EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Minatitlán

Ingeniería En Sistemas Computacionales

“MANUAL DE PRÁCTICAS DE LA MATERIA DE TÓPICOS AVANZADOS DE PROGRAMACIÓN”

MINATITLÁN, VER. SEPTIEMBRE 2023

INDICÉ

INTRODUCCIÓN

Este manual está diseñado para ofrecer a los estudiantes una guía completa y práctica para el aprendizaje y la aplicación de conceptos avanzados en el mundo de la programación.

En esta materia, exploraremos una amplia gama de temas que abarcan desde la programación concurrente hasta el acceso a datos y el desarrollo de aplicaciones móviles. A través de una serie de prácticas, los estudiantes tendrán la oportunidad de poner en práctica sus habilidades, abordar desafíos reales y desarrollar soluciones concretas, y en las mismas se consideran todos los temas de la asignatura, del 1 al 5, esas son las que consideraron.
Este manual servirá como una referencia constante a medida que avancen en su curso. Cada práctica está diseñada para brindar una experiencia de aprendizaje práctica y aplicable.

En el plan de estudios se proponen las prácticas que se pueden desarrollar en la asignatura.

Para la documentación de cada práctica se debe de incluir:

1. *Definición*
    1. De que se trata el caso de estudio

2. *Análisis*
    1. Tabla de Entrada Proceso Salida, *IPO Chart*

3. *Diseño*
    1. GUI
    2. Diccionario de objetos
    3. Diccionario de eventos

4. *Desarrollo*
    1. Nombre de la clases .java y .form anexando dichos archivos

Tema 1 - Interfaz gráfica de usuario.

Competencia Específica:

Desarrolla programas para interactuar con el usuario de una manera amigable, utilizando GUI (Interfaz Gráfica de Usuario) manipuladas a través de eventos.

Elabora tablas de entrada, proceso y salida, *IPO Chart*, GUI, diccionarios de datos y de eventos para los casos de estudio planteados.

Practica 1. Editar compilar y ejecutar distintos programas que incluyan el uso de formularios, botones, etiquetas y cajas de texto.

Objetivo: *El objetivo de esta práctica es familiarizarse con el desarrollo de aplicaciones que involucran la creación y manipulación de formularios interactivos. Aprenderás a utilizar elementos como botones, etiquetas y cajas de texto en la creación de programas funcionales.*

Desarrollo:

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido o una herramienta de desarrollo de tu elección.

Creación de una Aplicación en Blanco:

- Crea un nuevo proyecto o archivo en blanco en tu IDE.

Diseño de la Interfaz Gráfica:

- Diseña una interfaz gráfica que incluya al menos un formulario.
- Agrega etiquetas para describir los elementos.
- Incluye una o varias cajas de texto para la entrada de datos.
- Agrega botones que permitan realizar acciones en la aplicación.

Programación de la Interacción:

- Asocia eventos a los botones y cajas de texto.
- Programa la lógica de la aplicación para que responda a las interacciones del usuario.
- Por ejemplo, puedes crear un programa que tome valores de las cajas de texto, realice algún cálculo y muestre los resultados en etiquetas o en una nueva ventana.

Compilación y Ejecución:

- Compila el programa para asegurarte de que no haya errores.
- Ejecuta la aplicación y verifica que la interfaz gráfica sea funcional.

Pruebas y Depuración:

- Realiza pruebas exhaustivas de la aplicación para asegurarte de que todas las funcionalidades estén trabajando correctamente.
- Si encuentras errores, depura el código y realiza las correcciones necesarias.

Documentación:

- Documenta el código de manera adecuada según lo solicitado por el docente haciendo uso de tablas de entrada, proceso y salida, *IPO Chart*, GUI y explicando la funcionalidad de cada parte importante del programa.

Notas adicionales:

A medida que trabajes en esta práctica, asegúrate de entender cómo se comunican los elementos de la interfaz gráfica con la lógica de programación.

Experimenta con diferentes diseños y características para mejorar tus habilidades en la creación de aplicaciones interactivas.

Practica 2. Desarrollar aplicaciones que incluyan la programación de eventos.

Objetivo: El objetivo de esta práctica es aprender a desarrollar aplicaciones que respondan a eventos específicos. A través de la programación de eventos, podrás crear aplicaciones interactivas que reaccionen a las acciones del usuario.

