Ayudantía 5

Cosmic Team

Ejercicio 1: Afirme o refute las siguientes preguntas, argumentando su respuesta

1. “Los requerimientos de un proyecto solo se pueden expresar tanto en lenguaje formal como en lenguaje natural.”
2. “El objetivo de la Ingeniería de Software es la construcción de software de calidad.”
3. “Las precondiciones no necesariamente tienen que ser comprobadas en los casos de uso.”

SOLUCIÓN EJERCICIO 1

1. *Los requerimientos de un proyecto solo se pueden expresar tanto en lenguaje formal como en lenguaje natural.*

   Se refuta. Además de lo mencionado también se pueden expresar con diagramas.

2. *El objetivo de la Ingeniería de Software es la construcción de software de calidad.*

   Se refuta. Tiene por objetivo la construcción sistemática, eficaz y eficiente de software que sea eficaz y eficiente.

3. *Las precondiciones no necesariamente tienen que ser comprobadas en los casos de uso.*

   Se afirma. No tienen que ser comprobadas porque se consideran como condiciones verdaderas antes del inicio de los casos de uso.

Ejercicio 2: Patrón de Diseño: Transaction Script

Un hotel muy conocido posee un servicio de reserva *online* de habitaciones a través de su página web. Esta “transacción” es realizada desde el cliente, quien debe escoger un número de habitación y una cantidad de días. El sistema internamente comprueba la disponibilidad, calcula el precio total, realiza la reserva y completa la transacción (*commit*).

Ejercicio: escribir el pseudocódigo necesario para que un cliente reserve la habitación n° 42, durante 4 días, utilizando el patrón de diseño Transaction Script.

SOLUCIÓN EJERCICIO 2

```
public class BookRoomTS
{
        comprobarDisponibilidad(Room room){
        if(room.days==0) //disponible
        else //no disponible
        }

        public void bookRoom(int roomN, int days)
        {
        Room room = Room.find(42);
        comprobarDisponibilidad(room);//check availability
        room.setDays(4);
        //calculate price
        //book the room
        //commit transaction
        //print result
        }

        //...
}
```

```
Public class Room
{
        int numRoom;
        int days;
        int price;

        //getters
        //setters
        //Constructor
}
```

Aquí vemos que se cumplen las condiciones para usar el patrón Transaction Script:

- La lógica de negocio está organizada en procedimientos (transacciones)
- Un conjunto de transacciones está agrupado en una clase
- Las funcionalidades del sistema son poco complejas

