```
import java.util.*;
import java.io.*;
/**
 * Clase taller, lleva la gestión de todos los elementos que intervienen en un taller.
 * .
 * @author Ricardo García
 * @version 15/05/2017
 */
public class taller
{

    private List<Vehiculo> listaDeVehiculos;
    private List<Revision> listaDeRevisiones;
    private List<Cliente> listaDeClientes;
    private List<Mecanico> listaDeMecanicos;
    private Comercial comercialDelTaller;
    private JefeDeTaller JefeDeTaller;



    /**
     * Constructor for objects of class Taller
     */
    public taller()
    {
        // initialise instance variables
        this.listaDeVehiculos = new ArrayList<Vehiculo>();
        this.listaDeRevisiones = new ArrayList<Revision>();
        this.listaDeClientes = new ArrayList<Cliente>();
        this.listaDeMecanicos = new ArrayList<Mecanico>();
    }

    private void guardarDatos()
    {
        SerializeObject(listaDeVehiculos, "vehiculos.ser");
        SerializeObject(listaDeRevisiones, "revisiones.ser");
        SerializeObject(listaDeClientes, "clientes.ser");
        SerializeObject(listaDeMecanicos, "mecanicos.ser");
    }
```

```
private void cargarDatos()
{
    DeserializeObject(listaDeVehiculos, "vehiculos.ser");
    DeserializeObject(listaDeRevisiones, "revisiones.ser");
    DeserializeObject(listaDeClientes, "clientes.ser");
    DeserializeObject(listaDeMecanicos, "mecanicos.ser");
}

private void SerializeObject(Object e, String rutaDelArchivo)
{
     try {
     FileOutputStream fileOut =
     new FileOutputStream(rutaDelArchivo);
     ObjectOutputStream out = new ObjectOutputStream(fileOut);
     out.writeObject(e);
     out.close();
     fileOut.close();
     System.out.printf("Serialized data is saved in " + rutaDelArchivo);
    }
    catch(IOException i) {
     i.printStackTrace();
    }
}
private Object DeserializeObject(Object e, String rutaDelArchivo)
{

  try {
     FileInputStream fileIn = new FileInputStream(rutaDelArchivo);
     ObjectInputStream in = new ObjectInputStream(fileIn);
     e = in.readObject();
     in.close();
     fileIn.close();
  }catch(IOException i) {
     i.printStackTrace();
     return e;
  }catch(ClassNotFoundException c) {
     System.out.println("Class not found");
     c.printStackTrace();
     return null;
```

```
  }
  return null;
}
public static void main (String [ ] args)
{
    taller miTaller = new taller();
    miTaller.cargarDatos();
    ConsoleUI console = new ConsoleUI(miTaller);
    console.mainMenu();
    miTaller.guardarDatos();
}

Vehiculo darDeAltaVehiculo(Vehiculo vehiculo, Cliente cliente)
{
    cliente.agregarVehiculo(vehiculo);
    listaDeVehiculos.add(vehiculo);
    return vehiculo;
}
void darDeAltaCliente(Cliente cliente)
{
    listaDeClientes.add(cliente);
}

void darDeAltaRevision(Vehiculo vehiculo, Cliente cliente)
{
    listaDeRevisiones.add(new Revision(vehiculo, cliente));

}
List<Vehiculo> obtenerListaVehiculos()
{
    return listaDeVehiculos;
}
List<Cliente> obtenerListaClientes()
{
    return listaDeClientes;
}

Revision obtenerSiguienteRevision(Mecanico mecanico)
{
```

```
        for(Revision revision : listaDeRevisiones)
        {
            if(revision.getEstadoRevision() == EstadoRevision.PENDIENTE &&
revision.getMecanicoAsignado() == mecanico)
             return revision;
        }

        return null;
    }

    List<Revision> obtenerListaRevisionesPendientes(Mecanico mecanico)
    {
        ArrayList<Revision> listaDeRevisionesPendientes = new ArrayList<Revision>();
        for(Revision revision : listaDeRevisiones)
        {
            if(revision.getEstadoRevision() == EstadoRevision.PENDIENTE &&
revision.getMecanicoAsignado() == mecanico)
             listaDeRevisionesPendientes.add(revision);
        }

        return listaDeRevisionesPendientes;
    }

    List<Revision> obtenerListaRevisionesPendientes()
    {
        ArrayList<Revision> listaDeRevisionesPendientes = new ArrayList<Revision>();
        for(Revision revision : listaDeRevisiones)
        {
            if(revision.getEstadoRevision() == EstadoRevision.PENDIENTE)
