PROBLEMAS:

1. Abre un nuevo proyecto llamado '*T3P1 – Bombilla' (Menú Archivo → Nuevo → Proyecto).* Crea la clase Bombilla de las transparencias *(Menú Archivo → Nuevo → Clase),* ahora retoca la clase Bombilla para que cumpla con los siguientes requisitos:

   - Si la bombilla estaba encendida y se vuelve a encender se debe mostrar el mensaje "La bombilla ya estaba encendida" y no aumentar el contador de número encendidos.
   - Si la bombilla estaba apagada y se vuelve a apagar se debe mostrar el mensaje "La bombilla ya estaba apagada".
   - Cuando se produzca el encendido 1000 la bombilla debe fundirse y escribir el mensaje "La bombilla se ha fundido". En este estado, si se intenta encender o apagar se debe mostrar el mensaje "La bombilla está fundida".

   Para probar la clase Bombilla crearemos otra clase *PruebaBombilla (Menú Archivo → Nuevo → Clase)* y escribimos el siguiente código:

```
public class PruebaBombilla {

  public static void main(String[] args) {
        int i;
        Bombilla b = new Bombilla();
        // Creamos un objeto bombilla

        // La encendemos y apagamos 1000 veces
        for(i=1; i<=1000;i++)
        {
              b.encender();
              b.apagar();
        }
        // Y otra vez más, aunque ya debe estar fundida
        b.encender();
        b.apagar();
  }
}
```

2. Abre un nuevo proyecto llamado '*T3P2-4 – Televisor' (Menú Archivo → Nuevo → Proyecto).* A continuación crea y teclea la clase Televisor descrita a continuación:

Televisor
+ marca: String + modelo: String + anio: int // entero entre 1950 y 2200 + panoramica: boolean + stereo: boolean + encendida: boolean

+ volumen: int // entero entre 0 y 100 + canal: int // entero entre 0 y 99
+ void encender() + void apagar () + void seleccionarCanal (int nuevoCanal) + int obtenerCanal () + void subirCanal () + void bajarCanal () + void cambiarVolumen (int nuevoVolumen) + void imprimirCaracterísticas ()

Sabiendo que:

- El método "imprimirCaracteristicas" muestra en pantalla el valor de todas las propiedades del objeto.
- Si al cambiar el volumen o el canal de la televisión, el nuevo valor está fuera del rango permitido se debe mostrar un mensaje de error y dejar el valor que hubiera previamente.
- Si se intenta encender o apagar dos veces consecutivas el televisor no se debe mostrar ningún mensaje.
- Si el televisor está apagado sólo podremos encenderlo o imprimir sus características. En el resto de los métodos escribiremos: "Televisor apagado"
- Para el resto de los casos de los métodos se debe mostrar un mensaje que indique la acción que se ha realizado.

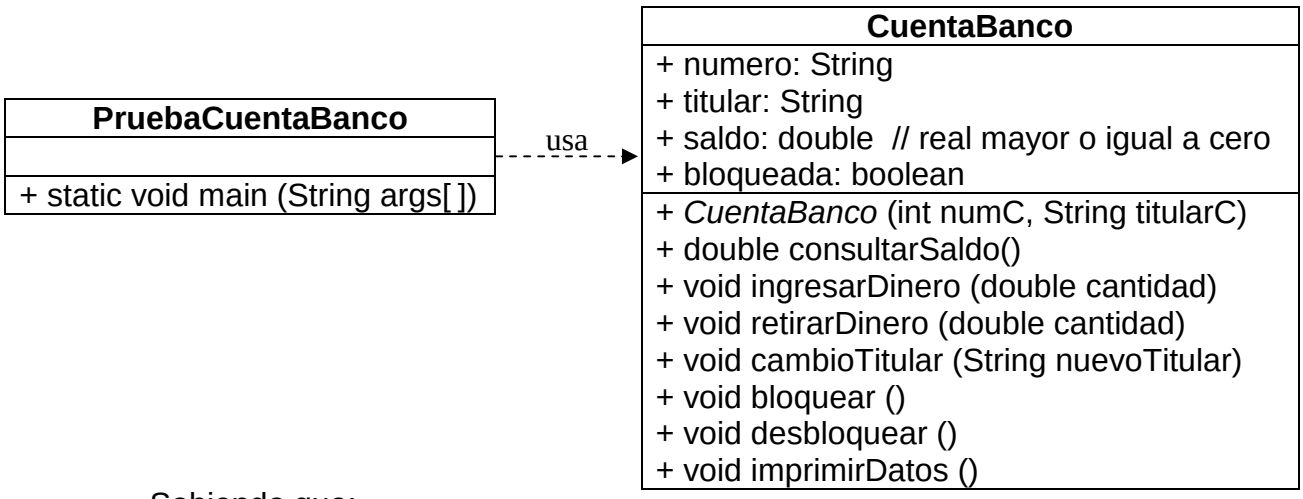
3. Añadir a la clase anterior un método constructor con el siguiente prototipo:
   - public Televisor (String marcaInicial, String modeloInicial, int anioIni)

