1. **Modificar el mensaje del método pitar sin alterar Auto:**
   Debes sobrescribir el método pitar en la clase Moto:

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

2. **Nueva clase Motoneta que hereda de Moto:**
   No hay problema con la herencia directa, pero si la clase Moto tiene métodos o atributos específicos que no son útiles para Motoneta, puede haber funcionalidad innecesaria heredada.
3. **Sobrescribir el método arrancar:**
   Sí, es posible sobrescribirlo porque el método en Auto es público, lo que permite modificar su comportamiento en las clases derivadas.
4. **Uso del método pitar en la línea 13 de Moto:**
   Puedes usarlo porque Moto hereda de Auto y pitar es un método público en Auto, accesible desde las clases derivadas.
5. **Obtener el número de autos creados con num_autos:**
   Como num_autos es estático, puedes acceder directamente desde la clase Auto:

```
System.out.println(Auto.num_autos);
```

6. **Método adelantar en la línea 7 de ObjTaller5H:**
   No puedes usarlo porque tiene modificador de acceso *default* (sin especificar) en Auto y solo es accesible desde el mismo paquete (paquete1).
7. **Uso del método arrancar en la línea 8 de ObjTaller5H:**
   Es posible porque arrancar es público en Auto y Bus hereda de Auto, lo que permite usarlo sin necesidad de redefinirlo.

9. **Impresión de getVelocidad en la línea 10:**
   Imprime 30, porque la clase Moto tiene su propia variable velocidad que **oculta** la de Auto. El método getVelocidad accede al atributo de la instancia actual (this.velocidad).

10. **Obtener placa, modelo y capacidad:**
    Agrega métodos *getter* en Moto y Bus:

```
public String getPlaca() {
   return placa;
}
public String getModelo() { // Para Moto
   return modelo;
}
public int getCapacidad() { // Para Bus
   return capacidad;
}
```