



| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 1 de 27 |

**MANUAL DE PRÁCTICAS DE
TOPICOS AVANZADOS DE PROGRAMACIÓN
PROGRAMA EDUCATIVO:
INGENIERÍA EN SISTEMAS COMPUTACIONALES**

AUTOR: JOSE LUIS LIRA TURRIZA


Calkiní, Campeche, a 14 de Julio de 2016

| Revisó | Aprobó | Autorizó |
|---------------------------------|-------------------------------|------------------------------|
| Gonzalo M. Quetz Aguirre | Jose Luis Lira Turriza | Miguel A. Cohuo Ávila |
| Presidente de Academia | Coordinador del PE | Dirección Académica |

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 2 de 27 |

ÍNDICE

| CONCEPTO | PÁGINAS |
|---|---------|
| PRESENTACIÓN..... | 3 |
| OBJETIVO GENERAL..... | 3 |
| SEGURIDAD | 3 |
| PRÁCTICA No. 1.- Generación de Aplicaciones con UI y eventos..... | 4 |
| PRÁCTICA No. 2.- Generación de Aplicaciones con gráficos en un componente tipo canvas | 10 |
| PRÁCTICA No. 3.- Bibliotecas y paquetes a partir de requerimientos | 14 |
| PRÁCTICA No. 4.- Programación concurrente para conexión a un origen de datos..... | 17 |
| PRÁCTICA No. 5.- Desarrollo de aplicación para móviles con formularios | 21 |
| PRÁCTICA No. 6.- Desarrollo de Pong para dispositivos móviles..... | 24 |

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 3 de 27 |

PRESENTACIÓN

El presente documento contiene las prácticas relacionadas a la asignatura TÓPICOS AVANZADOS DE PROGRAMACIÓN en la que se busca la implementación de aplicaciones computacionales para solucionar problemas en los que se integren tecnologías de programación concurrente y acceso a datos sobre diferentes plataformas gráficas y de programación móvil.

Para la obtención de las competencias es necesario que el estudiante tenga competencias previas en cuanto a paradigmas de programación, el uso de metodologías para la solución de problemas mediante la construcción de algoritmos utilizando un lenguaje de programación orientada a objetos, el manejo de conceptos básicos de Hardware y Software, construcción de modelos de software empleando el lenguaje de modelado unificado (UML).


Dentro de las competencias a adquirir se presentan prácticas sobre la Interfaz Gráfica de Usuario (GUI), a través de componentes gráficos en el entorno de desarrollo con manejo de eventos, creación de bibliotecas, implementación de soluciones multihilo, acceso a datos y de manera introductoria el uso de dispositivos móviles.

OBJETIVO GENERAL

El presente manual tiene como objetivo adquirir las competencias de desarrollo de soluciones de software para resolver problemas en diversos contextos utilizando programación concurrente, acceso a datos, que soporten interfaz gráfica de usuario y consideren dispositivos móviles.

SEGURIDAD

Es necesario para la realización de las prácticas de este manual leer el reglamento de Licenciatura que se puede encontrar en la liga siguiente: _____.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 4 de 27 |

PRÁCTICA No. 1.- Generación de Aplicaciones con UI y eventos

-INTRODUCCIÓN

Programación orientada a eventos.


La programación dirigida por eventos es un paradigma de la programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

Para entender la programación dirigida por eventos, podemos oponerla a lo que no es: mientras en la programación secuencial (o estructurada) es el programador el que define cuál va a ser el flujo del programa, en la programación dirigida por eventos será el propio usuario o lo que sea que esté accionando el programa, el que dirija el flujo del programa. Aunque en la programación secuencial puede haber intervención de un agente externo al programa, estas intervenciones ocurrirán cuando el programador lo haya determinado, y no en cualquier momento como puede ser en el caso de la programación dirigida por eventos.

El creador de un programa dirigido por eventos debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el administrador del evento. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo programador.

En la programación dirigida por eventos, al comenzar la ejecución del programa se llevarán a cabo las inicializaciones y demás código inicial y a continuación el programa quedará bloqueado hasta que se produzca algún evento. Cuando alguno de los eventos esperados por el programa tenga lugar, el programa pasará a ejecutar el código del correspondiente administrador de evento. Por ejemplo, si el evento consiste en que el usuario ha hecho clic en el botón de play de un reproductor de películas, se ejecutará el código del administrador de evento, que será el que haga que la película se muestre por pantalla.

