SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MINATITLÁN

INGENIERÍA EN SISTEMAS COMPUTACIONALES

"MANUAL DE PRÁCTICAS "

MATERIA

PROGRAMACIÓN ORIENTADA A OBJETOS

MINATITLÁN, VER. AGOSTO DEL 2023

3.2 ÍNDICE DEL MANUAL DE PRÁCTICAS

ÍNDICE

3.1 INTRODUCCIÓN

El presente manual dará a conocer las prácticas relacionadas con los temas de la materia Programación Orientada a Objetos, los cuales están divididos en 6 prácticas con respecto al temario de la materia:

1) Introducción al paradigma de la programación orientada a objetos
2) Clases y objetos
3) Herencia
4) Polimorfismo
5) Excepciones
6) Flujos y archivos

3.2 JUSTIFICACIÓN

Un Manual de prácticas puede definirse como un compendio de documentos que contemplan una serie de aportes a la práctica científica y social de los alumnos que se encuentren realizando dicha práctica, las cuales también incluyen las normas y procedimientos que orientarán el desempeño del alumno y facilitarán la integración de la teoría con la práctica, en un contexto real de aprendizaje.

Este manual de prácticas está basado según el contenido de “el libro Guía para la elaboración y registro de textos o trabajos académicos”, con el que cuenta el Tecnológico Nacional de México.

El manual de prácticas servirá como apoyo de aprendizaje para los alumnos de la materia de Programación Orientada a Objetos, así como apoyo didáctico para los maestros de dicha materia, ya que se presentarán consejos y sugerencias para dicha realización de las prácticas, también se dará materia de apoyo para estas mismas.

3.3 OBJETIVO GENERAL DEL MANUAL DE PRÁCTICAS

El objetivo general de la programación orientada a objetos (POO) es proporcionar una forma de organizar y estructurar el código de manera que sea más fácil de entender, mantener y reutilizar. La POO se basa en el concepto de "objetos", que

son instancias de clases, y se centra en la encapsulación, la herencia y el polimorfismo como principios fundamentales.

3.4 DESARROLLO

3.4.1 Práctica 1 Introducción al paradigma de la programación orientada a objetos.

3.4.1.1 Objetivo

Comprende y aplica los conceptos del paradigma de programación orientada a objetos para modelar situaciones de la vida real.

3.4.1.2 Introducción

La programación orientada a objetos (POO) es un paradigma de programación que se basa en el concepto de "objetos". Este enfoque se utiliza para diseñar y organizar el código de manera más eficiente y estructurada, al modelar el mundo real como una colección de objetos interconectados. La POO se ha convertido en uno de los paradigmas de programación más populares y ampliamente utilizados en la industria del software debido a su capacidad para abstraer y organizar datos y funcionalidades de manera más intuitiva.

3.4.1.3 Correlación Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 1.1 Elementos del modelo de objetos: clases, objetos, abstracción, modularidad, encapsulamiento, herencia y polimorfismo y 1.2 Lenguaje de modelado unificado: diagrama de clases.

3.4.1.4 Material Y Equipo Necesario

1. Equipo de cómputo

2. Internet

3.4.1.5 Metodología

- Conociendo los puntos obtenidos en cada una de las evidencias del tema 1, portafolio, cuestionario y actualización mostrar la calificación obtenida.

- Conociendo el total de estudiantes inscritos en la asignatura, así como el total de estudiantes que acudieron un día mostrar el porcentaje de asistencia.

- El precio de un producto se muestra en la tienda ya con el IVA incluido, mostrar el precio sin IVA así como el total de IVA.
- Dado un número entero de 2 dígitos muestre las decenas y unidades que lo forman.

3.4.1.6 Sugerencias Didácticas

- Se sugiere que el alumno tenga conocimientos básicos en programación para poder cumplir con todos los ejercicios propuestos.

3.4.1.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.1.8 Bibliografías

- ¿Qué es la Programación Orientada a Objetos (POO)? | EDteam

3.4.2 Práctica 2 Clases y objetos.

3.4.2.1 Objetivo

Aplica los conceptos de clases y objetos en el desarrollo de programas para solución de problemas de acuerdo con el paradigma orientado a objetos.

3.4.2.2 Introducción

Clases

Definición: Una clase es un plano o una plantilla para crear objetos. Especifica los atributos (propiedades) y métodos (comportamientos) que los objetos de esa clase tendrán. Ejemplo: Si estás creando un programa para administrar una biblioteca, podrías tener una clase llamada "Libro". Esta clase podría tener atributos como "título", "autor" y "año de publicación", así como métodos como "prestar" y "devolver".

