Ejercicio 2 Taller 5 JAVA.

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 modificar el mensaje del método pitar cuando se cree un objeto de tipo Moto sin alterar la clase Auto (ni de otras clases que dependan de él), se necesita sobrescribir el método pitar en la clase Moto. Entonces, al crear un objeto Moto y llamar al método pitar, se imprimirá "Las motos no pitan", al agregar:

```
@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?

Si se agrega la clase Motoneta, que hereda de Moto, y esta no está bien diseñada, se pueden presentar varios problemas, por ejemplo:

Si la clase Motoneta sobrescribe un método de Moto de manera incorrecta, puede generar comportamientos inesperados o errores.

Los miembros privados de Moto, como placa y modelo, no serán accesibles desde Motoneta a menos que se utilicen métodos getter o setter.

Si se hace un uso incorrecto del polimorfismo, podría haber errores al intentar tratar un objeto de tipo Moto como si fuera Motoneta, si no se manejan bien las referencias o las conversiones.

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

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

Sí, es posible sobrescribir el método arrancar() en la clase Moto. La clase Moto hereda de Auto, y el método arrancar() de Auto es no estático y es público, lo que permite que Moto sobrescriba ese método, ya que una subclase puede sobrescribir métodos heredados de la clase padre para su propio uso. Por esto, en Moto se podría sobrescribir arrancar() para que imprima un mensaje diferente o realice una acción específica para motos.

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

El método pitar() puede ser utilizado en la clase Moto porque la clase Moto hereda de la clase Auto, que tiene un método pitar() público. Al heredar de Auto, la clase Moto tiene acceso a todos los métodos públicos y protegidos de Auto. Además, cuando se crea un

objeto Moto, se invoca this.pitar(), lo que ejecuta el método pitar() de la clase Auto (ya que no existe uno sobrescrito en Moto).

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?
El número de autos creados desde la clase ObjTaller5H puede ser accedido directamente desde la variable estática num_autos de la clase Auto usando Auto.num_autos. Esta variable tiene un modificador de acceso public, se puede leer sin necesidad de modificarlo, ya que es accesible públicamente.

6. En la línea 7 de la clase ObjTaller5H, ¿Por qué no puedo utilizar el método adelantar, si este fue heredado?
No se puede utilizar el método adelantar() en la clase Moto porque este método no está presente en la clase Moto ni en sus subclases, y no se ha sobrescrito en ella. En la clase Auto, el método adelantar() es accesible para sus instancias, pero si es llamado desde un objeto Moto, se debe comprobar que Moto lo haya heredado correctamente o lo haya sobrescrito. Si no está presente, el método no será accesible directamente.

7. En la línea 8 de la clase ObjTaller5H, ¿por qué es posible utilizar el método arrancar si la clase Bus no lo define?
Es posible utilizar el método arrancar() en la clase Bus porque este método es heredado de la clase Auto. Aunque la clase Bus no define explícitamente el método arrancar(), como Bus es una subclase de Auto, hereda todos los métodos públicos de Auto, incluidos los métodos no sobrescritos.

8. En la línea 9 de la clase ObjTaller5H, ¿por qué no puedo utilizar el método pitar, si este fue heredado?
No se puede utilizar el método pitar() porque la clase Bus no sobrescribe el método pitar() de la clase Auto. Aunque el método pitar() es heredado de Auto, el método de la clase Auto no es el mismo que el de la clase Bus, ya que Bus no tiene ninguna implementación propia del método pitar().

9. En la línea 10 de la clase ObjTaller5H, ¿qué imprime el método getVelocidad()? ¿Por qué?
El método getVelocidad() imprime 10. Esto sucede porque la clase Moto tiene una variable velocidad de valor 30, pero la clase Moto no sobrescribe el método getVelocidad(), entonces, se usa el método heredado de la clase Auto, que devuelve el valor de velocidad que está en Auto y que está inicialmente definido como 10.

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?
Para obtener el valor de la placa, modelo y capacidad (en Moto y Bus), se deben agregar los métodos get para estos atributos en las clases correspondientes.
En la clase Moto se agregaría:

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

```
}
public String getModelo() {
    return this.modelo;
}
```

En la clase Bus se agregaría:

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