Desarrollo

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido o una herramienta de desarrollo de tu elección.

Creación de un Nuevo Proyecto o Archivo:

- Crea un nuevo proyecto en blanco o archivo de código, según las capacidades de tu entorno de desarrollo.

Diseño de la Interfaz Gráfica:

- Diseña una interfaz gráfica que incluya elementos como botones, etiquetas, cajas de texto u otros controles interactivos.

Programación de Eventos:

- Define eventos específicos que desees capturar, como hacer clic en un botón o ingresar texto en una caja de texto.
- Asocia funciones o métodos a estos eventos para que se ejecuten cuando ocurra el evento.

Lógica de la Aplicación:

- Dentro de las funciones asociadas a los eventos, programa la lógica de la aplicación.
- Por ejemplo, si tienes un botón "Calcular", programa una función que realice cálculos en función de la entrada del usuario y muestre los resultados en la interfaz gráfica.

Compilación y Ejecución:

- Compila el programa para asegurarte de que no haya errores.
- Ejecuta la aplicación y verifica que los eventos y la interacción con el usuario funcionen correctamente.

Pruebas y Depuración:

- Realiza pruebas exhaustivas de la aplicación para asegurarte de que los eventos se capturen y manejen adecuadamente.
- Si encuentras errores, depura el código y realiza las correcciones necesarias.

Documentación:

- Documenta el código de manera adecuada según lo solicitado por el docente haciendo uso de tablas de entrada, proceso y salida, *IPO Chart*, GUI y explicando la funcionalidad de cada parte importante del programa.

Notas adicionales:
Experimenta con diferentes tipos de eventos, como eventos de teclado, eventos de ratón y eventos personalizados según las necesidades de tu aplicación.
Asegúrate de comprender cómo se manejan y gestionan los eventos en el entorno de desarrollo que estás utilizando, ya que puede variar según la tecnología o lenguaje de programación.

Practica 3. Desarrollar aplicaciones que incluyan la generación de nuevos eventos.

Objetivo: El objetivo de esta práctica es aprender a desarrollar aplicaciones que generen eventos personalizados. A través de la generación de eventos, podrás crear aplicaciones más dinámicas y controlar mejor la interacción con el usuario.

Desarrollo

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido o la herramienta de desarrollo que prefieras.

Creación de un Nuevo Proyecto o Archivo:

- Crea un nuevo proyecto en blanco o archivo de código, según las capacidades de tu entorno de desarrollo.

Diseño de la Interfaz Gráfica:

- Diseña una interfaz gráfica que incluya elementos interactivos, como botones, etiquetas, cajas de texto u otros controles.

Generación de Eventos Personalizados:

- Identifica un escenario en el que desees generar un evento personalizado. Por ejemplo, puedes crear un botón "Enviar" que genere un evento personalizado cuando se haga clic en él.
- Define y crea el código para generar y lanzar ese evento personalizado en respuesta a una acción específica.

Manejo de Eventos Personalizados:

- En otro lugar de tu código, programa la lógica para manejar y responder a este evento personalizado.
- Puedes realizar acciones específicas cuando se dispare este evento, como enviar datos al servidor o actualizar la interfaz gráfica.

Compilación y Ejecución:

Compila el programa para asegurarte de que no haya errores.
Ejecuta la aplicación y verifica que los eventos personalizados se generen y manejen correctamente.

Pruebas y Depuración:

- Realiza pruebas exhaustivas para asegurarte de que los eventos personalizados funcionen como se espera.
- Si encuentras errores, depura el código y realiza las correcciones necesarias.

Documentación:

- Documenta el código de manera adecuada según lo solicitado por el docente haciendo uso de tablas de entrada, proceso y salida, *IPO Chart*, GUI y explicando la funcionalidad de cada parte importante del programa.

Notas adicionales:
Asegúrate de comprender la sintaxis y la estructura de los eventos personalizados en el lenguaje o tecnología que estás utilizando.
Los eventos personalizados son útiles cuando deseas crear una arquitectura de aplicación más modular y escalable.

Practica 4. Desarrollar aplicaciones que maneje gráficos en un componente tipo canvas.

Objetivo: El objetivo de esta práctica es aprender a desarrollar aplicaciones que utilicen un componente tipo canvas para dibujar gráficos y visualizar información de manera interactiva.