Ejercicio 3

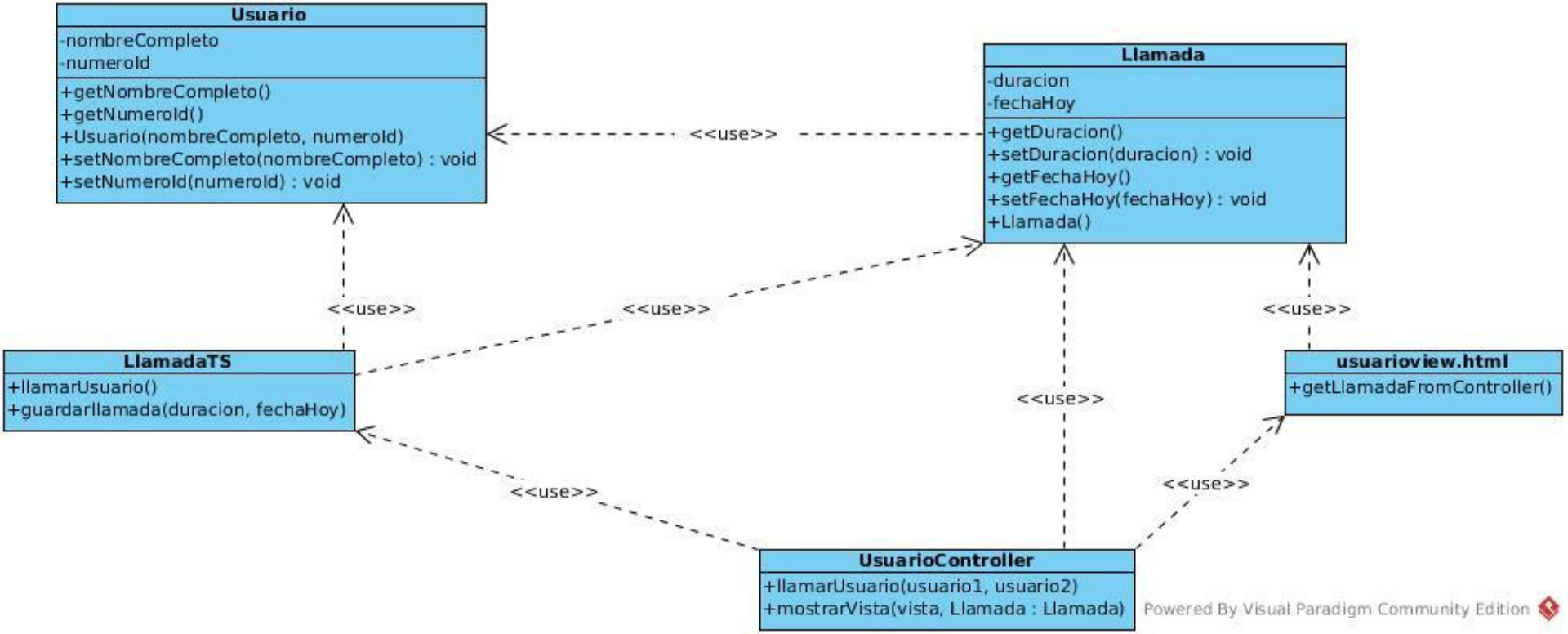
Una equipo de desarrolladores de la empresa “Stack Awesome” están trabajando en un software (app) capaz de hacer llamadas entre smartphones sin el uso de una red de las compañías telefónicas, para esto es necesario implementar la conocido “llamada Voip” (utilizando un servidor que trabajara protocolos SIP). Este equipo no posee conocimientos de redes Voip y sus derivados, por lo que le piden ayuda a los sansanos de la USM para que le implementen este módulo a un precio aceptable para ellos.
Ahora, el equipo pide los sgtes requisitos:

1. Para la llamada, deben existir 2 **Usuarios** que tengan nombre completo y número de identificación (por ej: +56432123)
2. Que el sistema guarde la duración y la fecha de la **Llamada**.
3. Que muestre por pantalla los valores guardados.

Utilizando los subpatrones Template View, Page Controller y Transaction Script fabrique:

1) Diagrama de clases (Cuidado con la notación UML y los nombres de las clases)
2) Escribir el código necesario para implementar la solución que Ud. propone.

Solución Ejercicio 3



Ejercicio 4

Construcción de un sistema MVC (sólo controlador y vista) con PHP

Enunciado: la idea es, utilizando sólo PHP, construir un sistema MVC que considere sólo las separaciones controlador y vista.
Supongamos tenemos un formulario que envía un nombre de usuario y contraseña. El formulario (una página PHP cualquiera), envía los datos vía POST, pero además envía por GET la variable action = 1 que indica que se intenta hacer el login.
Responsabilidades de nuestro controlador (page controller):
-Debe decodificar la URL: eso lo hace con $_GET[‘action’].
-Debe decidir una acción: si el valor de lo decodificado (action) es igual a 1, entonces va a la base de datos y chequea que exista el nombre de usuario junto con su contraseña. (por ahora, no nos preocupamos del modelo).
-Debe decidir qué vista es la encargada de mostrar la información que está en el modelo. Si en el modelo (por ahora, nuestra base de datos). Si en el modelo está el usuario y su contraseña, hace un redirect

