Ejercicio 1:

Para el siguiente supuesto práctico indicar:
El concesionario de coches TuCocheFavorito está especializado en la venta y mantenimiento de vehículos y desea crear una intranet que facilite el trabajo colaborativo entre sus vendedores, recepcionistas y mecánicos. El procedimiento ideado es crear equipos de trabajo de tres o cuatro personas de cada sección (vendedores, recepcionistas y mecánicos) dirigidos por un coordinador, cada trabajo será asignado a un miembro del equipo y los demás tendrán acceso al documento pudiendo leer, aportar o modificar su contenido. El sistema deberá tener un control de modificaciones y evitar los accesos simultáneos, el coordinador deberá conocer el estado del informe y habrá un control de tiempos. El cliente tiene prisa y desea que se le vayan entregando partes de la aplicación, comenzando con la principal que es el trabajo colaborativo en la elaboración de informes. El cliente quiere que haya controles de acceso y que tan solo los miembros de un mismo equipo puedan acceder a sus documentos. En una segunda versión se implementará un sistema de correo instantáneo entre los miembros de un mismo equipo y de correo normal entre todo el personal del concesionario. Se deberá velar por la confidencialidad, estableciendo los mecanismos de seguridad necesarios.

a) Indicar un lenguaje de programación para el desarrollo de este proyecto en el que se precise de una máquina virtual para la ejecución de su código intermedio.
Principales lenguajes de programación orientados a objetos: Ada, C++, VB.NET, Delphi, Java, PowerBuilder, etc.

De todos ellos el que utiliza una máquina virtual por excelencia, es Java (Java Virtual Machine ó Máquina Virtual de Java). Una máquina virtual es un tipo especial de software cuya misión es separar el funcionamiento del ordenador de los componentes hardware instalados. Esta capa de software desempeña un papel muy importante en el funcionamiento de los lenguajes de programación, tanto compilados como interpretados.
Con el uso de máquinas virtuales podremos desarrollar y ejecutar una aplicación sobre cualquier equipo, independientemente de las características concretas de los componentes físicos instalados. Esto garantiza la portabilidad de las aplicaciones.
Características de las máquinas virtuales:

a) La máquina virtual aísla la aplicación de los detalles físicos del equipo en cuestión. Cuando el código fuente se compila se obtiene código objeto (bytecode, código intermedio). Para ejecutarlo en cualquier máquina se requiere tener independencia respecto al hardware concreto que se vaya a utilizar. La máquina virtual funciona como una capa de software de bajo nivel y actúa como puente entre el bytecode de la aplicación y los dispositivos físicos del sistema.
b) La máquina virtual verifica todo el bytecode antes de ejecutarlo.
c) La máquina virtual protege direcciones de memoria.

b) ¿Qué herramientas necesitaríamos usar para desarrollar este proyecto?
Para desarrollar el proyecto solicitado necesitaremos un entorno *integrado de desarrollo* (IDE) y herramientas CASE, que es un tipo de software compuesto por un conjunto de herramientas de programación. En concreto, el IDE se compone de:

- Editor de código de programación.

- Compilador.
- Intérprete.
- Depurador.
- Constructor de interfaz gráfico.

El IDE elegido es NetBeans en su última versión, que a día de hoy, es la 12.5. Para la instalación de este IDE, tal y como se recoge en su página web oficial:
"Apache NetBeans 12.5 runs on JDK LTS releases 8 and 11, with experimental support for JDK 17, i.e., the current JDK release at the time of this NetBeans release."
Necesitaremos instalar JDK (Java Development Kit) y elegiremos su última versión recomendada que es la 11 (la 17 tiene soporte experimental, por lo que podría ocurrir que algunas herramientas no funcionen correctamente).

c) ¿En qué fase del desarrollo de software nos encontraríamos si estamos dividiendo el sistema en partes y determinando la función de cada una de ellas?
En la fase de diseño que es cuando se divide el sistema en partes y se determina la función de cada una, es decir, decidir qué hará exactamente cada parte. En definitiva, debemos crear un modelo funcional-estructural de los requerimientos del sistema global, para poder dividirlo y afrontar las partes por separado.
En este punto, se deben tomar decisiones importantes, tales como:

- Entidades y relaciones de las bases de datos.
- Selección del lenguaje de programación que se va a utilizar.
- Selección del SGBD.
- Definición de diagrama de clases.
- Definición de diagrama de colaboración.
- Definición de diagrama de paso de mensajes.
- Definición de diagrama de casos de uso.
- Etc.