             listaDeRevisionesPendientes.add(revision);
        }

        return listaDeRevisionesPendientes;
    }

    List<Revision> obtenerListaRevisionesRealizadas()
    {
        ArrayList<Revision> listaDeRevisionesRealizadas = new ArrayList<Revision>();
        for(Revision revision : listaDeRevisiones)
        {
```

```
            if(revision.getEstadoRevision() == EstadoRevision.COMPLETADA)
             listaDeRevisionesRealizadas.add(revision);
        }

        return listaDeRevisionesRealizadas;
    }
    List<Vehiculo> obtenerListaVehiculosRevisados()
    {
        ArrayList<Vehiculo> listaDeVehiculosRevisados = new ArrayList<Vehiculo>();
        for(Revision revision : listaDeRevisiones)
        {
            if(revision.getEstadoRevision() == EstadoRevision.COMPLETADA)
             listaDeVehiculosRevisados.add(revision.getVehiculo());
        }

        return listaDeVehiculosRevisados;
    }
    List<Revision> obtenerListaRevisiones()
    {
        return listaDeRevisiones;
    }
    List<Mecanico> getListaMecanicos()
    {
        return listaDeMecanicos;
    }

}
import java.util.*;
/**
 * Write a description of class Revision here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class Revision
{
    // instance variables - replace the example below with your own
    private Mecanico mecanico;
    private Vehiculo vehiculo;
```

```
private Cliente cliente;
private List<TareaDeMantenimiento> tareaMantenimiento;
private EstadoRevision estadoRevision;

/**
 * Constructor for objects of class Revision
 */
public Revision(Vehiculo vehiculo, Cliente cliente)
{
    this.vehiculo = vehiculo;
    this.cliente = cliente;
    this.tareaMantenimiento = new ArrayList<TareaDeMantenimiento>();
}

public Vehiculo getVehiculo()
{
    return vehiculo;
}
public Cliente getCliente()
{
    return cliente;
}

public EstadoRevision getEstadoRevision()
{
    return estadoRevision;
}
public void setEstadoRevision(EstadoRevision estadoRevision)
{
    this.estadoRevision = estadoRevision;
}
public void setMecanicoAsignado(Mecanico mecanico)
{
    this.mecanico = mecanico;
}
public Mecanico getMecanicoAsignado()
{
    return mecanico;
}
```

```
}

/**
 * Enumeration class EstadoRevision - Estados por los que pasa la revisión
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public enum EstadoRevision
{
    PENDIENTE, COMPLETADA, PAUSADA
}

/**
 * Write a description of class TareasDeMantenimento here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class TareaDeMantenimiento
{
   private String description = "";
    public TareaDeMantenimiento(String descripcion)
    {
        this.description = description;
    }

    public String getDescription()
    {
        return description;
    }

}
/**
 * Enumeration class TipoDeVehiculo - write a description of the enum class here
 *
 * @author Ricardo García
 * @version 15/05/2017
```

```
 */
public enum TipoDeVehiculo
{
    COCHE, MOTO, COCHESP, MOTOSP
}
/**
 * Write a description of class Mecanico here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class Mecanico extends Usuario
{
    /**
     * Constructor for objects of class Mecanico
     */
    public Mecanico(String nombre, String telefono)
    {
        super(nombre, telefono);
    }

}


/**
 * Write a description of class JefeDeTaller here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class JefeDeTaller extends Mecanico
{

    /**
     * Constructor for objects of class JefeDeTaller
     */
    public JefeDeTaller(String nombre, String telefono)
    {
        super(nombre, telefono);
    }
```

```
}
import java.util.*;
/**
 * Write a description of class Cliente here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class Cliente extends Usuario
{
    // instance variables - replace the example below with your own
   private List<Vehiculo> listaDeVehiculos;
   private String nif;

    /**
     * Constructor for objects of class Cliente
     */
    public Cliente(String nombre, String telefono, String nif)
    {
       super(nombre, telefono);
        // initialise instance variables
       listaDeVehiculos = new ArrayList<Vehiculo>();
