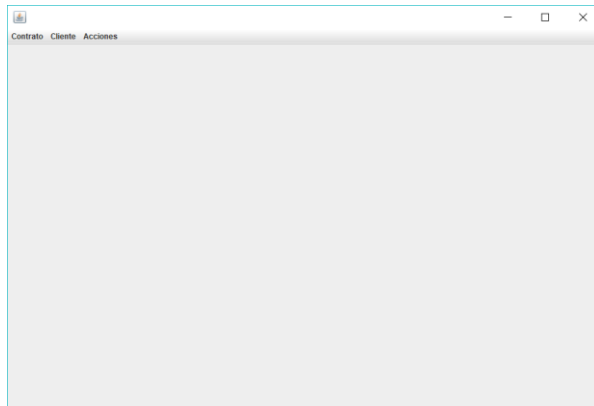


La historia

Tenemos que realizar una parte de una aplicación para una tienda de móviles. En concreto vamos a realizar dos “Acciones” que puede utilizar el administrador de la tienda.



En la primera tenemos un formulario en la que el usuario puede introducir el número de minutos y el de megas mensuales que cree necesitar.

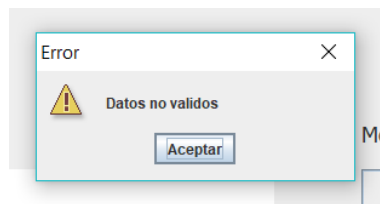
A screenshot of the 'SIMULADOR CONTRATO' form within the application window. The form has a title 'SIMULADOR CONTRATO'. It contains two input fields: 'Minutos' and 'Megas'. Below these is a large text area labeled 'Características'. To the right of the text area is a dropdown menu labeled 'Móviles' and a text input field labeled 'Contacto'. At the bottom right are two buttons: 'Validar' and 'Pedir'.

Si pulsamos el botón “Validar” (mientras no se valide el botón de pedir está velado) nos dará la tarifa más adecuada a los valores dados (su primer requisito es que se adecúe al número de minutos y luego al de megas cumpliendo ambos siempre por exceso)

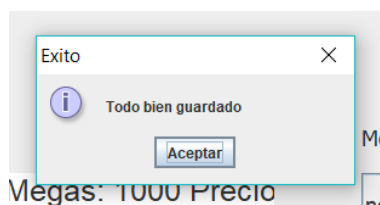
A screenshot of the 'SIMULADOR CONTRATO' form with data entered. The 'Minutos' field contains '11' and the 'Megas' field contains '11'. The 'Características' text area now displays 'Tarifa asilo Minutos: 200 Megas: 1000 Precio'. The 'Móviles' dropdown menu is set to 'no solicita'. The 'Contacto' field is empty. The 'Validar' and 'Pedir' buttons are still visible at the bottom right.

El botón validar también rellena un combo box con los móviles que se ofertan para esa tarifa. Podemos seleccionar uno de ellos o el valor “no solicita”.

Para que la simulación sea correcta se debe introducir un número de teléfono de contacto. Dicho número debe tener nueve dígitos numéricos, siendo el primero un “9”, un “7” o un “6”. Cualquier valor erróneo en alguno de los campos, tanto al validar como al pedir debe mostrar un mensaje:

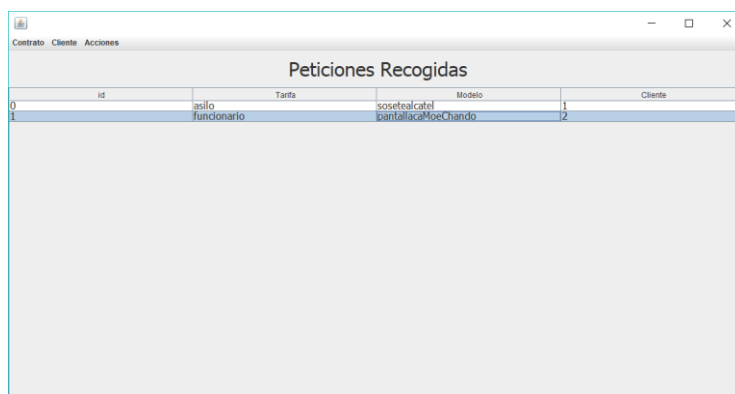


Si al pulsar el botón de pedir los datos son correctos se almacena una petición con ellos y se muestra el mensaje:



Una vez aceptado el mensaje se borran todos los campos del formulario.