Desarrollo

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido o la herramienta de desarrollo que prefieras.

Creación de un Nuevo Proyecto o Archivo:

- Crea un nuevo proyecto o archivo en blanco según las capacidades de tu entorno de desarrollo.

Diseño de la Interfaz Gráfica:

- Diseña una interfaz gráfica que incluya un componente tipo canvas en el que puedas dibujar.
- Puedes agregar botones u otros controles para interactuar con los gráficos, como cambiar colores, formas o datos a visualizar.

Programación del Componente Canvas:

- Utiliza las funciones proporcionadas por el lenguaje o la biblioteca gráfica que estés utilizando para dibujar en el componente canvas.
- Experimenta con la representación gráfica de datos, como gráficos de barras, líneas, círculos o cualquier otro elemento gráfico que sea relevante para tu aplicación.

Interacción con el Usuario:

- Programa la interacción con el usuario para que pueda interactuar con los gráficos en el canvas.
- Por ejemplo, permite que el usuario dibuje o manipule los elementos gráficos.

Compilación y Ejecución:

- Compila el programa y ejecútalo para asegurarte de que los gráficos se muestren correctamente en el canvas.

Pruebas y Depuración:

- Realiza pruebas exhaustivas para garantizar que los gráficos sean interactivos y se comporten como se espera.
- Si encuentras errores, depura el código y realiza las correcciones necesarias.

Documentación:
Documenta el código adecuadamente, explicando cómo se dibujan los gráficos en el canvas y cómo se interactúa con ellos.

Notas adicionales:
Asegúrate de entender cómo funcionan las coordenadas en el componente canvas y cómo se controla el dibujo en él.
Experimenta con diferentes tipos de gráficos y efectos para mejorar tus habilidades en el manejo de gráficos en un canvas.

Practica 5. Desarrollar aplicaciones que maneje componentes visuales y no visuales proporcionados por la IDE.

Objetivo: El objetivo de esta práctica es aprender a desarrollar aplicaciones que utilicen componentes visuales y no visuales proporcionados por la IDE. Estos componentes pueden incluir controles visuales como botones, barras de desplazamiento, ventanas, así como componentes no visuales como bases de datos o temporizadores.

Desarrollo

Configuración del Entorno de Desarrollo:
- Abre tu entorno de desarrollo integrado (IDE) preferido.

Creación de un Nuevo Proyecto:
- Crea un nuevo proyecto en blanco o basado en plantillas proporcionadas por la IDE.

Diseño de la Interfaz Gráfica:
- Diseña una interfaz gráfica utilizando componentes visuales proporcionados por la IDE. Esto podría incluir botones, etiquetas, campos de texto, listas desplegables, etc.
- Organiza los componentes de manera que la interfaz sea intuitiva y fácil de usar.

Programación de Componentes Visuales:
- Programa la lógica de la aplicación para que los componentes visuales respondan a eventos, como clics en botones o selecciones en listas desplegables.
- Implementa la funcionalidad de la aplicación utilizando estos componentes visuales.

Uso de Componentes No Visuales:
- Si es necesario, utiliza componentes no visuales proporcionados por la IDE. Esto podría incluir conexiones a bases de datos, temporizadores, hilos, etc.
- Utiliza estos componentes para realizar tareas como recuperar datos de una base de datos, realizar operaciones en segundo plano o gestionar eventos no visuales.

Compilación y Ejecución:
- Compila el programa para asegurarte de que no haya errores.
- Ejecuta la aplicación y verifica que los componentes visuales y no visuales funcionen correctamente.

Pruebas y Depuración:
- Realiza pruebas exhaustivas de la aplicación para asegurarte de que todos los componentes funcionen según lo esperado.
- Si encuentras errores, depura el código y realiza las correcciones necesarias.

Documentación:
Documenta el código adecuadamente, explicando cómo se utilizan los componentes visuales y no visuales en la aplicación y cómo se resuelven problemas específicos.

Notas adicionales:
Aprovecha las ventajas que ofrecen los componentes visuales y no visuales proporcionados por la IDE para agilizar el desarrollo de la aplicación.
Asegúrate de entender la forma en que los componentes visuales interactúan con la lógica de tu programa y cómo se manejan los eventos.