       this.nif = nif;
    }

    /**
     * Agrega un vehículo al usuario
     */
    public void agregarVehiculo(Vehiculo vehiculo)
    {
        listaDeVehiculos.add(vehiculo);
    }

    /**
     * Devuelve el NIF
     */
    public String getNIF()
```

```
    {
        return nif;
    }

    /**
     * Devuelve la lista de vehículos
     */
    public List<Vehiculo> getListaVehiculos()
    {
        return listaDeVehiculos;
    }
}
import java.util.*;



/**
 * Abstract class Vehiculo - write a description of the class here
 *
 * @author:
 * Date:
 */
public abstract class Vehiculo
{
    protected int numeroDeRuedas;
    protected int numeroDePuertas;
    protected boolean esVehiculoDeServicioPublico;
    protected TipoCombustible tipoDeCombustible;
    private int kilometros;
    private Date fechaDeFabricacion;
    private TipoDeVehiculo tipoDeVehiculo;
    private String modelo;
    private String matricula;
    private List<TareaDeMantenimiento> listaDeTareasDeMantenimiento;

    public Vehiculo(String matricula, String modelo, TipoCombustible tipoCombustible)
    {
        listaDeTareasDeMantenimiento = new ArrayList<TareaDeMantenimiento>();
        this.matricula = matricula;
        this.modelo = modelo;
```

```
    this.tipoDeCombustible = tipoCombustible;
}
public String getMatricula()
{
    return matricula;
}
public String getModelo()
{
    return modelo;
}

public int getNumeroDeRuedas() {
    return numeroDeRuedas;
}
public int getNumeroDePuertas()
{
    return numeroDePuertas;
}
public boolean getEsVehiculoDeServicioPublico() {
    return esVehiculoDeServicioPublico;
}
public TipoDeVehiculo getTipoDeVehiculo()
{
    return tipoDeVehiculo;
}

public TipoCombustible getTipoCombustible()
{
    return tipoDeCombustible;
}
public void setFechaDeFabricacion(Date value)
{
    fechaDeFabricacion = value;
}
public Date getFechaFabricacion()
{
    return fechaDeFabricacion;
}
```

```
    public void setKilometros(int value)
    {
        kilometros = value;
    }

    public int getKilometros()
    {
        return kilometros;
    }

    public List<TareaDeMantenimiento> getTareasDeMantenimiento()
    {
        return listaDeTareasDeMantenimiento;
    }
}
/**
 * Abstract class Usuario - write a description of the class here
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public abstract class Usuario
{
   private String nombre;
   private String telefono;

   public Usuario(String nombre, String telefono)
   {
       this.nombre = nombre;
       this.telefono = telefono;
   }

   public String getNombre()
   {
       return nombre;
   }
   public String getTelefono()
   {
       return telefono;
```

```
   }
}

/**
 * Enumeration class TipoCombustible - write a description of the enum class here
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public enum TipoCombustible
{
    GASOLINA, DIESEL, ELECTRICO
}
/**
 * Write a description of class Comercial here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class Comercial extends Usuario
{

    /**
     * Constructor for objects of class Comercial
     */
    public Comercial(String nombre, String telefono)
    {
        super(nombre, telefono);
    }

}

/**
 * Write a description of class Moto here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class Moto extends Vehiculo
```

```
{

    /**
     * Constructor for objects of class Moto
     */
    public Moto(String matricula, String modelo, TipoCombustible tipoCombustible)
    {
        super(matricula, modelo, tipoCombustible);
        this.numeroDeRuedas = 2;
        this.numeroDePuertas = 0;
    }

}
/**
 * Write a description of class Coche here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class Coche extends Vehiculo
{

    /**
     * Constructor for objects of class Coche
     */
    public Coche(String matricula, String modelo, TipoCombustible tipoCombustible)
    {
        super(matricula, modelo, tipoCombustible);
        this.numeroDeRuedas = 4;
        this.numeroDePuertas = 4;
    }



}
/**
 * Write a description of class MotoDeServicioPublico here.