La otra acción consiste en mostrar los valores de las diferentes peticiones realizadas en una Tabla que tiene la siguiente forma:



Peticiones Recogidas				
id	Tarifa	Modelo	Cliente	
0	asilo	sosetrealcatel	1	
1	funcionario	canillacalfoeChando	12	

Los campos reflejados consisten en el número de identificación de la petición (un valor único), el nombre de la tarifa propuesta y el posible modelo asociado a la misma seleccionado. Finalmente en el campo cliente se escribe el número de contacto previamente escrito.

Esta tabla no tiene previsto la interacción con ella.

La primera es la opción “Simulaciones” del menú “Acciones”. Esta segunda es la opción “Peticiones” del menú “Acciones”.

La estructura del proyecto

El proyecto se presenta incompleto. Se divide en una serie de paquetes: acción, eventos, lógica, acceso, data, utiles, test y vista.

En cada uno de ellos verás clases que deben estar y que deben ser completadas. Veamos cada una de ellas

Modelo

Donde encontramos los tipos de datos a usar. Tarifa, Peticion y Movil. También encontrarás la clase Datos que puede usarse como fachada para la lógica.

Modelo.acceso

Donde además del conocido DAO para ficheros de objetos se encuentran dos Bridges conocidos: AlmacenIndice que permite el almacenamiento y obtención de objetos grabados secuencialmente a través de un atributo clave y AlmacenIndividual que se comporta como un adaptador del DAO para almacenar un único elemento (que puede ser de cualquier tipo).

Todas las clases de estos paquetes mencionados funcionan, salvo catástrofe, y NO, o sea NO, NIEN, NIET pueden ser modificados (Excepto alguna cosilla).

Lógica

En esta parte se define la funcionalidad del programa mediante la declaración de algunos métodos que deberán ser implementados. Se te permite la definición de tantos métodos privados como quieras pero no la de más métodos públicos.

La lógica se acompaña de un test para que pruebes si funciona.

Accion

Aquí encontramos dos Controladores. SimuladorController que se encarga de las acciones a llevar a cabo en el panel de simulación y PeticionController que se encarga del panel de peticiones. Ambos tienen pocos métodos y no puede ser modificada su declaración (observa que el bridge se pasa como parámetro). De nuevo todos los privados que queráis pero no públicos

Eventos

Sitio preparado para el bridge. El uso de los validadores se puede hacer en sitio que creas más adecuado. Observa que se ha definido la propiedad Logica en el Bridge.

Vista

Para las clases del UI

Lo que se pide:

La fachada de datos:

La clase Datos debe definir los almacenes necesarios para satisfacer las necesidades de información.

Debes usar la clase CreadorElementos para crear el fichero de Tarifas. Las peticiones deben almacenarse con un AlmacenIndice. El número de petición se debe obtener de un fichero igualmente. Puedes crear los métodos que creas necesarios.

La clase AlmacenIndice.

Es la conocida por todos. Solo permite la obtención y grabación de elementos. Se pide la implementación del método

```
public ArrayList<T> getCollection()
```

Que nos entrega una lista de todos los elementos almacenados en el almacenIndice (en nuestro caso peticiones). Para la implementación de este método NO,NIEN pueden usarse, directamente, métodos de la clase DAO.