Un ejemplo claro lo tenemos en los sistemas de programación Lógico y Visual Basic, en los que a cada elemento del programa (objetos, controles, etcétera) se le asignan una serie de eventos que generará dicho elemento, como la pulsación de un botón del ratón sobre él o el redibujado del control.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 5 de 27 |

La programación dirigida por eventos es la base de lo que llamamos interfaz de usuario, aunque puede emplearse también para desarrollar interfaces entre componentes de Software o módulos de núcleos.

-OBJETIVO

El estudiante deberá de identificar los elementos de programación de eventos además del desarrollo de aplicaciones de software, en base al uso de diagramas uml y programación en el lenguaje Java.

LUGAR

AULA

-SEMANA DE EJECUCIÓN

SEMANA DOS (Parcial 1)

- MATERIAL Y EQUIPO

- Sistema Operativo
- Procesador de Textos
- Software para el desarrollo de aplicaciones “eclipse”.
- Cañón
- Plumones
- Pizarrón.
- Equipos de cómputo para todos los estudiantes de la asignatura.

- DESARROLLO DE LA PRÁCTICA

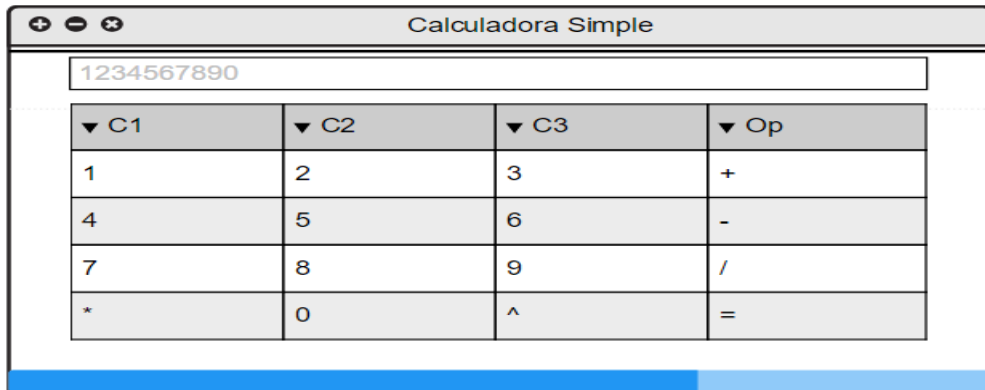


Figura 1. Calculadora Simple


Se desea manipular un conjunto de elementos para implementar la funcionalidad de las operaciones de una calculadora simple de acuerdo a lo siguiente:

1.- Se requiere realizar una aplicación de escritorio que realice los cálculos simples de números enteros de hasta 20 dígitos. En ella solamente se podrá introducir un número por vez seguido de una operación matemática; las operaciones matemáticas tienen prioridades para realizarse que deben aplicarse en la calculadora; el resultado final deberá mostrarse cuando se presione el operador =. En el caso de que la operación pueda realizarse se debe mostrar el resultado en el mismo espacio donde se introducen los números, en caso de algún error, éste se debe mostrar en el mismo espacio. Los errores que se deben considerar son los de división entre cero y desborde de memoria en caso de exceder los 20 dígitos. En esta tarea se deben escribir los casos de uso de la aplicación y los diagramas de caso de uso. También se debe representar la aplicación al menos en los diagramas UML de clases, de secuencia y de casos de uso.

2.- Una vez que se ha diseñado la aplicación se deberá desarrollar la aplicación dividida en dos partes: la interfaz de usuario y la lógica.

2.1 La interfaz gráfica de usuario (GUI por sus siglas en inglés) debe ser construido a partir del siguiente esqueleto:

```
public class CalculadoraGUI extends JFrame
{
    // Los atributos pueden ser componentes de la interfaz como
    // botones y campos de texto
    // En el constructor podemos tener un parámetro que sea el
```

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 7 de 27 |

```
//título que le queremos dar al marco, dicho parámetro se lo
//pasaremos al constructor
public CalculadoraGUI (String nombre)
{
    super (nombre);
    // A continuación se definen y añaden las componentes de la
    //interfaz
}
}
```

En esta etapa se deben inicializar cada uno de los elementos que para esta práctica serán 16 botones distribuidos dentro de un grid, y un campo de texto para introducir los números y mostrar los resultados. Configura los elementos de la siguiente manera:

Ventana principal con título “Calculadora Simple” de tamaño en pixeles de 500 x 400.