 *
 * @author Ricardo García
 * @version 15/05/2017
```

```
 */
public class MotoDeServicioPublico extends Moto
{

    /**
     * Constructor for objects of class MotoDeServicioPublico
     */
    public MotoDeServicioPublico(String matricula, String modelo, TipoCombustible
tipoCombustible)
    {
        super(matricula, modelo, tipoCombustible);
        this.esVehiculoDeServicioPublico = true;
    }



}
/**
 * Write a description of class CocheDeServicioPublico here.
 *
 * @author Ricardo García
 * @version 15/05/2017
 */
public class CocheDeServicioPublico extends Coche
{

    /**
     * Constructor for objects of class CocheDeServicioPublico
     */
    public CocheDeServicioPublico(String matricula, String modelo, TipoCombustible
tipoCombustible)
    {
        super(matricula, modelo, tipoCombustible);
    }


}
import java.util.*;
/**
 * Write a description of class ConsoleUI here.
 *
 * @author Ricardo García
```

```
 * @version 15/05/2017
 */
public class ConsoleUI
{

    private taller miTaller;
    public ConsoleUI(taller miTaller)
    {
     this.miTaller = miTaller;
    }

    private void clearScreen() {
        System.out.print ('\f');
        System.out.flush();
    }

    /**
     * Muestra el menú principal de la aplicación.
     */
    public void mainMenu()
    {
        int teclaPulsada=0;
        Scanner sc = new Scanner(System.in);
        do
        {
            clearScreen();
            System.out.print("Taller de coches.\n");
            System.out.print("Pulse una opción:\n");
            System.out.print("1 - Entrar como recepcionista\n");
            System.out.print("2 - Entrar como mecánico.\n");
            System.out.print("3 - Entrar como comercial.\n");
            System.out.print("4 - Entrar como Jefe de taller.\n");
            System.out.print("5 - Salir.\n");
            teclaPulsada = sc.nextInt();

            switch(teclaPulsada)
            {
                case 1:
                menuRecepcionista();
```

```
            break;
            case 2:
            menuMecanico();
            break;
            case 3:
            menuComercial();
            break;
            case 4:
            menuJefeTaller();
            break;
        }

    }
    while(teclaPulsada!=5);

}

/**
 * Muestra el menu del recepcionista.
 */
private void menuRecepcionista()
{
    int teclaPulsada=0;
    do
    {
        clearScreen();
        Scanner sc = new Scanner(System.in);
        System.out.print("Taller de coches: menú recepcionista.\n");
        System.out.print("Pulse una opción:\n");
        System.out.print("1 - Dar de alta cliente.\n");
        System.out.print("2 - Dar de alta nueva ficha de reparación.\n");
        System.out.print("3 - Dar de alta nuevo vehículo.\n");
        System.out.print("4 - Salir.\n");
        teclaPulsada = sc.nextInt();
        switch(teclaPulsada)
        {
            case 1:
            darDeAltaCliente();
            break;
```

```
            case 2:
            darDeAltaFichaDeReparacion();
            break;
            case 3:
            darDeAltaVehiculo();
            break;
        }
    }
    while(teclaPulsada!=4);
}

/**
 * Da de alta un nuevo usuario con un vehículo
 */
private void darDeAltaCliente()
{
    Cliente cliente = crearCliente();
    miTaller.darDeAltaCliente(cliente);
    darDeAltaVehiculo(cliente);

}

/**
 * Crea un nuevo objeto usuario.