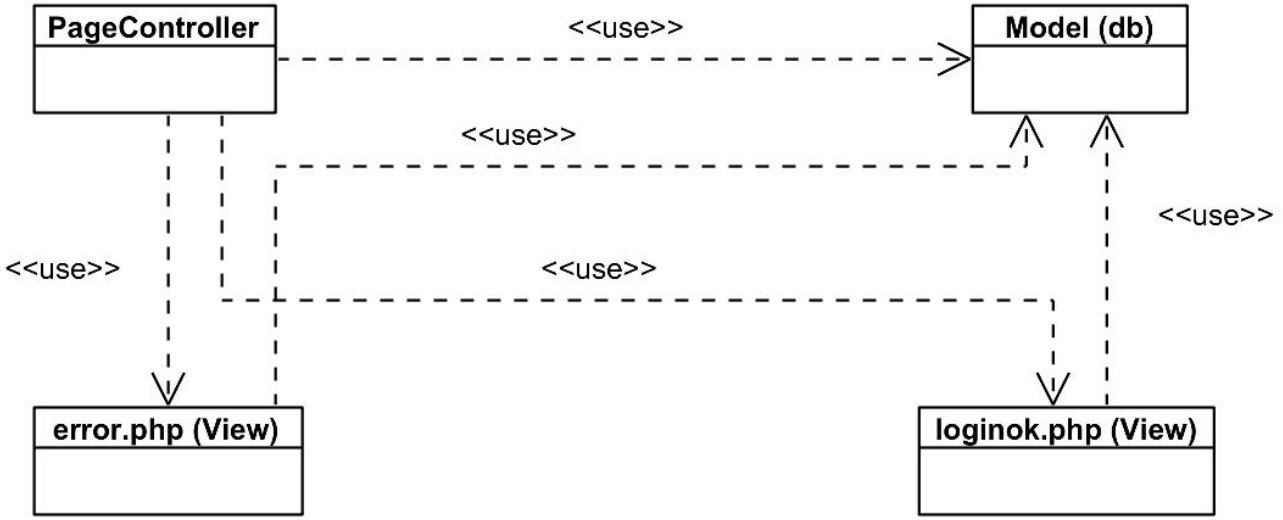
```
If (user ok && password ok)
        Redirect (loginok.php?username=$username)
Else
        Redirect (error.php?username=$username)
```

Ejercicio 4

Responsabilidades de nuestras vistas:
-Vista loginok: mostrar datos relacionados con el login de $username
-Vista error: mostrar mensaje de error para el login de $username
-Ambas vistas utilizan tags html para “decorar” el objeto php $username

1)Construya un diagrama de Clases

Solución Ejercicio 4



en el caso de este ejemplo, las vistas están relacionadas con "uso" hacia el modelo, pero esto es para indicar dependencia, nada más. porque en el ejemplo se hizo sin uso entre las vistas y modelo así que pueden omitir estas relaciones (las de las vistas hacia el modelo). menciono esto por si alguien pregunta por qué no salen las mismas líneas que en las diapos.