Campo de Texto con el texto inicial de “1234567890” en color gris.

16 botones en formato plano como se muestra en la figura 1, cuyo texto sean los números del 1 al 9, 0, +, -, *, / y ^, como los números naturales, el cero, suma, resta, multiplicación, división y potencia respectivamente. La primera fila contiene etiquetas que hacen referencia a las columnas C1, C2, C3 y Op., incluye éstos con ese texto.

Utiliza para la distribución general de la calculadora un administrador de componentes de tipo BorderLayout y para páneles internos un administrador GridLayout.

Implementa la visibilidad de la ventana en la creación de instancia de la clase. Al finalizar ejecuta tu aplicación y evidencie su resultado con una imagen de su escritorio.

2.2 La lógica de la aplicación deberá estar diseñada de acuerdo a un patrón. En nuestro caso utilizaremos el Modelo-Vista-Controlador (MVC) cuya vista ha sido desarrollada en el punto 2.1 y por lo que deberemos construir el Controlador y el Modelo de acuerdo al diagrama de la figura 2.

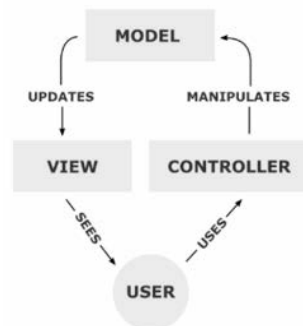



Figura 2. Patrón Modelo-Vista-Controlador (MVC)

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 8 de 27 |

2.2.1 El Modelo. Estará compuesto por la implementación de la lógica de negocio en la que se implementarán las operaciones tales como las CRUD o alguna otra necesaria como por ejemplo validarNumero; las clases de tipo Value Object(VO) en el que representamos los objetos con sus atributos y sus métodos set y get; y las clases de tipo data access object (DAO) que serán implementados como clases Hlp con un método action de cada tipo de operación de acceso a los datos.

2.2.2 El Controlador. Esta parte del patrón define la lógica de administración del sistema, establece la conexión entre la vista y el modelo.

Para la implementación del controlador se creará una clase Principal única y una clase Controlador por cada interfaz gráfica de la aplicación. Estos últimos deben contener la relación entre el modelo y las vistas.

En la clase Principal implementar el método main donde se instanciarán cada una de las vistas, del modelo y del controlador.


En la clase Controlador se deben tener instancias de la vista y de la lógica del negocio y métodos que permitan realizar las acciones definidas en las dos partes, como por ejemplo sumar o multiplicar.

Para terminar el armado del patrón, se modificará la vista incluyendo la implementación del ActionListener a la clase para obtener los datos de la interfaz gráfica y vaciarlos al objeto y viceversa a través de la implementación de los métodos dataToUserInterface y userInterfaceToData así como el método actionPerformed.

3. Realiza pruebas del prototipo con datos límite de acuerdo al planteamiento del problema, evidencie esta etapa con el diseño de una prueba con el siguiente formato:

| | |
|-----------------------------|--|
| Nombre de la prueba: | |
| Descripción: | |
| Datos de entrada de prueba: | |
| Datos de salida esperado: | |

4. Realice un informe de la práctica en donde se muestren los resultados y presente una discusión analizando los beneficios del patrón en la aplicación desarrollada.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 9 de 27 |

- EVALUACIÓN Y RESULTADOS


Se describe la forma de evaluar la práctica desarrollada mediante la entrega del Informe Técnico solicitado por el Profesor, el cual puede contener tablas, planos, prototipos, gráficas, diagramas o dibujos, observaciones, conclusiones, cuestionario y referencias.

-REFERENCIAS

Se presenta el listado de bibliografías utilizadas en el fundamento teórico y en el desarrollo de la práctica en sistema de referencia APA.

-ANEXOS

Se recomienda poner en anexos las partes del trabajo que, intercaladas en medio del texto romperían la continuidad de lectura del mismo, por ejemplo deducciones detalladas de alguna expresión, cálculos largos y detallados, códigos y enumeración detallada de algunos componentes. El docente según considere podría proporcionar tablas de valores, indicadores, manuales de equipos u otros.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 10 de 27 |

PRÁCTICA No. 2.- Generación de Aplicaciones con gráficos en un componente tipo canvas

- INTRODUCCIÓN

La interfaz es la parte visible de las aplicaciones, siendo lo que se percibe de las mismas; por ello, cada vez se les está dando una importancia mayor y se está poniendo más cuidado en su desarrollo. La creación de interfaces de usuario es un área, dentro del desarrollo de software, que ha evolucionado mucho en los últimos años y lo sigue haciendo a día de hoy.