Practica 6. Desarrollar aplicaciones que maneje librerías proporcionados por la IDE.

Objetivo: El objetivo de esta práctica es aprender a desarrollar aplicaciones que hagan uso de librerías proporcionadas por la IDE (Entorno de Desarrollo Integrado) o el framework que estés utilizando. Estas librerías pueden incluir funcionalidades avanzadas y componentes que simplifican el desarrollo de aplicaciones.

Desarrollo

Configuración del Entorno de Desarrollo:
- Abre tu IDE y asegúrate de que la librería que deseas utilizar esté configurada correctamente en tu proyecto.

Creación de un Nuevo Proyecto:
- Crea un nuevo proyecto en blanco o basado en plantillas proporcionadas por la IDE.

Identificación de la Librería:
- Decide qué librería proporcionada por la IDE deseas utilizar en tu proyecto. Esto podría ser una librería para manipulación de gráficos, procesamiento de datos, comunicación de red, entre otros.

Diseño de la Aplicación:
- Diseña la estructura y la interfaz de tu aplicación teniendo en cuenta la funcionalidad que proporciona la librería. Por ejemplo, si estás utilizando una librería de gráficos, planifica cómo se mostrarán los gráficos en la interfaz de usuario.

Programación con la Librería:
- Utiliza la librería proporcionada por la IDE para implementar las funcionalidades clave de tu aplicación.
- Aprovecha las funciones y componentes que ofrece la librería para simplificar el desarrollo de tu aplicación.

Compilación y Ejecución:
- Compila el programa para asegurarte de que no haya errores.
- Ejecuta la aplicación y verifica que la librería se esté utilizando correctamente para lograr la funcionalidad deseada.

Pruebas y Depuración:
- Realiza pruebas exhaustivas para asegurarte de que la librería funcione como se espera en tu aplicación.
- Si encuentras errores, depura el código y realiza las correcciones necesarias.

Documentación:
Documenta el código adecuadamente, explicando cómo se utiliza la librería en la aplicación y cómo se resuelven problemas específicos con la ayuda de la librería.

Notas adicionales:
Asegúrate de consultar la documentación de la librería proporcionada por la IDE para comprender completamente sus capacidades y cómo se integran en tu proyecto.
Experimenta con diferentes funcionalidades de la librería para aprovechar al máximo su potencial.

Tema 2 - Componentes y librerías.

Competencia Específica:

Diseña e implementa componentes y librerías para lograr la reutilización de código.

Elabora tablas de entrada, proceso y salida, *IPO Chart*, GUI, diccionarios de datos y de eventos para los casos de estudio planteados.

Practica 1. Desarrollar componentes visuales o no visuales a partir de requerimientos previamente definidos y aplicarlos en el diseño de aplicaciones.

Objetivo: El objetivo de esta práctica es aprender a desarrollar componentes personalizados, ya sean visuales o no visuales, basados en requerimientos específicos y utilizarlos en el diseño de aplicaciones.

Desarrollo

Definición de Requerimientos:

- Comienza por definir claramente los requerimientos para los componentes que deseas desarrollar. Esto incluye determinar su funcionalidad, apariencia y comportamiento.

Elección de Tecnología y Lenguaje:

- Selecciona la tecnología y el lenguaje de programación adecuados para el desarrollo de tus componentes. Esto puede variar según tus necesidades y preferencias, pero asegúrate de que sean compatibles con la plataforma en la que planeas utilizar los componentes.

Diseño y Desarrollo de Componentes:

- Diseña y desarrolla tus componentes siguiendo los requerimientos previamente definidos. Si estás desarrollando componentes visuales, considera la apariencia y la interacción con el usuario. Si son componentes no visuales, asegúrate de que realicen su tarea de manera eficiente.

Pruebas de Componentes:

- Realiza pruebas exhaustivas de los componentes para garantizar que funcionen de acuerdo con los requerimientos. Esto puede incluir pruebas unitarias y de integración.

Integración en una Aplicación:

- Crea una aplicación de ejemplo o utiliza una aplicación existente para integrar y probar tus componentes. Asegúrate de que funcionen correctamente dentro del contexto de una aplicación.

Compilación y Ejecución:

Compila la aplicación que contiene tus componentes y ejecútala para verificar que todo funcione como se esperaba.