Ejercicio 5

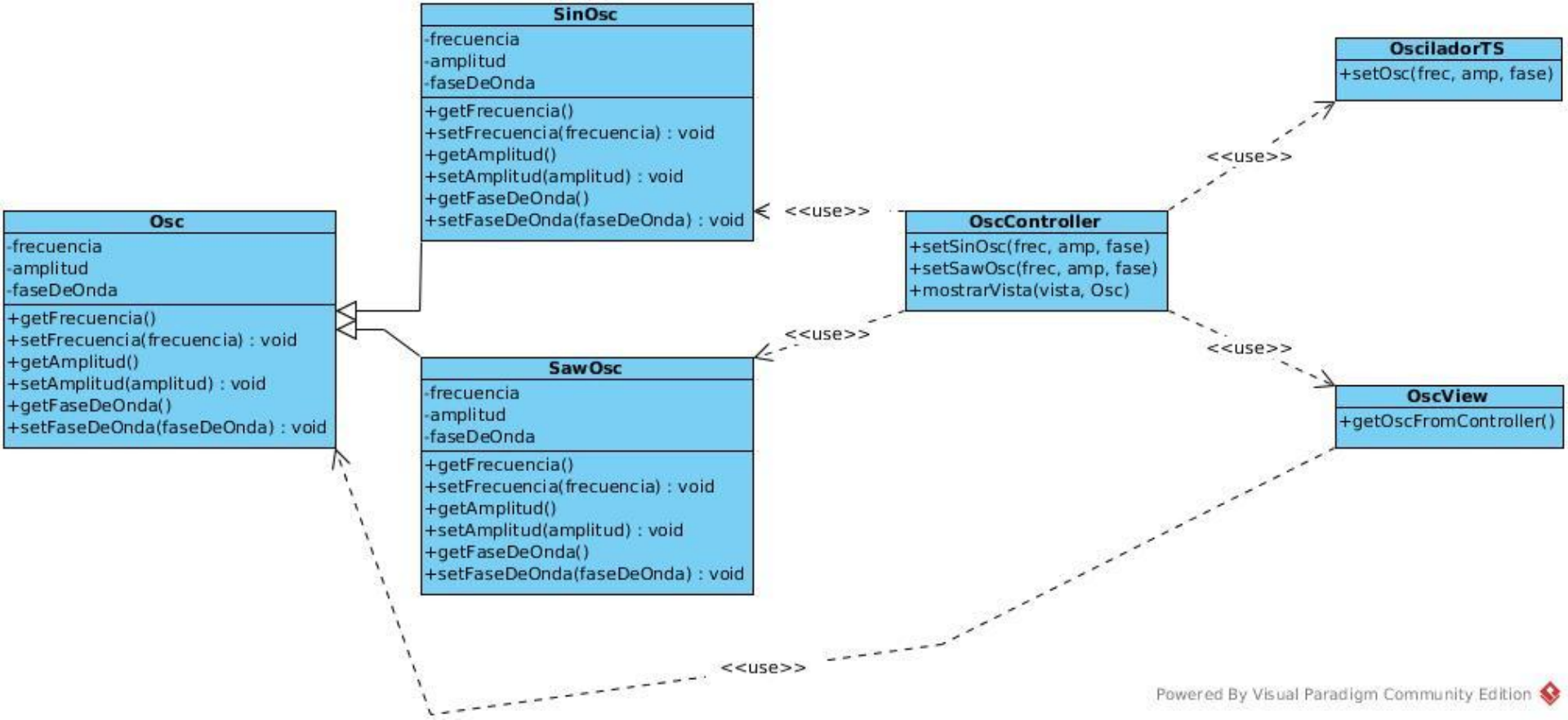
Una conocida empresa que fabrica sintetizadores de sonidos contacta ingenieros de SW de la UTFSM para diseñar SW de sintetizador, el HW del sintetizador es muy sencillo y solo contiene 2 osciladores, 1 de onda senoidales(SinOsc) y otra de ondas dentadas(SawOsc). Cada oscilador tiene 3 controles asociados que setean la frecuencia, amplitud y fase de onda.

Cuando se modifica la frecuencia, amplitud o fase de la onda, se cambian los nuevos valores en un archivo que es tomado e interpretado por el módulo de sonido del sintetizador. Además se muestra por pantalla los nuevos valores para la onda en cuestión después de un análisis inicial. Rápidamente se da cuenta de que el patrón MVC es ideal para este sistema, utilizando los subpatrones Page Controller, Template View y Transaction Script.

1)Construya un modelo de clase para la solución que Ud. propone. (Utilize UML y sea cuidadoso con la notación de las asociaciones)

2)Escriba el código necesario, y más importante, en términos funcionales para implementar esta solución, sea cuidadoso con los nombres de las clases y métodos.