Características

Atributos: Las variables que almacenan información específica de un objeto.

Métodos: Las funciones que definen el comportamiento de un objeto de esa clase.

Constructor: Un método especial que se llama cuando se crea un objeto de la clase. Se utiliza para inicializar sus atributos.

3.4.2.3 Correlación Con Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 2.1 Declaración de clases: atributos, métodos, encapsulamiento 2.2 Instanciación de una clase 2.3 Referencia al objeto actual 2.4 Métodos: declaración, mensajes, paso de parámetros, retorno de valores 2.5 Constructores y destructores declaración, uso y aplicaciones 2.6 Sobrecarga de métodos 2.7 Sobrecarga de operadores: Concepto y utilidad, operadores unarios y binarios.

3.4.2.4 Material Y Equipo Necesario

1. Equipo de cómputo
2. Internet

3.4.2.5 Metodología

Elabora tablas de entrada, proceso y salida, IPO Chart, diagrama de clases y tabla de los métodos set para los atributos de salida, así como el desarrollo, archivos .java, para los casos de estudio planteados.

- Crear un programa que instancie y use un objeto predefinido por el lenguaje para practicar el envío de mensajes, el uso de parámetros y la recepción de su respuesta. Sugerencia: objeto de clase String.
- Implementar clases para instanciar objetos que modelen sus contrapartes de la vida real usando tipos de datos simples y objetos como parámetros y valores de retorno, así como métodos sin valores de retorno.
- Intercambiar clases de objetos entre compañeros para usar sus miembros con valores o situaciones erróneas que evidencien la necesidad de protegerlos con modificadores de acceso. Modificar el código fuente aplicando los distintos niveles de acceso para experimentar y descubrir (aprender) el impacto de cada uno de ellos.
- Implementar la clase Persona con los atributos nombre y edad; un constructor, un destructor, y al menos el método crecer para mapear el ciclo de vida de una persona con el de un objeto.

- Implementar la clase Calculadora que realice al menos las cuatro operaciones básicas de la aritmética sobrecargando métodos para cada tipo de dato numérico del lenguaje de los parámetros.

- Implementar la clase Matriz que sobrecargue los operadores +, -, * y / para este tipo de dato definido por el usuario.

3.4.2.6 Sugerencias Didácticas

- Se sugiere que el alumno cuente con todos los requisitos previos para poder cumplir con la practica asignada.

3.4.2.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.2.8 Bibliografías

- Clases y Objetos - Programación Orientada por Objetos en Ruby (gitbook.io)

3.4.3 práctica 3 Herencia.

3.4.3.1 Objetivo

Identifica y aplica relaciones de herencia en clases derivadas para reutilizar los miembros de una clase base en el desarrollo de aplicaciones.

3.4.3.2 Introducción

Aquí hay algunos conceptos clave relacionados con la herencia en la POO:

Clase base (superclase): Una clase base es una clase existente que sirve como modelo o plantilla para crear nuevas clases. Contiene atributos y métodos comunes que pueden ser compartidos por sus clases derivadas. Por ejemplo, una clase base podría ser "Vehículo" con atributos como "velocidad" y "color".

Clase derivada (subclase): Una clase derivada es una nueva clase que se crea tomando como base una clase existente. La subclase hereda los atributos y métodos de la superclase y puede agregar sus propios atributos y métodos

específicos. Por ejemplo, una subclase de "Vehículo" podría ser "Coche" con atributos adicionales como "marca" y métodos adicionales como "arrancar".

Herencia de atributos y métodos: La herencia permite que las subclases hereden los atributos y métodos de la superclase. Esto significa que una subclase puede acceder y utilizar los atributos y métodos de su superclase sin necesidad de volver a definirlos. Sin embargo, también puede agregar nuevos atributos y métodos o modificar los existentes según sea necesario.

Método de constructor de la superclase: Cuando se crea una instancia de una subclase, a menudo se llama al constructor de la superclase para inicializar los atributos heredados de la superclase. Esto se hace mediante la palabra clave "super" en muchos lenguajes de programación.

Sobreescritura (override): Las subclases tienen la opción de sobrescribir (override) los métodos heredados de la superclase. Esto significa que una subclase puede proporcionar su propia implementación de un método, reemplazando la implementación de la superclase. Esto es útil cuando una subclase necesita un comportamiento específico diferente al de la superclase.