Se podría decir que la interfaz de usuario es lo primero que se juzga de una aplicación, y si no tiene la calidad adecuada puede producir rechazo por parte del usuario. Una interfaz puede incluso poner limitaciones en la comunicación de la máquina con el usuario. Todo aquello que no se pueda expresar a través de la interfaz se perderá, por ello, tiene gran importancia dentro del desarrollo de una aplicación el diseño de su interfaz.

Los elementos que componen la interfaz gráfica son elementos visuales, y a través de ellos el usuario puede interactuar con la aplicación. En esta interacción el usuario introduce datos que el programa necesita para llevar a cabo su funcionalidad y obtiene los resultados de procesar dichos datos. Por ejemplo, las ventanas, los botones, las imágenes, etc. Son elementos gráficos. Una diferencia clara entre una aplicación de consola y una aplicación con interfaz gráfica de usuario, es que la primera no tiene ningún elemento gráfico, mientras que en la segunda éstos si existen.

Las interfaces gráficas están formadas por ventanas de diferentes tipos que se pueden solapar, mover, cerrar, etc. Dentro de estas ventanas se encuentran otros elementos (botones, etiquetas, campos de texto, imágenes, etc.) que permiten introducir datos y mostrar el estado de la aplicación. El ratón y el teclado permiten manejar los elementos que forman parte de la interfaz.

Para crear nuevas formas existe la clase canvas. Ésta existe para tener descendientes, no hace nada por sí misma; sólo proporciona una forma para implementar un componente personalizado. Por ejemplo, los Lienzos (Canvas) son útiles para áreas de dibujo para imágenes y gráficos del cliente, tanto si se desea o no manejar eventos que ocurran dentro del área de pantalla.

Los Canvas también son útiles cuando se quiera un control -- un botón, por ejemplo -- que se parezca a la implementación por defecto de ese control. Como no se puede cambiar la apariencia de los controles estándares creando una subclase de la clase Component correspondiente (Button, por ejemplo), en su lugar se puede crear una subclase de Canvas que tenga el comportamiento que se quiera y el mismo comportamiento que la implementación por defecto del control.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 11 de 27 |

-OBJETIVO

El estudiante será capaz de implementar interfaces gráficas utilizando las clases canvas para dibujar en su superficie.

LUGAR


AULA

-SEMANA DE EJECUCIÓN

SEMANA CUATRO (Parcial 1)

- MATERIAL Y EQUIPO

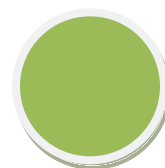
- Sistema Operativo
- Procesador de Textos
- Software para el desarrollo de aplicaciones “eclipse”.
- Cañón
- Plumones
- Pizarrón.
- Equipos de cómputo para todos los estudiantes de la asignatura.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 12 de 27 |

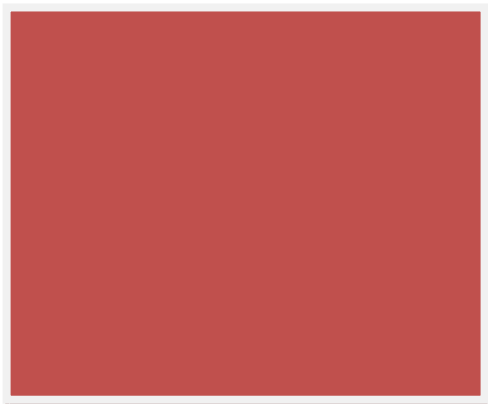
- DESARROLLO DE LA PRÁCTICA

Una empresa de desarrollo de video juegos pretende crear sus propios componentes visuales para sus juegos, los componentes que desea crear deben ser dependientes del teclado y del mouse debido a que son juegos de PC. Se desean crear 3 componentes diferentes, todos ellos asociados a los botones comunes de las aplicaciones de escritorio, por lo que se desea que reaccionen al clic del mouse.

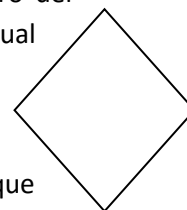
1. El primer componente es el botón, en éste se debe poner el texto asociado a la acción a utilizar y la posibilidad de marcar una letra para poder ejecutar la acción rápida con el teclado.