La clase Logica

Cuenta con tres métodos públicos que se explican en la propia clase. La declaración de estos métodos no puede modificarse. Hay que implementarlos. Todo uso de datos debe ser a través de la fachada Datos. No se pueden añadir más métodos públicos pero si privados

La mejor tarifa es aquella que se acerque lo más posible, siempre por exceso, a los criterios de minutos y megas. Teniendo en cuenta que su principal criterio son los megas. Es decir si tiene que elegir entre dos tarifas elegirá aquella que se acerque más al número de megas precisados.

```
public Tarifa obtenerMejor(int minutos, int megas)

public String obtenerCondiciones(Tarifa tarifa)

public ArrayList<Movil> obtener(Tarifa tarifa)

public boolean guardaPetición(Tarifa tarifa, int selectedItem,
String contacto)

public Collection<Petición> obtenerTodas()
```

Las clases de acción

SimuladorController

Satisface la acción asociada a los dos botones del panel simulador. Validar y Pedir. En el primero es necesario que en los campos minutos y megas del formulario haya valores válidos. Utiliza y/o modifica los validadores proporcionados. Tanto los minutos como los megas deben ser un número entero positivo entre 0 y 31999. Con estos datos se debe obtener la tarifa más adecuada, calculada en el método de la clase Logica correspondiente.

Si se valida se deben mostrar: las condiciones de la tarifa y rellenar el combo con los móviles asociados a ella, también se habilita el botón pedir y se prohíbe la edición de los campos minutos y megas.

El botón pedir comprueba que el campo “contacto” es un número valido de teléfono. Usa validadores. Si todo es correcto debe almacenar la petición, debe limpiar todos los elementos del panel, habilitar para edición los campos megas y minutos y deshabilitar el botón pedir.

Tanto para una cosa como para la otra pueden darse circunstancias de fallo o éxito. En ambas debes usar la clase Dialogo para mostrar una ventana que nos comunique el resultado de las operaciones:

Para el botón validar. Un Dialogo si existe un problema en las validaciones (genérico).

Para el botón pedir. Un dialogo si existe un problema en la validación y otro si la petición se ha almacenado correctamente.

PeticionController

Es un controlador asociado a la carga del panel de peticiones. Lo que debe hacer es cargar la rejilla con todas las peticiones almacenadas.

Eventos

Solo cuenta con la clase bridge que usa los controllers anteriores.

Vista

Debes crear una ventana con los menús propuestos. Debes asociar un panel a cada opción del menú acciones. En concreto un panel para el simulador y otro para las peticiones.

Test

No tienes porque hacer test si no quieres. Te proporcionamos uno para probar el método de la lógica propuesto.

Como se puntúa

Fachada

1 Peticiones 1 punto

2 Tarifas 0,5

Logica

3 Obtener mejor tarifa 1

4 Obtener condiciones 0,5

5 Obtener los móviles de una tarifa 0.5

6 Guardar la petición 1

7 Obtener las peticiones almacenadas 0.5

Simulador Controller

8 Uso de validadores 0.5

9 Gestionar los componentes del panel correctamente 1

Peticion controller

10 Usar jdialog 0.5

11 Carga de la rejilla 1

Bridge

12 Uso correcto de la estructura Datos-Logica-Acciones-Eventos 1

Como hacer el examen

Se puede dividir el examen en dos caminos:

Primer camino

Realizar los elementos necesarios para la simulación (a excepción de guardar petición) Que sería la correcta realización de los puntos

2+3+4+5+8+9+12

Que curiosamente suman cinco.

Segundo camino

1+6+7+10+11+12

Que suman seis (el doce solo cuenta una vez)

Yo en vuestro caso intentaría el primer camino que es el más sencillo. Y contaría con que el UI bien hecho (que funcione) me ayudaría por si me falta algún punto.

Una vez hecho esto me centraría en el segundo camino.

Algunas cosas son sencillas, tales como el uso de validadores para el simulador o de diálogos para las peticiones. Sin embargo el uso de ambas herramientas en ambos casos será tenido en cuenta para la puntuación. Otras son más difíciles, como conseguir la mejor tarifa. Si esta no os sale podéis usar un elemento mock, es decir, que el método que obtiene la mejor tarifa posible siempre entregue la misma tarifa, sin hacer nada más. Esto no puntúa pero os permite seguir con el examen. En general podéis hacer esto siempre que lo necesitéis. Mucha suerte