 */
private Cliente crearCliente()
{
    String  nombreCliente = "";
    String telefono = "";
    String nif = "";
    clearScreen();
    Scanner sc = new Scanner(System.in);
    System.out.print("Dar de alta cliente.\n");
    System.out.print("Nombre del cliente:\n");
    nombreCliente = sc.nextLine();
    System.out.print("Nif del cliente:\n");
    nif = sc.nextLine();
    System.out.print("Teléfono del cliente:\n");
    telefono = sc.nextLine();
```

```
    return new Cliente(nombreCliente, telefono, nif);
}

/**
 * Da de alta una nueva ficha de reparación
 */
private void darDeAltaFichaDeReparacion()
{
    Cliente cliente = seleccionarCliente();
    Vehiculo vehiculo = seleccionarVehiculoCliente(cliente);
    miTaller.darDeAltaRevision(vehiculo, cliente);
}

/**
 * Selecciona un vehículo de entre los que tiene el cliente
 */
private Vehiculo seleccionarVehiculoCliente(Cliente cliente)
{
    List<Vehiculo> listaDeVehiculos = cliente.getListaVehiculos();

    if(listaDeVehiculos.size() == 1)
        return listaDeVehiculos.get(0);

    int teclaPulsada=0;
    Scanner sc = new Scanner(System.in);
    do
    {
        clearScreen();
        System.out.print("Seleccione un vehiculo:\n");
        for(int n=1; n<=listaDeVehiculos.size() ;n++)
        {
            Vehiculo vehiculoActual = listaDeVehiculos.get(n-1);
            System.out.printf("%d - %s", vehiculoActual.getModelo());
        }
        try
        {
        teclaPulsada = sc.nextInt();
        }
        catch(InputMismatchException ex)
```

```
        {
            teclaPulsada=0;
        }
    }
    while(teclaPulsada <1 || teclaPulsada > listaDeVehiculos.size());
    return listaDeVehiculos.get(teclaPulsada - 1);

}

/**
 * Crea un objeto de tipo vehículo
 */
private Vehiculo crearVehiculo()
{
    clearScreen();
    String matricula = "";
    String modelo = "";
    Vehiculo vehiculo = null;
    TipoDeVehiculo tipoDeVehiculo = null;
    TipoCombustible tipoCombustible = null;
    int teclaPulsada=0;
    Scanner sc = new Scanner(System.in);

    System.out.print("Dar de alta vehículo.\n");
    System.out.print("Matrícula:\n");
    matricula = sc.nextLine();

    System.out.print("Modelo:\n");
    matricula = sc.nextLine();

    System.out.print("Seleccione tipo de vehículo:\n");
    System.out.print("1 - Coche.\n");
    System.out.print("2 - Moto.\n");
    System.out.print("3 - Coche de servicio público.\n");
    System.out.print("4 - Moto de servicio público.\n");
    teclaPulsada = sc.nextInt();

    switch(teclaPulsada)
    {
```

```
        case 1:
        vehiculo = new Coche(matricula, modelo, tipoCombustible);
        break;
        case 2:
        vehiculo = new Moto(matricula, modelo, tipoCombustible);
        break;
        case 3:
        vehiculo = new CocheDeServicioPublico(matricula, modelo, tipoCombustible);
        break;
        case 4:
        vehiculo = new MotoDeServicioPublico(matricula, modelo, tipoCombustible);
        break;
    }

    while(vehiculo == null);
    return vehiculo;
}

/**
 * Da de alta un nuevo vehiculo asociado a un usuario
 */
private void darDeAltaVehiculo()
{
     darDeAltaVehiculo(seleccionarCliente());
}

/**
 * Da de alta un nuevo vehículo
 */
private void darDeAltaVehiculo(Cliente cliente)
{
    miTaller.darDeAltaVehiculo( crearVehiculo(), cliente);
}

/**
 * Muestra el menu del mecánico
 */
private void menuMecanico()
{
```

```
    int teclaPulsada=0;
    do
    {
        clearScreen();
        Scanner sc = new Scanner(System.in);
        System.out.print("Taller de coches: menú mecánico.\n");
        System.out.print("Pulse una opción:\n");
        System.out.print("1 - Ver fichas pendientes.\n");
        System.out.print("2 - Gestionar ficha.\n");
        System.out.print("3 - Salir.\n");
        try
        {
            teclaPulsada = sc.nextInt();
        }
        catch(InputMismatchException e)
        {
            teclaPulsada = 0;
        }
        switch(teclaPulsada)
        {
            case 1:
            verFichas();
            break;
            case 2:
            gestionarFicha();
            break;
        }

    }
    while(teclaPulsada!='3');
}

/**
 * Muestra la lista de fichas pendientes de un mecánico concreto
 */
private void verFichas()
{
    List<Mecanico> listaDeMecanicos = miTaller.getListaMecanicos();
    int teclaPulsada=0;
```

```
        Scanner sc = new Scanner(System.in);
        do
        {
            clearScreen();
            System.out.print("Seleccione un vehiculo:\n");
            for(int n=1; n<=listaDeMecanicos.size();n++)
            {
                 Mecanico mecanicoActual = listaDeMecanicos.get(n-1);
                 System.out.printf("%d - %s", n, mecanicoActual.getNombre());
            }
            try
            {
            teclaPulsada = sc.nextInt();
            }
            catch(InputMismatchException ex)
            {
                teclaPulsada=0;
            }
        }
        while(teclaPulsada < 1 | teclaPulsada > listaDeMecanicos.size());
        List<Revision> listaRevisionesPendientes =
miTaller.obtenerListaRevisionesPendientes(listaDeMecanicos.get(teclaPulsada-1));
        for(Revision revision : listaRevisionesPendientes)
        {
            System.out.printf("%s - Pendiente\n", revision.getVehiculo().getMatricula());
        }
    }

    /**
     * Permite gestionar las fichas de reparación por parte de los mecánicos.