2. El segundo elemento es un contenedor, que permitirá “arrastrar” los objetos de un lado a otro dentro de su área dibujable con los elementos del mouse.




3. Y el tercer componente será un contenedor temporal, que contendrá un conjunto de componentes “arrastrables” que permitan llevar los objetos que se usarán dentro del componente contenedor y del cual se recuperarán dichos componentes.



4. Escribe una aplicación que utilice los diseños creados y demuestren la funcionalidad de cada uno.
5. Documente los resultados.

- EVALUACIÓN Y RESULTADOS


Se describe la forma de evaluar la práctica desarrollada mediante la entrega del Informe Técnico solicitado por el Profesor, el cual puede contener tablas, planos, prototipos, gráficas, diagramas o dibujos, observaciones, conclusiones, cuestionario y referencias.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 13 de 27 |

-REFERENCIAS

Como utilizar la clase Canvas. Link:
[http://www.binarycode.com/bdescargas/Manuales%20y%20Documentos/JAVA/Interfaces%20de%20Usuario/Tutorial%20JAVA%20avanzado%20\(I\)/gui/utilizacion/uicanvas.html](http://www.binarycode.com/bdescargas/Manuales%20y%20Documentos/JAVA/Interfaces%20de%20Usuario/Tutorial%20JAVA%20avanzado%20(I)/gui/utilizacion/uicanvas.html). Febrero de 2016

-ANEXOS

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 14 de 27 |

PRÁCTICA No. 3.- Bibliotecas y paquetes a partir de requerimientos

- INTRODUCCIÓN

Para ayudar a explicar el tema de paquetes se debe imaginar una ciudad en la cual hay varios bloques de apartamentos que pertenecen a una única empresa inmobiliaria. Además de los departamentos, la empresa tiene diferentes servicios como comercios, zonas de recreo y almacenes. La empresa tiene organizadas todas sus propiedades como una lista de referencias, para que pueda hacer uso de ella en el momento en que lo necesite.

Si ahora se mira lo anterior en términos de Java, la empresa inmobiliaria es el *paquete*. Los paquetes agrupan a bibliotecas de clases, como las bibliotecas que contienen información sobre distintas propiedades comerciales. Un paquete será entonces, la mayor unidad lógica de objetos en Java.

Los paquetes se utilizan en Java de forma similar a como se utilizan las bibliotecas en C++, para agrupar funciones y clases, sólo que en Java agrupan diferentes clases y/o interfaces. En ellas las clases son únicas, comparadas con las de otros paquetes, y además proporcionan un método de control de acceso. Los paquetes también proporcionan una forma de ocultar clases, evitando que otros programas o paquetes accedan a clases que son de uso exclusivo de una aplicación determinada.


-OBJETIVO

Al finalizar la práctica, el alumno podrá tener la capacidad de crear paquetes dentro de un sistema informático de acuerdo a un conjunto de requerimientos.

LUGAR

AULA

-SEMANA DE EJECUCIÓN

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 15 de 27 |

Semana SIETE (Parcial 2)

- MATERIAL Y EQUIPO

- Sistema Operativo
- Procesador de Textos
- Software para el desarrollo de aplicaciones “eclipse”.
- Cañón
- Plumones
- Pizarrón.
- Equipos de cómputo para todos los estudiantes de la asignatura.


- DESARROLLO DE LA PRÁCTICA

Los paquetes que pertenecen a un lenguaje fortalecen y aceleran el desarrollo de las aplicaciones en él. En esta práctica deberán elaborar una aplicación que permita consultar una página web (itescam de preferencia) y traer los datos informativos (IP, documento principal, directorios existentes, tiempo de carga de la página, etc) de ella.

1. De acuerdo a las necesidades de la aplicación crear un paquete llamado informacionweb, que contenga las clases que representen la información web que se pueda traer de una página web. De ser posible crear paquetes internos que permita clasificar las clases de acuerdo a sus características.
2. Realiza la tarea tanto en Java como en C#.
3. Realiza una comparativa con los dos lenguajes remarcando sus diferencias.

- EVALUACIÓN Y RESULTADOS


Se describe la forma de evaluar la práctica desarrollada mediante la entrega del Informe Técnico solicitado por el Profesor, el cual puede contener tablas, planos, prototipos, gráficas, diagramas o dibujos, observaciones, conclusiones, cuestionario y referencias.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 16 de 27 |

-REFERENCIAS

Agustín Froufe. Paquetes y declaración de paquetes. Link:
<http://dis.um.es/~bmoros/Tutorial/parte5/cap5-14.html>. Febrero de 2016

-ANEXOS

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 17 de 27 |

PRÁCTICA No. 4.- Programación concurrente para conexión a un origen de datos.