   Sabiendo que:
   - Si el valor de inicio del año se sale del rango permitidos entonces se le pone 2000.
   - El resto de las propiedades deben inicializarse con los valores por defecto que pone el compilador.

4. Añadir una clase llamada *PruebaTelevisor* al proyecto del ejercicio anterior que debe verificar el buen funcionamiento de los objetos de tipo Televisor. Para ello haremos la siguiente secuencia de acciones:
   - Crearemos un televisor de la marca "Sony", modelo "Trinitron 4" del año 2003.
   - A continuación imprimiremos sus características.
   - Encendemos el televisor e imprimimos el canal seleccionado.
   - Ahora bajamos de canal (debe dar un mensaje de error)

- Seleccionamos el canal 23 y después subimos de canal.
- Ahora intentamos poner el volumen a 300 (debe dar un error).
- Ponemos el volumen a 50 e imprimimos las características de nuevo.
- Apagamos el televisor
- Seleccionamos el canal 60 (debe decir que está apagado)
- Y por último apagamos el televisor otra vez.

5. Abre un nuevo proyecto llamado 'T3P5 – CuentaBanco' en el que vamos a crear las clases del siguiente diagrama:



Sabiendo que:

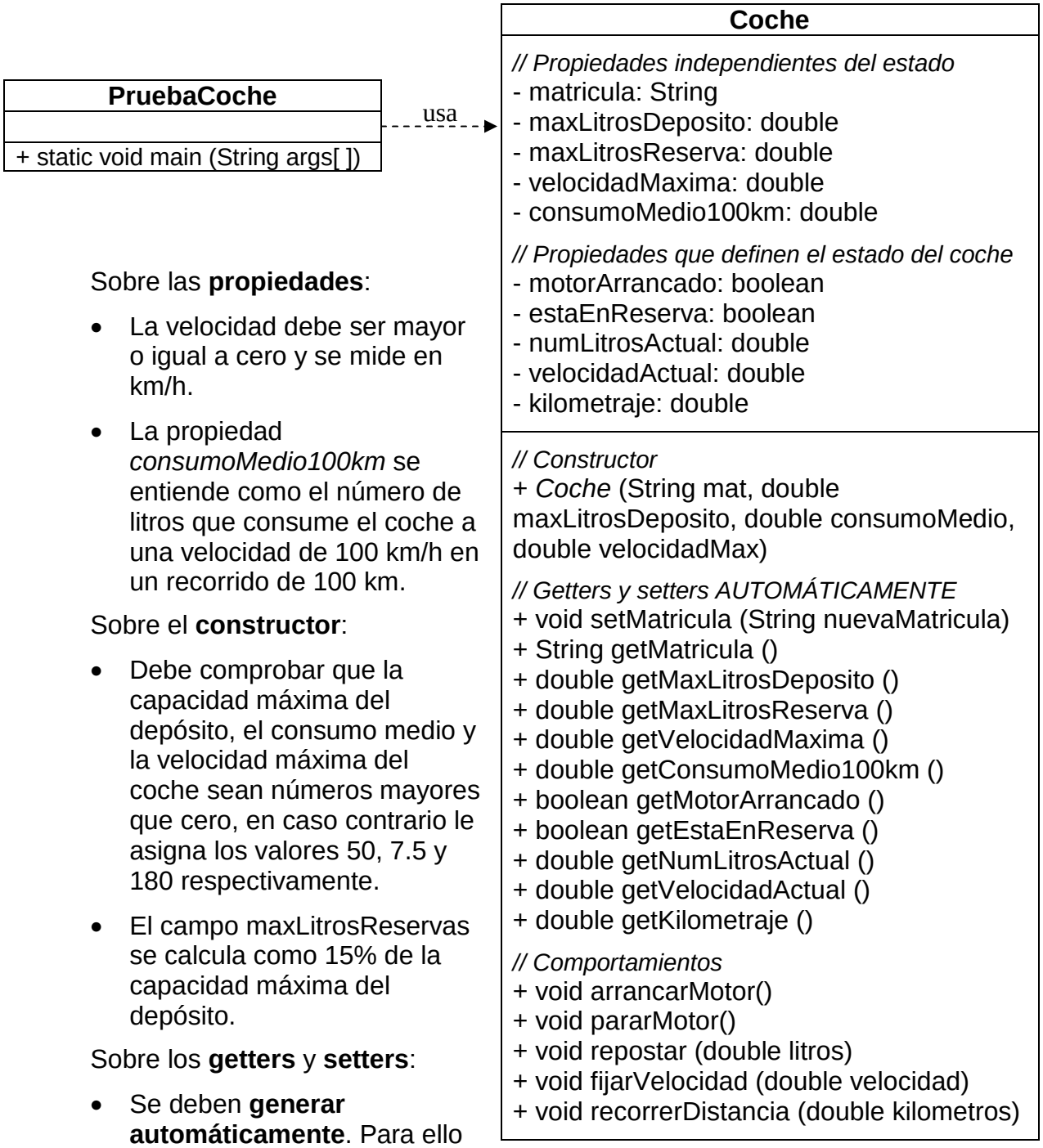
- Las cuentas se crean desbloqueadas y con saldo cero
- Si se intenta retirar más dinero del que hay en la cuenta, la operación queda sin efecto y se debe emitir un mensaje de error.