     */
    private void gestionarFicha()
    {
        List<Mecanico> listaDeMecanicos = miTaller.getListaMecanicos();
        int teclaPulsada=0;
        Scanner sc = new Scanner(System.in);
        do
        {
            clearScreen();
```

```
            System.out.print("Seleccione un vehiculo:\n");
            for(int n=1; n<=listaDeMecanicos.size();n++)
            {
                 Mecanico mecanicoActual = listaDeMecanicos.get(n-1);
                 System.out.printf("%d - %s", n, mecanicoActual.getNombre());
            }
            try
            {
            teclaPulsada = sc.nextInt();
            }
            catch(InputMismatchException ex)
            {
                teclaPulsada=0;
            }
        }
        while(teclaPulsada < 1 | teclaPulsada > listaDeMecanicos.size());
        List<Revision> listaRevisionesPendientes =
miTaller.obtenerListaRevisionesPendientes(listaDeMecanicos.get(teclaPulsada-1));
        teclaPulsada=0;
        do
        {
            clearScreen();
            System.out.print("Seleccione una ficha de reparación:\n");
            for(int n=1; n<=listaRevisionesPendientes.size();n++)
            {
                Revision revisionActual = listaRevisionesPendientes.get(n-1);
                System.out.printf("%d - %s\n", revisionActual.getVehiculo().getMatricula());
            }
            try
            {
            teclaPulsada = sc.nextInt();
            }
            catch(InputMismatchException ex)
            {
                teclaPulsada=0;
            }
        }
        while(teclaPulsada < 1 | teclaPulsada > listaRevisionesPendientes.size());
        Revision fichaDeReparacionSeleccionada = listaRevisionesPendientes.get(teclaPulsada -
1);
```

```
    teclaPulsada=0;
    do
    {
        clearScreen();
        System.out.print("Seleccione un estado:\n");
        System.out.printf("1 - Pausado\n");
        System.out.printf("2 - Completado\n");
        System.out.printf("3 - Pendiente\n");
        System.out.printf("4 - Salir\n");
        try
        {
        teclaPulsada = sc.nextInt();
        }
        catch(InputMismatchException ex)
        {
            teclaPulsada=0;
        }
    }
    while(teclaPulsada < 1 | teclaPulsada > 4);
    switch(teclaPulsada)
    {
        case 1:
            fichaDeReparacionSeleccionada.setEstadoRevision(EstadoRevision.PAUSADA);
            break;
        case 2:
            fichaDeReparacionSeleccionada.setEstadoRevision(EstadoRevision.COMPLETADA);
            break;
        case 3:
            fichaDeReparacionSeleccionada.setEstadoRevision(EstadoRevision.PENDIENTE);
            break;

    }
}

/**
 * Muestra el menú del comercial
 */
private void menuComercial()
{
```

```
    int teclaPulsada=0;
    Scanner sc = new Scanner(System.in);
    do
    {
        clearScreen();
        System.out.print("Taller de coches: menú comercial.\n");
        System.out.print("Pulse una opción:\n");
        System.out.print("1 - Gestionar ofertas.\n");
        System.out.print("2 - Gestionar clientes.\n");
        System.out.print("3 - Gestionar ITVs.\n");
        System.out.print("4 - Salir.\n");
        try
        {
            teclaPulsada = sc.nextInt();
        }
        catch(InputMismatchException e)
        {
            teclaPulsada = 0;
        }
        switch(teclaPulsada)
        {
            case 1:
            menuRecepcionista();
            break;
            case 2:
            menuMecanico();
            break;
            case 3:
            menuComercial();
            break;
            case 4:
            menuJefeTaller();
            break;
        }

    }
    while(teclaPulsada!=5);
}
```

```
/**
 * Muestra el menú del jefe de taller.