- INTRODUCCIÓN

La creación de hilos junto con las conexiones remotas permite interactuar a más de un usuario dentro del mismo sistema.

Un hilo es un hilo de ejecución en un programa. La JVM permite a una aplicación tener múltiples hilos de ejecución corriendo concurrentemente.

Cada hilo tiene una prioridad. Los Hilos con una prioridad alta son ejecutados sobre los hilos con baja prioridad. Cada hilo puede o puede que no también ser marcado como un “daemon”. Cuando el código corre algún hilo crea un nuevo objeto Thread, el nuevo hilo tiene su prioridad inicialmente igual a la prioridad del hilo que lo crea, y es un hilo “daemon” si y solo si el hilo que lo crea lo es.

Cuando una JVM comienza, hay usualmente un hilo simple “no-daemon” (el cual típicamente llama al método llamado main de alguna clase diseñada). La JVM continua ejecutando hilos hasta algo de lo siguiente ocurre:

El método exit de la clase Runtime es llamado y el administrador de seguridad ha permitido la operación de salida.


Todos los hilos que no son “daemon” han muerto, ya sea por regresando al método que llamó al método run o lanzando una excepción que se propaga más allá del método run.

Hay dos maneras para crear un nuevo hilo de ejecución. Uno es declarar una clase a ser una subclase de Thread. Esta subclase debería sobrescribir el método run de dicha clase. Una instancia de la subclase puede entonces ser localizada y comenzar.

La otra manera para crear un hilo es declarar una clase que implementa la interface Runnable. Esta clase implementa el método run. Una instancia de la clase puede entonces ser localizada, pasada como un argumento cuando se crea un objeto Thread e iniciar.

-OBJETIVO

Al finalizar la práctica el alumno será capaz de implementar hilos para realizar conexiones de red para una aplicación.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 18 de 27 |

LUGAR


AULA

-SEMANA DE EJECUCIÓN

Semana NUEVE (Parcial 2)

- MATERIAL Y EQUIPO

- Sistema Operativo
- Procesador de Textos
- Software para el desarrollo de aplicaciones “eclipse”.
- Cañón
- Plumones
- Pizarrón.
- Equipos de cómputo para todos los estudiantes de la asignatura.

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 19 de 27 |

- DESARROLLO DE LA PRÁCTICA

Battleship o Batalla Naval es un juego de lápiz y papel tradicional de adivinanza que involucra a dos participantes.


Se compone de dos tableros por jugador, dividido cada uno en cuadrículas. Los tableros típicos son cuadrados de 10 por 10 casillas, y cada posición se identifica con números para las columnas (de 1 a 10) y con letras para las filas (de la A a la J). En uno de los tableros el jugador coloca sus barcos y registra los tiros del oponente. En el otro, se registran los tiros propios. Antes de comenzar, cada jugador posiciona los barcos de forma secreta o invisible al oponente, generalmente con el tablero en posición vertical como pizarra. Cada uno ocupa, según su modelo, una cierta cantidad de posiciones, ya sea horizontal o verticalmente. De esta forma, no se permiten lugares solapados, ya que cada uno ocupa posiciones únicas. Ambos participantes poseen y deben ubicar igual número de naves.

Una vez todas las naves han sido posicionadas, se inicia una serie de rondas. En cada ronda, cada jugador en su turno indica una posición del tablero de su oponente. Si esa posición es ocupada por una parte de un barco, el oponente indica averiado (toque o tocado) y el atacante marca con rojo esa posición, con un pin. Cuando todas las posiciones de un mismo barco han sido dañadas debe indicarse hundido dando a conocer tal circunstancia que indicará al atacante la importancia de la nave destruida. Ahora bien, si la posición indicada, efectivamente, no posee un barco alojado, se indica con agua, y será marcada con un pin blanco.

Quien descubra primero todas las naves será el vencedor, pero en caso de que el participante que comenzó la partida hunda el barco en su última jugada, el otro participante tiene una última posibilidad para alcanzar el empate.