Solución Ejercicio 5



Solución Ejercicio 5

```
public class Oscillator Controller {
   public void handleSinOsc (float freq, float amp, float phase) {
         // controlador crea oscilador y establece parámetros
         SinOsc sinOsc = new SinOsc();
         sinOsc.setFreq(freq);
         sinOsc.setAmp(amp);
         sinOsc.setPhase(phase);

         // llamamos al objeto "service" (Transaction Script)
         // supondremos método estático para ahorrar espacio
         OscillatorService.writeParameters(sinOsc);

         // elegimos vista "SynthView" y le pasamos el objeto sinOsc
         forward("SynthView", sinOsc);

   }

   public void handleSawOsc(float freq, float amp, float phase) {
         // handleSawOsc se construye de manera análoga
         // recordar: elegimos la opción de un controlador para
         // ambos osciladores
   }

}
```

También se acepta como respuesta válida utilizar un controlador para cada oscilador (ej: SinOscController y SawOscController).

La vista (SynthView) recibe el objeto de alguna forma. En este caso supondremos que el *framework* ya tiene un método `getObjectFromController()`. Por tanto, parte del código de la vista sería:

```
// recibir objeto desde controlador
objOscillator = getObjectFromController();

// imprimir en pantalla
print ("Frequency is {0}", objOscillator.getFreq());
print ("Amplitude is {0}", objOscillator.getAmp());
print ("Phase is {0}", objOscillator.getPhase());
```

En este caso, el modelo se diseña utilizando el patrón Transaction Script:

```
public class OscillatorService {
      public void writeParameters(Oscillator osc) {
            open("paramsFile");
            write("paramsFile", freq, osc.getFreq());
            write("paramsFile", amp, osc.getAmp());
            write("paramsFile", phase, osc.getPhase());
            close("paramsFile");
      }
}
```

En el código se ha dado por hecho que el *framework* contiene un método `open()` que abre archivos especificados.

El método `write()` escribe en el archivo especificado los parámetros y valores indicados.

Ejercicio 6

Una aplicación móvil llamada “DroneControl” sirve para controlar todas las funcionalidades de un drone marca X. La vía en que se conecta el dispositivo móvil con el drone es mediante Wifi, al estar enlazados el usuario entra a la aplicación y hace clic donde dice “connect to drone”, en ese mismo momento la aplicación utiliza el método connectToDrone() que se encuentra almacenada en el X-SDK (SDK donde se encuentran almacenadas muchas métodos para interactuar con el drone), gracias a este método la aplicación solicita acceso a controlar el drone, ya que puede ser que otro dispositivo móvil está controlando en ese mismo momento al drone, si la conexión es exitosa la aplicación muestra al usuario el menú principal de funcionalidades del drone, caso contrario, le muestra un mensaje de alerta de que el drone está ocupado. Una de estas funcionalidades es mostrar la galería multimedia de las fotos y videos que están almacenadas en el drone, cuando el usuario hace clic en “Media” la aplicación utiliza el método getDroneMode() para averiguar en qué modo está el drone, ya que para acceder a la sd del drone este debe estar en el modo Download, si el drone no está en el modo Download se debe llamar al método setCameraMode(“Download”), cabe agregar que los 2 métodos nombrados son parte del X-SDK. Para poder mostrar las imágenes pequeñas de la galería se deben descargar imágenes de baja resolución, para esto primero se debe descargar un lista donde se encontrará la información de cada imagen o video, esta lista se obtiene utilizando el método fetchMediaListWithCompletion() del X-SDK, esta lista tiene referencias a objetos llamados MediaX, donde X es el índice de cada imagen o video, ahora para descargar la imagen de baja resolución de cada uno de estos objetos se debe llamar al método MediaX.fetchThumbnailWithCompletion del X-SDK, para cada objeto Media, luego de recibir cada imagen, llamamos a la función showImage(MediaX.Thumbnail) para ir mostrando cada una de estas imagen, showImage está almacenada en la librería avMediaPlayer. Realice una diagrama de secuencia de componente de la galería.

Respuesta
Ejercicio 6