Ejercicio 2:

Explica detalladamente y de forma clara en qué se diferencia la *desactivación y la desinstalación* de un plugin en NetBeans.

En la desactivación, el módulo o plugin sigue instalado, pero en estado inactivo (no aparece en el entorno) y no consume recursos.

En la desinstalación, el módulo o plugin se elimina físicamente del entorno de forma permanente.

Ejercicio 3:

```
package bono;
```

```
public class Bono {


private int viajes;//viajes disponibles en el bonobus
private int precio;//precio del viaje
private String propietario;//titular del bonobus
private int viajes_max;//número máximo de viajes en el bonobus


/*Constructor por defecto*/
public Bono(){}


/*Constructor por parámetros*/
public Bono(int viajes, int precio, String propietario,int viajes_max){
        this.viajes=viajes;
        this.precio=precio;
        this.propietario=propietario;
        this.viajes_max=viajes_max;
}


/*Método que permite obtener el número disponible de viajes en el bonobus*/


public int obtenerViajes(){return viajes;}


/*Método pàra modificar el número de viajes disponible en el bono*/


public void actualizarViajes(int viajes){this.viajes=viajes;}


/*Método que permite pagar viajes con el bonobus, es necesario que el bono contenga suficientes
viajes y que el número de viajes que se desean pagar sea positivo. */


public void pagarViajes(int viajes) throws Exception{
        if (this.viajes < viajes) {
        throw new Exception("No hay suficientes viajes en el bono");
}
        this.viajes=this.viajes - viajes;
}


/*Método que permite recargar el bonobus, se debe tener dinero suficiente y no se puede superar el
número máximo de viajes que tiene permitido el bonobus. Este método se probará con JUnit*/


public void recargarBono(int viajes, int dinero) throws Exception{
        if (dinero<=0){
        throw new Exception("Se necesita una cantidad positiva de dinero");
        }
        if (viajes<=0){
        throw new Exception("Es necesaria una cantidad positiva de viajes");
        }
        if (this.viajes + viajes > viajes_max) {
        throw new Exception("No se puede superar el número máximo de viajes del bono");
        }
        if (dinero < viajes * precio){
        throw new Exception("No tiene dinero suficiente");
        }
        this.viajes=this.viajes+viajes;
        }
}
```

Dada la clase definida al final de este examen, realiza los siguientes ejercicios:

1. Para un bono Bono(10,1,"Antonio Sánchez Sánchez",20) realiza un test de pruebas de JUnit para el método recargarBono teniendo 2€ de dinero e intentando recargar 5 viajes.

```
@Test
public void testRecargarBonoNoValido() throws Exception {
        System.out.println("Test para probar el método recargarBono con un valor"
        + " dinero insuficiente");
        // Con 2 de dinero no se pueden comprar 5 viajes.
        int viajes = 5;
        int dinero = 2;
        Bono instance = new Bono(10,1,"Antonio Sánchez Sánchez",20);
        try {
                instance.recargarBono(viajes, dinero);
                fail("Deberá haber saltado una excepción");
        } catch (Exception e){
                assertTrue(instance.obtenerViajes()==10);
                //Al no tener dinero suficiente no deben cambiar los viajes disponibles
        }
        System.out.println("Los viajes disponibles: "+ instance.obtenerViajes());

         }
```

2. Si ejecutásemos el test de pruebas del ejercicio anterior (suponiendo que está correctamente desarrollado), ¿la prueba pasaría o sería fallida? Justifica la respuesta (de no estar justificada no se dará por válida).

Si el método dispara la excepción, debe pasar los test. Si el método funciona bien no se realizará la recarga, pues el dinero es insuficiente.