Pruebas y Depuración:

- Realiza pruebas adicionales en la aplicación completa para garantizar que los componentes se integren correctamente y no causen problemas. Si se encuentran errores, depura el código y realiza las correcciones necesarias.

Documentación:

- Documenta adecuadamente tus componentes, explicando su funcionalidad, cómo se utilizan y cualquier configuración necesaria. Esta documentación será útil para futuros desarrolladores que utilicen tus componentes.

Notas adicionales:

Asegúrate de seguir buenas prácticas de diseño y desarrollo de software al crear tus componentes para garantizar su calidad y mantenibilidad.

Puedes explorar el uso de herramientas de desarrollo de interfaces gráficas si estás desarrollando componentes visuales para facilitar su diseño y personalización.

Practica 2. Desarrollar librerías a partir de requerimientos previamente definidos y aplicarlos en el diseño de aplicaciones.

Objetivo: El objetivo de esta práctica es aprender a desarrollar librerías personalizadas a partir de requerimientos definidos previamente y aplicarlas en el diseño de aplicaciones para mejorar la reutilización de código y la eficiencia del desarrollo.

Desarrollo

Definición de Requerimientos para la Librería:

- Comienza por definir claramente los requerimientos y funcionalidades que deseas que incluya tu librería. Estos requerimientos deben estar basados en las necesidades de la aplicación que planeas desarrollar.

Elección de Tecnología y Lenguaje:

- Selecciona la tecnología y el lenguaje de programación adecuados para desarrollar tu librería. Asegúrate de que sean compatibles con la plataforma en la que planeas utilizar la librería.

Diseño y Desarrollo de la Librería:

- Diseña y desarrolla la librería siguiendo los requerimientos previamente definidos. Asegúrate de que la librería sea modular, reutilizable y bien documentada.
- Organiza el código en módulos o clases que ofrezcan funcionalidades específicas y cohesivas.

Integración en una Aplicación:

- Crea una aplicación de ejemplo o utiliza una aplicación existente para integrar y probar tu librería. Asegúrate de que la librería funcione correctamente dentro del contexto de la aplicación.

Compilación y Ejecución:

- Compila la aplicación que contiene tu librería y ejecútala para verificar que todo funcione como se esperaba.

Pruebas y Depuración:

- Realiza pruebas adicionales en la aplicación completa para garantizar que la librería se integre correctamente y no cause problemas. Si se encuentran errores, depura el código y realiza las correcciones necesarias.

Pruebas de la Librería:

- Realiza pruebas exhaustivas de la librería para asegurarte de que cumple con los requerimientos y funcione correctamente en diversos escenarios.

Documentación de la Librería:

- Documenta la librería de manera adecuada, incluyendo instrucciones sobre cómo se utiliza, ejemplos de uso y descripciones detalladas de las funciones y clases disponibles.

Notas adicionales:
Asegúrate de seguir buenas prácticas de diseño y desarrollo de software al crear tu librería para garantizar su calidad y facilidad de uso.
Documenta tu librería de manera exhaustiva para que otros desarrolladores puedan entender y utilizar eficazmente tus componentes.

Tema 3 - Programación concurrente (Multihillos).

Competencia Específica:

Crea subprogramas para resolver problemas concurrentes utilizando Multihilos.

Elabora tablas de entrada, proceso y salida, *IPO Chart*, GUI, diccionarios de datos y de eventos para los casos de estudio planteados.

Practica 1. Analizar las diferencias de funcionalidad entre programas de flujo único contra programas de flujo múltiple.

Objetivo: El objetivo de esta práctica es comprender las diferencias fundamentales en la funcionalidad entre programas de flujo único (monohilo) y programas de flujo múltiple (multihilo).

Desarrollo:

Definición de Términos:

- Comienza por definir claramente qué es un programa de flujo único (monohilo) y qué es un programa de flujo múltiple (multihilo). Explica cómo funcionan y cuáles son sus características principales.

Ventajas de los Programas de Flujo Único:

- Analiza las ventajas de los programas de flujo único. Estas pueden incluir simplicidad, facilidad de desarrollo y predictibilidad.

Desventajas de los Programas de Flujo Único:

- Examina las desventajas de los programas de flujo único. Esto puede abordar limitaciones en el rendimiento y la capacidad de respuesta en situaciones intensivas de cómputo o E/S.

