Ejercicio G

- ♥ No cumple con el protocolo MVC ya que no hay de conexión entre el modelo y la vista la cual lo solucione creando una lista de listeners en el modelo
- ♥ Luego, el main me pareció mas adecuado que este en el controlador en vez de que este con la vista, ya que el main no muestra nada y en el ejercicio H el cual implementa un buen protocolo de MVC está en el controlador
- ♥ En el main se crea un frame el cual no me parece un buen lugar para posicionarlo, por lo que lo traslade a una nueva clase en la vista llamada “CreatorOfFrame”

```
public class CreatorOfFrame {
    protected static void createFrame(JSkiRentingPriceViewImpl JSkiRentingPriceViewImp) {
        JFrame frame = new JFrame( title: "");
        frame.setContentPane(JSkiRentingPriceViewImp.getContent());
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

- ♥ Por otro lado, en la interfaz JSkiRentingPriceController le agregue el método onEventCalculate(), me pareció adecuado que todos los controladores implementen ese método
- ♥ Luego, en la clase JSkiRentingPriceControllerImpl agregue los métodos calculatePrice() ya que no había ninguna interacción sobre el cambio de estado y start() en donde crea una nueva vista e inicia el controlador
- ♥ Cree un repositorio ya que en el ejercicio H que tiene uno en donde hay métodos donde se manejan con los tickets y obviamente su interfaz
- ♥ En la clase Ticket agregue el método areTheSame() en donde verifica si 2 tickets son iguales
- ♥ Para la clase de JSKiRentingModelImpl agregue los métodos de addListener(), getFinalPrice() y para el método calculatePrice() en vez de recibir un listener y los minutos me pareció más adecuado que reciba solo los minutos. A su vez, también le agregue que le notifica al listener sobre los cambios hechos cosa que debería cumplir el protocolo MVC. También tiene su interfaz correspondiente