1. Crea un paquete que contenga las clases necesarias para el control de la comunicación en la aplicación.
2. Adicionalmente crea un paquete que implemente la interfaz de usuario para la aplicación propuesta.
3. También usa un paquete que implemente la lógica del juego.
4. Ten en cuenta implementar un algoritmo para la versión automática de adversario.
5. Documenta toda la aplicación.


- EVALUACIÓN Y RESULTADOS

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 20 de 27 |

-REFERENCIAS

Oracle © Java Thread. Link: <https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html>.
Febrero 2016.

-ANEXOS

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 21 de 27 |

PRÁCTICA No. 5.- Desarrollo de aplicación para móviles con formularios

- INTRODUCCIÓN

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente. Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Ya hemos visto que existen 2 configuraciones CLDC y CDC, cada una con unas características propias que veremos en profundidad más adelante. Como consecuencia, cada una requiere su propia máquina virtual. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM.

En esta práctica se pretende implementar los conceptos de formularios implementados a aplicaciones para J2ME que es la versión móvil de la familia de Java.


-OBJETIVO

El estudiante podrá crear aplicaciones con interfaces de usuario basado en formularios.

LUGAR

AULA

-SEMANA DE EJECUCIÓN

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 22 de 27 |

Semana DOCE (Parcial 3)

- MATERIAL Y EQUIPO


- Sistema Operativo
- Procesador de Textos
- Software para el desarrollo de aplicaciones “eclipse”.
- Cañón
- Plumones
- Pizarrón.
- Equipos de cómputo para todos los estudiantes de la asignatura.

- DESARROLLO DE LA PRÁCTICA

Toda aplicación con un mínimo de seguridad necesita una pantalla de autenticación de usuario (Login) que nos da como resultado dos respuestas: pantalla de bienvenida si el usuario existe o pantalla de error si el usuario no existe.

1. De acuerdo al planteamiento, crear una aplicación en eclipse de tipo J2ME en el que se definirán tres pantallas: formLogin, formLoginOk y formLoginError.
2. También se definen dos cajas de texto y una etiqueta para mostrar la petición al usuario o el mensaje de error: txtUsuario, txtPassword, strMensaje.
3. También se definen los comandos o acciones a realizar sobre las pantallas: cmdEntrar, cmdSalir y cmdAtras.
4. Define un método mostrarPantalla (Displayable nextDisplay) que recibe la siguiente pantalla a mostrar.
5. Implementa un método validarUsuario que obtiene los valores de las cajas de texto y compara con los valores definidos para ser correctos y llama al método dentro del commandAction.
6. Muestra los resultados con pantallas que demuestren su funcionamiento.

- EVALUACIÓN Y RESULTADOS


| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 23 de 27 |

-REFERENCIAS

Sergio Galvez Rojas, Lucas Ortega Díaz. Java a tope: J2ME. Universidad de Málaga. 2003

Heriberto Vazquez. Empezando con J2ME. Link:
<http://programandodemadrugada.blogspot.mx/2011/10/1-empezando-con-j2me.html>. Febrero 2016

-ANEXOS

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 24 de 27 |

PRÁCTICA No. 6.- Desarrollo de Pong para dispositivos móviles

- INTRODUCCIÓN

En esta práctica se describe como desarrollar y publicar un juego básico Pong que corre bajo Android. Se desarrolla el juego con modo de un único jugador. Pong es el hola mundo del desarrollo de los juegos y fue originalmente desarrollado como una versión de ping pong electrónico. La idea es mantener la bola en juego y esperar que el contrincante no lo haga. En esta práctica permite ejemplificar Actividades, Vistas y Canvas para desarrollar animaciones simples para Android.

-OBJETIVO

El estudiante que termine la práctica podrá implementar animaciones a través de clases de Canvas sobre Actividades y vistas implementadas en Android.

LUGAR


AULA

-SEMANA DE EJECUCIÓN

Semana CATORCE (Parcial 3)

- MATERIAL Y EQUIPO

- Sistema Operativo
- Procesador de Textos
- Software para el desarrollo de aplicaciones “eclipse”.
- Cañón

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 25 de 27 |

- Plumones
- Pizarrón.
- Equipos de cómputo para todos los estudiantes de la asignatura.

- DESARROLLO DE LA PRÁCTICA

Desarrollar el juego del PingPong. La mecánica del juego es bien sencilla. Nosotros manejamos una especie de raqueta que debe golpear una pelota para marcar un tanto en el lado contrario del terreno de juego. Como contrincante tenemos a otra raqueta controlada por la máquina y que intentará hacer lo mismo que nosotros para marcarse ella el tanto. El primero de los dos que llegue a 10 tantos gana el partido.