Ventajas de los Programas de Flujo Múltiple:

- Analiza las ventajas de los programas de flujo múltiple (multihilo). Estas pueden incluir la capacidad de aprovechar múltiples núcleos de CPU, mejorar la capacidad de respuesta y permitir la concurrencia.

Desventajas de los Programas de Flujo Múltiple:

- Examina las desventajas de los programas de flujo múltiple. Esto puede abordar desafíos como la complejidad adicional, la posibilidad de condiciones de carrera y la necesidad de sincronización.

Ejemplos Prácticos:

- Proporciona ejemplos concretos de situaciones en las que sería más adecuado utilizar un programa de flujo único o un programa de flujo múltiple. Esto puede incluir aplicaciones de procesamiento por lotes, aplicaciones web, aplicaciones de tiempo real, entre otros.

Casos de Uso Reales:

- Investiga y presenta casos de uso reales de programas de flujo único y programas de flujo múltiple en aplicaciones conocidas o en la industria. Esto puede ayudar a ilustrar cómo se aplican en el mundo real.

Resumen y Conclusiones:

- Resume las diferencias clave entre los programas de flujo único y los programas de flujo múltiple.
- Ofrece conclusiones sobre cuándo es apropiado utilizar cada enfoque en el desarrollo de software.

Documentación:

- Documenta tus hallazgos y conclusiones en un informe que sea fácil de entender y seguir.

Practica 2. Desarrollar programas que implementen el uso de un hilo y manipulen sus distintos estados.

Objetivo: El objetivo de esta práctica es aprender a desarrollar programas que utilicen hilos y manipulen sus distintos estados, como la creación, la ejecución y la finalización.

Desarrollo

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido y configura un proyecto nuevo o utiliza uno existente.

Creación de un Hilo:

- Define y crea un hilo en tu programa. Puedes hacerlo mediante la extensión de la clase Thread o la implementación de la interfaz Runnable.

Implementación de la Funcionalidad del Hilo:

- Programa la lógica que el hilo llevará a cabo. Puede ser cualquier tarea que desees que se ejecute en paralelo con el hilo principal.

Manipulación de Estados del Hilo:

- Utiliza métodos y propiedades para manipular los estados del hilo. Algunos de los métodos comunes incluyen start() para iniciar el hilo, sleep() para suspenderlo temporalmente y join() para esperar a que el hilo finalice.

Monitorización de Estados:

- Agrega instrucciones para imprimir mensajes o registrar información sobre el estado actual del hilo en diferentes momentos de su ejecución.

Compilación y Ejecución:

- Compila el programa y ejecútalo para observar cómo cambian los estados del hilo durante su ejecución.

Pruebas y Análisis:

- Realiza pruebas exhaustivas y analiza los registros o mensajes de estado para verificar que el hilo esté funcionando según lo esperado.

Documentación:

- Documenta tu código explicando cómo se crea, manipula y monitoriza el estado del hilo en tu programa.

Asegúrate de entender los conceptos básicos de la programación multihilo, como la creación, la ejecución y la finalización de hilos, así como los posibles estados que puede tener un hilo (activo, en espera, finalizado, etc.).

Practica 3. Desarrollar programas que implementen el uso de varios hilos que compartan recursos y estén sincronizados.

Objetivo: El objetivo de esta práctica es aprender a desarrollar programas que utilicen múltiples hilos que compartan recursos y estén sincronizados correctamente para evitar problemas de concurrencia, como las condiciones de carrera.

Desarrollo

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido y configura un proyecto nuevo o utiliza uno existente.

Definición de Recursos Compartidos:

- Identifica los recursos que serán compartidos entre los hilos. Esto puede incluir variables, estructuras de datos, archivos, o cualquier otro recurso compartido.

Creación de Múltiples Hilos:

- Define y crea varios hilos en tu programa, cada uno de los cuales realizará una tarea que involucra el acceso a los recursos compartidos.

Implementación de Funcionalidades de Hilos:

- Programa la lógica de cada hilo, asegurándote de que accedan a los recursos compartidos de manera segura y sincronizada utilizando mecanismos de sincronización adecuados.

Mecanismos de Sincronización:

- Utiliza mecanismos de sincronización como semáforos, bloqueos o monitores para coordinar el acceso a los recursos compartidos y evitar problemas de concurrencia.

