DESARROLLO EJERCICIO 2:

1.Si se desea cambiar el mensaje del método pitar sin modificar la clase Auto, se puede sobrescribir el método pitar en la clase Moto. Para hacerlo, se debe redefinir el método pitar dentro de la clase Moto, así:

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

2.No habría problema, esta nueva clase Motoneta presentaría polimorfismo y heredaría de la clase auto y la clase moto.

3.Si es posible, además porque se habla de la sobreescritura cuando el método es público de lo contrario se está definiendo un método independiente de la clase padre que solo pertenece a la clase hija.

4.Se puede usar el método pitar porque la clase Moto lo hereda de la clase Auto, además, se está utilizando el this.pitar() en el constructor de Moto, lo cual hace que se ejecute el método correspondiente de la clase Moto si este ha sido sobrescrito.

5. Puedes acceder a la variable estática num_autos de la clase Auto directamente desde la clase ObjTaller5H utilizando el nombre de la clase Auto. Aunque num_autos tiene un modificador de acceso public, puedes acceder a él sin necesidad de un objeto.

6. El método adelantar no está disponible para la instancia de Moto en la línea 7 porque el método adelantar no está definido como public o protected en la clase base Auto. Es probable que este método tenga un modificador de acceso package-private (por defecto), lo que significa que solo es accesible dentro del paquete paquete1. Como la clase Moto está en el paquete paquete2, no puede acceder a este método directamente. Si quieres que adelantar sea accesible desde otros paquetes, debes hacerlo public

7. Es posible usar el método arrancar() de la clase Auto en la línea 8, incluso si la clase Bus no lo define explícitamente, porque la clase Bus hereda de Auto, y Auto tiene el método arrancar disponible. Esto es un ejemplo de herencia en la que los métodos de la clase base (como arrancar) son heredados por las clases derivadas.

8.Si puede Usar el método pitar porque el método esta publico en la clase Auto por lo tanto lo puede heredar.

9. El método getVelocidad() devolverá el valor de la variable velocidad de la clase en la que se encuentra el objeto. En este caso, moto es un objeto de la clase Moto, y Moto tiene una variable velocidad que está definida como int velocidad = 30. Así que el valor impreso será: 30

10.para acceder a la placa de las clases bus y moto se crea el método getPlaca en la clase Auto y luego para la clase ObjTaller5H Se la obtiene para bus y moto respectivamente moto.getPlaca()

bus.getPlaca()

///Definicion del metodo en la clase Auto

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

///Y para Modelo se define el método en la clase Moto y capacidad en la clase Bus como sigue:

///Modelo

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

///para llamar al método

```
moto.getModelo()
}
```

///Capacidad

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

///para llamara al método

bus.getCapacidad()