1. Crea una clase PingPong que herede de MIDlet con los siguientes atributos:


```
private Display pantalla;
private Menu m;
private Tablero t;
private Resultado r;
private Dificultad d;
private int nivel = 1;
private NuevoRecord nr;
```

2. Para cada atributo declarado crea un método set y uno get que controle los valores de cada uno.
3. Crea la clase Menu que herede de List e implemente de CommandListener con dos atributos: midlet y salir, de tipo PingPong y Command respectivamente con un constructor como se muestra a continuación:

```
public Menu(PingPong m){
    super("Menu",List.IMPLICIT);
    this.append("Jugar",null);
    this.append("Dificultad",null);
    this.append("Resultados",null);
    this.midlet = m;
    salir = new Command("Salir",Command.EXIT,1);
    this.addCommand(salir);
    this.setCommandListener(this);
}
```

4. De igual manera crea la clase Dificultad con las mismas herencias que Menu con el siguiente constructor:

```
public Dificultad(PingPong m){
    super("Dificultad",List.EXCLUSIVE);
    this.append("Baja",null);
    this.append("Normal",null);
    this.append("Alta",null);
    this.setSelectedIndex(0,true);
    midlet = m;
    atras = new Command("Atras",Command.BACK,1);
    aceptar = new Command("Aceptar",Command.OK,1);
}
```

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 26 de 27 |

```
this.addCommand(aceptar);
this.addCommand(atras);
this.setCommandListener(this);
```

5. Crea la clase NuevoRecord que herede de Form con CommandListener con los siguientes atributos y constructor:

```
private TextField txtNombre;
private Command aceptar;
private PingPong midlet;
private long puntuacion;
public NuevoRecord(PingPong m){
    super("Nuevo Record");
    midlet = m;
    puntuacion = 0;
    txtNombre = new TextField("Nombre", "", 3, TextField.ANY);
    aceptar = new Command("Aceptar", Command.OK, 1);
    this.append(txtNombre);
    this.addCommand(aceptar);
    this.setCommandListener(this);
}
```

6. Crea la clase Pelota y la clase Paleta que herende de Sprite que pinte la pelota del juego y la raqueta con las imágenes de los Anexos, la pelota debe ser dibujada hacia una dirección hacia x o y.
7. Por último crea una clase Tablero que herede de GameCanvas e implemente Runnable y CommandListener, esta clase será la principal y que reúne las clases anteriores. Diseñalas de acuerdo al bosquejo de métodos siguientes:


```
crearFondo(): Nos crea el fondo de pantalla.
crearPelota(): Nos crea el objeto pelota.
crearPaleta(): Nos crea el objeto paleta.
inicializar(): Nos inicializa los elementos del juego.
start(): Comenzamos el juego.
run(): Posee el cuerpo principal del juego.
verColisiones(): Detecta cualquier colisión que produzca la pelota y actúa en consecuencia.
calcularTrayectoria(Paleta pal): Calcula la nueva trayectoria de la pelota tras ocurrir una colisión.
verEntrada(): Aquí se detecta cualquier pulsación de teclas que efectúe el usuario.
moverPelota(): Movemos la pelota según la dirección que posea.
dibujar(Graphics g): Dibujamos todos los elementos por pantalla.
mostrarResultados(int r1, int r2): Nos muestra el resultado por pantalla tras un tanto.
```

8. Muestra las pantallas diseñadas en el emulador de J2ME

- EVALUACIÓN Y RESULTADOS

Se describe la forma de evaluar la práctica desarrollada mediante la entrega del Informe Técnico solicitado por el Profesor, el cual puede contener tablas, planos, prototipos, gráficas, diagramas o dibujos, observaciones, conclusiones, cuestionario y referencias.

-REFERENCIAS

| | | |
|--|---------------------|-------------------------|
|  | Dirección Académica | Código: CPE-FO-02-03 |
| | | Revisión: 1 |
| | MANUAL DE PRÁCTICAS | Página: 27 de 27 |

Patrick Kalkman. Androng, a pong clone for Android. Link:
<http://www.codeproject.com/Articles/189515/Androng-a-Pong-clone-for-Android#Introduction0>.
 Febrero 2016

-ANEXOS

ANEXO 1. Imagen de pelota de pong.



ANEXO 2. Imagen de Pong original