Compilación y Ejecución:

- Compila el programa y ejecútalo para observar cómo los hilos interactúan y comparten los recursos de manera segura.

Pruebas y Análisis:
Realiza pruebas exhaustivas para verificar que los hilos funcionen de manera sincronizada y que no haya condiciones de carrera ni bloqueos indefinidos.

Documentación:

- Documenta tu código explicando cómo se implementó la sincronización entre los hilos y cómo se evitan los problemas de concurrencia.

Notas adicionales:
Asegúrate de comprender los conceptos de sincronización en programación multihilo y los mecanismos disponibles en el lenguaje de programación que estás utilizando.
Realiza pruebas exhaustivas para detectar y corregir problemas de sincronización antes de finalizar la práctica.

Practica 4. Desarrollar una aplicación con programación concurrente que resuelva un problema real.

Objetivo: El objetivo de esta práctica es aplicar los conceptos de programación concurrente para resolver un problema real mediante el uso de múltiples hilos de ejecución.

Desarrollo

Selección del Problema Real:

- Elije un problema real que requiera la aplicación de la programación concurrente para su solución. Puede ser un problema de procesamiento de datos, gestión de recursos, simulación, optimización, entre otros.

Diseño de la Aplicación:

- Diseña la estructura de la aplicación y define cómo los hilos se utilizarán para abordar el problema. Considera qué recursos se compartirán entre los hilos y cómo se coordinarán.

Implementación de Hilos:

- Crea los hilos necesarios en tu aplicación y programa la lógica de cada uno de ellos. Asegúrate de que los hilos se comuniquen y coordinen según sea necesario para resolver el problema.

Mecanismos de Sincronización:

- Utiliza mecanismos de sincronización, como semáforos, mutex o barreras, para garantizar que los hilos trabajen de manera segura y eviten condiciones de carrera.

Pruebas y Optimización:

- Realiza pruebas exhaustivas de la aplicación con diferentes conjuntos de datos y escenarios para asegurarte de que funcione correctamente y sea eficiente en términos de rendimiento.

Compilación y Ejecución:

- Compila la aplicación y ejecútala en situaciones de prueba para verificar su funcionamiento.

Optimización:

- Identifica oportunidades de optimización y realiza mejoras en el rendimiento de la aplicación, si es necesario.

Documentación:

- Documenta tu código explicando cómo se implementó la programación concurrente en la aplicación y cómo se resuelve el problema real.

Notas adicionales:
Asegúrate de entender bien el problema real que estás abordando y cómo la programación concurrente puede ayudar a resolverlo de manera más eficiente.

Tema 4 - Acceso a datos.

Competencia Específica:

Establece conexiones a diferentes orígenes de datos para su manipulación y visualización de información.

Elabora tablas de entrada, proceso y salida, *IPO Chart*, GUI, diccionarios de datos y de eventos para los casos de estudio planteados.

Practica 1. Desarrollar una aplicación que permita establecer la conexión a un origen de datos.

Objetivo: El objetivo de esta práctica es aprender a desarrollar una aplicación que pueda establecer una conexión a un origen de datos, como una base de datos, un servicio web o cualquier otro recurso de almacenamiento de datos.

Desarrollo

Definición del Origen de Datos:

- Comienza por definir claramente qué tipo de origen de datos vas a utilizar en tu aplicación. Puede ser una base de datos SQL, una base de datos NoSQL, un servicio web, un archivo local, etc.

Configuración del Entorno de Desarrollo:

- Abre tu entorno de desarrollo integrado (IDE) preferido y configura un proyecto nuevo o utiliza uno existente.

Selección de una Biblioteca o Framework:

- Decide qué biblioteca o framework de acceso a datos utilizarás en tu aplicación. Esto dependerá del tipo de origen de datos que hayas definido en el paso 1.

Programación de la Conexión:

- Escribe el código necesario para establecer una conexión al origen de datos. Esto puede incluir la configuración de parámetros de conexión, la autenticación (si es necesaria) y la inicialización de objetos de conexión.

Manejo de Errores:

- Implementa un manejo adecuado de errores para gestionar situaciones en las que la conexión al origen de datos falle. Esto puede incluir la captura de excepciones y la generación de mensajes de error informativos.