 *
 */
private void menuJefeTaller()
{
    int teclaPulsada=0;
    Scanner sc = new Scanner(System.in);
    do
    {
        clearScreen();
        System.out.print("Taller de coches: menu jefe de taller.\n");
        System.out.print("Pulse una opción:\n");
        System.out.print("1 - Mostar fichas pendientes.\n");
        System.out.print("2 - Asignar fichas.\n");
        System.out.print("3 - Ver fichas proceadas.\n");
        System.out.print("4 - Ver vehículos reparados.\n");
        System.out.print("5 - Salir.\n");
        try
        {
            teclaPulsada = sc.nextInt();
        }
        catch(InputMismatchException e)
        {
            teclaPulsada = 0;
        }
        switch(teclaPulsada)
        {
            case 1:
            mostrarFichasPendientes();
            break;
            case 2:
            asignarFichas();
            break;
            case 3:
            verFichasProcesadas();
            break;
            case 4:
            verVehiculosReparados();
```

```
                break;
            }

        }
        while(teclaPulsada!=5);
    }

    /**
     * Muestra una lista de fichas de reparación pendientes.
     */
    private void mostrarFichasPendientes()
    {
       clearScreen();
       System.out.printf("Fichas pendientes:");
       List<Revision> listaDeRevisiones= miTaller.obtenerListaRevisionesPendientes();
       for(Revision revision : listaDeRevisiones)
        {
             System.out.printf("%s - Pendiente\n", revision.getVehiculo().getMatricula());
        }
    }

    /**
     * Muestra un menú para poder asignar una ficha de reparación a un mecánico.
     */
    private void asignarFichas()
    {
       int teclaPulsada=0;
       List<Revision> listaDeRevisiones= miTaller.obtenerListaRevisionesPendientes();
       Scanner sc = new Scanner(System.in);
        do
        {
            clearScreen();
            System.out.printf("Seleccione la ficha que quiere asignar:");

            for(int n=1; n <= listaDeRevisiones.size(); n++)
            {
                System.out.printf("%d - %s\n", n, listaDeRevisiones.get(n-
1).getVehiculo().getMatricula());
            }
```

```
        try
        {
            teclaPulsada = sc.nextInt();
        }
        catch(InputMismatchException ex)
        {
            teclaPulsada=0;
        }
    }
    while(teclaPulsada < 1 || teclaPulsada > listaDeRevisiones.size());

    //Falta seleccionar el mecánico
}

/**
 * Muestra una lista con las fichas de reparacíon procesadas.
 */
private void verFichasProcesadas()
{
   clearScreen();
   System.out.printf("Fichas pendientes:");
   List<Revision> listaDeRevisiones= miTaller.obtenerListaRevisionesPendientes();
   for(Revision revision : listaDeRevisiones)
    {
         System.out.printf("%s - Pendiente\n", revision.getVehiculo().getMatricula());
    }
}

/**
 * Muestra una lista con los vehiculos reparados.
 */
private void verVehiculosReparados()
{
   clearScreen();
   System.out.printf("Fichas pendientes:");
   List<Vehiculo> listaDeVehiculosRevisados= miTaller.obtenerListaVehiculosRevisados();
   for(Vehiculo vehiculo : listaDeVehiculosRevisados)
    {
         System.out.printf("%s - Revisadon\n", vehiculo.getMatricula());
```

```
        }
    }

    /**
     * Muestra un menú para seleccionar el cliente.
    */
    private Cliente seleccionarCliente()
    {
        List<Cliente> listaDeClientes = miTaller.obtenerListaClientes();

        int teclaPulsada=0;
        Scanner sc = new Scanner(System.in);
        do
        {
            clearScreen();
            System.out.print("Seleccione un cliente:\n");
            for(int n=1; n<=listaDeClientes.size() ;n++)
            {
                Cliente clienteActual = listaDeClientes.get(n-1);
                System.out.printf("%d - %s", n, clienteActual.getNombre());
            }
            try
            {
            teclaPulsada = sc.nextInt();
            }
            catch(InputMismatchException ex)
            {
                teclaPulsada=0;
            }
        }
        while(teclaPulsada < 1 || teclaPulsada > listaDeClientes.size());
        return listaDeClientes.get(teclaPulsada - 1);
    }

}
```