- Si una cuenta está bloqueada sólo podremos realizar las siguientes operaciones: imprimir los datos de la cuenta, desbloquearla y consultar su saldo. En el resto de las operaciones se debe emitir un mensaje de error indicando que la cuenta está bloqueada.
- La operación de cambio de titular debe asegurarse que la cadena del nombre del nuevo titular no esté vacía.

  Para poder hacer una comparación de cadenas debemos utilizar el método **equals** definido en la clase String de la siguiente forma:

  if (nuevoTitular**.**equals ("")==**true**)

  // El nombre contenido en *nuevoTitular* es la cadena vacía
- La clase *PruebaCuentaBanco* sólo cuenta con el método *main* que debe crear uno o varios objetos de tipo CuentaBanco y probar el correcto funcionamiento de dicha clase.

6. Abre un nuevo proyecto llamado 'T3P6 – Coche' en el que vamos a crear las clases del siguiente diagrama:



Sobre las **propiedades**:

- La velocidad debe ser mayor o igual a cero y se mide en km/h.
- La propiedad *consumoMedio100km* se entiende como el número de litros que consume el coche a una velocidad de 100 km/h en un recorrido de 100 km.

Sobre el **constructor**:

- Debe comprobar que la capacidad máxima del depósito, el consumo medio y la velocidad máxima del coche sean números mayores que cero, en caso contrario le asigna los valores 50, 7.5 y 180 respectivamente.
- El campo maxLitrosReservas se calcula como 15% de la capacidad máxima del depósito.