Pruebas de Conexión:

- Realiza pruebas de conexión para asegurarte de que la aplicación pueda establecer una conexión exitosa con el origen de datos.

Compilación y Ejecución:

- Compila la aplicación y ejecútala para verificar que la conexión se establece correctamente.

Documentación:

- Documenta tu código explicando cómo se establece la conexión al origen de datos y cualquier configuración importante que deba ser considerada.

Asegúrate de entender los detalles específicos de la biblioteca o framework que estás utilizando para acceder al origen de datos. Cada tecnología puede tener sus propios métodos y configuraciones.

Tema 5 - Programación de dispositivos móviles.

Competencia Específica:

Desarrollar aplicaciones básicas para dispositivos móviles, considerando su entorno operativo.

Elabora tablas de entrada, proceso y salida, *IPO Chart*, GUI, diccionarios de datos y de eventos para los casos de estudio planteados.

Practica 1. Desarrollar un juego para un dispositivo móvil.

Objetivo: El objetivo de esta práctica es aprender a desarrollar un juego para dispositivos móviles, aprovechando las capacidades de la plataforma y brindando una experiencia de juego interactiva y entretenida.

Desarrollo

Idea y Diseño del Juego:

- Comienza por conceptualizar y diseñar tu juego. Define la mecánica del juego, los objetivos, los personajes, los niveles y cualquier elemento importante de la jugabilidad.
- Crea un diseño de interfaz de usuario (UI) que sea atractivo y adecuado para la pantalla del dispositivo móvil.

Elección de la Plataforma y Tecnología:

- Decide en qué plataforma móvil deseas desarrollar tu juego (iOS, Android, etc.) y selecciona la tecnología adecuada, como el desarrollo nativo (por ejemplo, usando Swift para iOS o Java/Kotlin para Android) o el desarrollo multiplataforma (por ejemplo, utilizando un framework como Unity o Flutter).

Desarrollo del Juego:

- Implementa la lógica del juego y crea los niveles, personajes y elementos interactivos según tu diseño.
- Asegúrate de que el juego sea fácil de entender y jugar en una pantalla táctil.

Gráficos y Sonido:

- Crea o adquiere los recursos gráficos y de sonido necesarios para tu juego, como sprites, imágenes de fondo, efectos de sonido y música.
- Integra estos elementos en tu juego de manera que se ajusten al tema y estilo del juego.

Pruebas y Ajustes:

- Realiza pruebas exhaustivas del juego en diferentes dispositivos móviles para asegurarte de que funcione correctamente y que la jugabilidad sea agradable.
- Realiza ajustes en la dificultad y el equilibrio del juego según los resultados de las pruebas.

Optimización y Rendimiento:

- Optimiza el rendimiento del juego para garantizar que funcione de manera fluida y eficiente en dispositivos móviles de diferentes capacidades.

- Considera la administración de recursos y la gestión de memoria para evitar problemas de rendimiento.

Documentación:

- Documenta el proceso de desarrollo del juego, incluyendo las decisiones de diseño, los desafíos enfrentados y las soluciones implementadas.

Asegúrate de cumplir con los requisitos de diseño y rendimiento de la plataforma móvil para la que estás desarrollando.
Considera la monetización si planeas publicar tu juego en una tienda de aplicaciones.

Referencias:

- *Aguilar, L. J. (2010). Programacion en c/c++ java y UML. México: McGraw Hill.*
- *Bell, D. (2011). Java para estudiantes. México: Pearson.*
- *Ceballos, F. J. (2010). JAVA 2: Curso de programación. Madrid: RA-MA.*
- *Dean, J. (2009). Introducción a la programación con Java. México: McGraw Hill.*
- *Deitel, D. y. (2010). Java Cómo Programar. México: Prentice Hall.*
- *Friesen, J. (2011). Java para desarrollo android. España: Anaya Multimedia.*
- *Huddleston, R. (2011). Android para todos. España: Anaya Multimedia.*
- *Lauren Darcey, S. C. (2012). Android 4. Madrid: Anaya Multimedia.*
- *Soriano, J. E. (2011). Android: Programación de dispositivos móviles a través de ejemplos.*
- *México: Marcombo, S.A.*
- *Raynal, Michel. (2012). Concurrent Programming: Algorithms, Principles, and Foundations.*
- *Springer.*