor en Motoneta ya que Java automáticamente crearía un constructor por defecto con el super().

3. Normalmente, por convención se usa @override antes de sobrescribir un método, sin embargo, si puedes sobrescribir el método de esa manera siempre y cuando la firma del método coincida con la del método que se quiere sobrescribir ya que, en caso de que suceda lo contrario, se creará un nuevo método que no sobrescribe a ninguno y esto abre paso a que cuando llames al método que se supone que sobrescribiste entonces se va a llamar al método original en lugar del que tu querías que se llame.

4. En la línea 13 de la clase Moto puedes utilizar el método pitar porque si bien no está definido dentro de la clase Moto, este es un método que hereda de la clase Auto, por lo cual es capaz de utilizarlo.

5. Para ello puedes crear un método getter para este atributo num_autos (puede ser getNumAutos() por ejemplo) y este método será heredado a todas las subclases que se creen a partir de Auto y así podrás acceder al número de autos desde ObjTaller5H.

6. No se puede usar el método adelantar por el nivel de encapsulamiento que tiene. Al no tener ni public, ni private, ni nada, entonces este método podrá ser solo utilizado dentro del paquete donde se definió, entonces si quieres que se pueda utilizar ese método debes de añadirle public.

7. Porque Bus hereda el método arrancar de la clase Auto, por lo cual no es necesario definirlo dentro de Bus ya que cuando se llame al método desde una instancia de Bus, este inmediatamente utilizará el método descrito en la clase Auto.

8. En realidad, si se puede usar al método pitar en la línea 9 de la clase ObjTaller5H.

9. Cuando llamas al método getVelocidad() en la línea 10 de ObjTaller5H se debería de imprimir 10. Si bien la moto tiene su propio atributo velocidad con un valor de 30, en realidad se está llamando a un método que no está definido dentro de la clase Moto pero si dentro de la clase Auto, por lo cual, se llama a este método de la clase Auto y por ende retorna el valor de velocidad de la clase Auto que corresponde a 10.

10. Bastaría con añadir los métodos getter correspondientes para los atributos que se desean obtener. En el caso de la clase Moto, basta con poner un get para placa y para modelo; en el caso de la clase Bus basta con poner un get para placa y para capacidad.