Sobre los **getters** y **setters**:

- Se deben **generar automáticamente**. Para ello primero hay que declarar las propiedades y después seleccionamos:

  Menú Código fuente → Generar métodos de obtención y establecimiento

  y marcamos las opciones que queramos.

Sobre los **comportamientos**:

- Si un método debe escribir algún mensaje en la pantalla, éste comenzará con el encabezado: **"El coche con matrícula *matricula bla bla blá"*** de esta forma podremos manejar varios objetos coches y los podremos distinguir por su matrícula.
- **Arrancar** el motor:
  - Sólo se arranca si queda algo de combustible, en cuyo caso escribe en pantalla: El coche con matrícula *matricula* ha arrancado.
  - Además si está en reserva debe escribir: El coche con matrícula *matricula* está en reserva de combustible
  - Si no queda combustible o ya estaba arrancado se debe escribir un mensaje que describa la situación.
- **Parar** el motor:
  - Si ya estaba en marcha se para y se escribe un mensaje diciendo que se ha parado el motor.
  - Si no estaba en marcha no hacemos nada.
- **Repostar** combustible:
  - Si lo que queda en el depósito más los litros que se echan superan la capacidad del tanque entonces llenamos el tanque hasta arriba y se muestra el mensaje: El coche con matrícula *matricula* ha rebosado el depósito.
  - En caso contrario se añaden los litros a los que ya había en el tanque.
  - Si el parámetro de los litros es negativo no cambiamos nada.
  - En cualquier caso el método debe imprimir el mensaje: El coche con matrícula *matricula* tiene *numero* litros de combustible.
- **Fijar velocidad**:
  - Solo cambiamos podemos fijar la velocidad del coche si el motor está arrancado, en caso contrario mostramos un mensaje que describa la situación.
  - Si la velocidad es mayor que la velocidad máxima del coche, entonces la fijamos a la velocidad máxima.
  - Si la velocidad es negativa la ponemos a cero.
  - Si cambiamos la velocidad debemos escribir un mensaje en pantalla.
- **Recorrer distancia**:
  - Si el motor está parado o bien está arrancado pero la velocidad es 0 entonces mostramos un mensaje y terminamos.
  - Si la distancia es negativa o cero mostramos un mensaje de error y terminamos.

- En caso contrario tenemos que preguntarnos si con el combustible que quede y a la velocidad que hemos fijado somos capaces de recorrer tal distancia o nos quedamos sin combustible.
- Para ello vamos a suponer que si voy a 110 km/h consumo un 10 % más del valor de la propiedad *consumoMedio100km* de esta forma podríamos calcular el **consumo instantaneo** del coche así:

  consumoMedio100km * (1 + (velocidadActual – 100) / 100)
- Ahora nos preguntamos cuántos litros harían falta para cubrir esa distancia yendo a esa velocidad. Para ello declaramos otra variable *litrosNecesarios* cuyo valor se calcula así:

  litrosNecesarios = distancia * consumoInstantaneo / 100
- Si el número de litros necesarios es menor que lo que queda en el depósito entonces podemos recorrer la distancia sin problemas. Actualizamos el kilometraje y los litros del depósito y escribimos el mensaje:

  El coche con matricula *matricula* ha recorrido *distancia* kilómetros

  Además si el coche ha entrado en reserva después del consumo de combustible hay que cambiar la propiedad *estaEnReserva*
- En caso contrario nos vamos a quedar sin combustible en un determinado punto del trayecto. La **distancia real recorrida** se calcula como:

  distanciaReal = 100 * numLitrosActual / consumoInstantaneo
- Solo nos queda añadir esa distancia al kilometraje del coche, poner el depósito a cero, marcar que hemos entrado en reserva y mostrar los siguientes mensajes:

  El coche con matrícula *matricula* ha recorrido *distancia* kilómetros

  El coche con matrícula *matricula* esta sin combustible

  El coche con matrícula *matricula* esta parado.

Sobre la clase **PruebaCoche**:

- Escribir un método main que cree un coche con matrícula “5466-FNZ”, con 60 litros de capacidad de depósito, con un consumo medio de 7.1 litros y con una velocidad máxima de 200 km/h.
- Después escribir el código necesario para repostar 15 litros de combustible, arrancarlo, fijar la velocidad a 80 km/h y recorrer 10 km.
- Por último, fijar la velocidad a 120 km/h e intentar recorrer 300 km.

7. Abre un nuevo proyecto llamado 'T3P7 – CuentaBancoEstatica'. A continuación vamos a importar la clase **CuentaBanco** del ejercicio 5 para ello hacemos:

   Menú Archivo → Importar → Seleccionamos '*Sistema de archivos*' → Siguiente → Examinar → Buscamos la carpeta del 'workspace', dentro de ésta, la del problema 5, hasta llegar a los archivos .java → Marcamos los archivos '*CuentaBanco.java*' y '*PruebaCuentaBanco.java*' → Finalizar

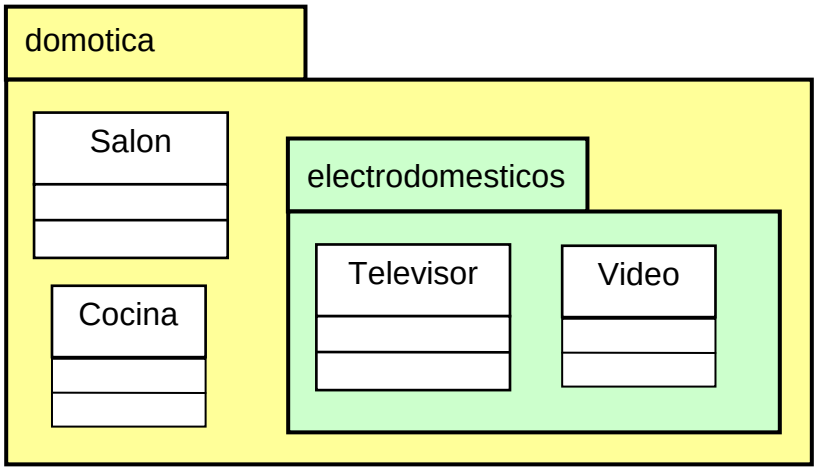
   Lo que hay que hacer es añadir a la clase CuentaBanco los **métodos estáticos**:

   - public static int getNumeroTotalCuentas() que debe devolver el número de objetos cuentas que se han creado de la clase CuentaBanco.
   - public static double getSumaSaldosCuentas() que debe devolver el sumatorio de los saldos de todos los objetos CuentaBanco creados.
   - Además vamos a **sobrecargar el constructor** de forma que ahora recibamos como parámetros de entrada: el número de la cuenta a crear, el nombre del titular, el saldo inicial y su estado de bloqueo.

   Y cambiar el main de *PruebaCuentaBanco* para que:

   - Cree tres cuentas distintas (con el constructor que se desee) y se le ingrese 1000 euros a cada una.
   - Después se imprimirán los valores retornados por los dos métodos estáticos creados.
   - Por último, retiraremos 100 euros de cada una y volveremos a imprimir los valores devueltos por los métodos estáticos.

8. Abre un nuevo proyecto llamado 'T3P8 – Paquetes' en el que vamos a crear el siguiente diagrama:



Para ello seguiremos los siguientes pasos:

- Crearemos un paquete en el proyecto actual: Menú Archivo → Nuevo → Paquete → Nombre: *domotica*
- Ahora repetimos la operación para el paquete de nombre *domotica.electrodomesticos*
- A continuación crearemos las **clases vacías**: Salon, Cocina, Televisor y Video cada una dentro de su paquete.

Para visualizar la estructura de carpetas y ficheros que se ha creado, abrimos la carpeta **c:\workspace\T3P8 – Paquetes**