La herencia es una técnica poderosa en la POO, ya que permite organizar el código de manera jerárquica y modular, promoviendo la reutilización y la extensibilidad del software. Al heredar características de clases base y agregar características específicas en subclases, los programadores pueden crear sistemas más complejos y estructurados de manera más eficiente.

3.4.3.3 Correlación Con Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 3.1 Definición: clase base, clase derivada 3.2 Clasificación: herencia simple, herencia múltiple 3.3 Reutilización de miembros heredados 3.4 Referencia al objeto de la clase base 3.5 Constructores y destructores en clases derivadas 3.6 Redefinición de métodos en clases derivadas.

3.4.3.4 Material Y Equipo Necesario

1. Equipo de cómputo

2. Internet

3.4.3.5 Metodología

Elabora tablas de entrada, proceso y salida, IPO Chart, diagrama de clases y tabla de los métodos set para los atributos de salida, así como el desarrollo, archivos .java, para los casos de estudio planteados.

- Programar una aplicación sobre figuras geométricas que implemente la clase base Figura Geométrica de la cual hereden sus miembros las clases derivadas y que éstas solo especialicen sus características o comportamientos.
- Implementar constructores y destructores a las clases base y derivadas de la aplicación sobre figuras geométricas para experimentar y comprender su funcionamiento cuando está implicada la herencia.

3.4.3.6 Sugerencias Didácticas

- Se sugiere que el alumno tenga los conocimientos previos para poder cumplir con los ejercicios propuestos.

3.4.3.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.3.8 Bibliografías

- **III - Herencia (unam.mx)**

3.4.4 Práctica 4 Polimorfismo

3.4.4.1 Objetivo

Aplica el concepto de polimorfismo para la definición de clases abstractas e interfaces que permitan reutilización de código.

3.4.4.2 Introducción

El concepto de polimorfismo es en realidad algo muy básico. Realmente, cuando estamos aprendiendo Programación Orientada a Objetos (también conocida por sus

siglas POO / OOP) muchos estudiantes nos hacemos un embolado tremendo al tratar de entender el concepto, pero en su base es algo extremadamente sencillo.

Trataremos de explicarlo en este artículo con palabras sencillas, pero para los valientes, aquí va una primera definición que no es mía y que carece de la prometida sencillez. Pero no te preocupes, pues la entiendas o no, luego lo explicaré todo de manera más llena.

Definición: El polimorfismo es una relajación del sistema de tipos, de tal manera que una referencia a una clase (atributo, parámetro o declaración local o elemento de un vector) acepta direcciones de objetos de dicha clase y de sus clases derivadas (hijas, nietas, ...).

3.4.4.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema 4.1 Definición, 4.2 Clases abstractas: definición, métodos abstractos, implementación de clases abstractas, modelado de clases abstractas, 4.3 Interfaces: definición, implementación de interfaces, herencia de interfaces, 4.4 Variables polimórficas (plantillas): definición, uso y aplicaciones y 4.5 Reutilización de código.

3.4.4.4 Material Y Equipo Necesario

1. Equipo de cómputo
2. Internet

3.4.4.5 Metodología

Elabora tablas de entrada, proceso y salida, IPO Chart, diagrama de clases y tabla de los métodos set para los atributos de salida, así como el desarrollo, archivos .java, para los casos de estudio planteados.

Modificar la clase FiguraGeometrica para convertirla en abstracta y programar al menos un método abstracto que todas las clases derivadas deberán implementar con su propio comportamiento.

Programar la interfaz Vehiculo con un conjunto de métodos abstractos que todo vehículo de la vida real debería tener. Programar varias clases que

implementen la interfaz anterior y definan el comportamiento particular de sus métodos.

Especializar la interfaz Vehiculo en al menos dos subinterfaces (VehiculoTerreste o VehiculoAereo) que agreguen comportamientos abstractos que las clases deberán implementar.

3.4.4.6 Sugerencias Didácticas

- Se sugiere que el alumno cuente con los conocimientos básicos para poder realizar los ejercicios propuestos.

3.4.4.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.4.8 Bibliografías

- Polimorfismo en Programación Orientada a Objetos (desarrolloweb.com)

3.4.5 Práctica 5 Excepciones

3.4.5.1 Objetivo

Comprende y aplica las condiciones apropiadas para evitar los errores que pueden interrumpir el flujo normal de ejecución de un programa a través del manejo de excepciones.

3.4.5.2 Introducción

Una excepción es la indicación de que se produjo un error en el programa. Las excepciones, como su nombre lo indica, se producen cuando la ejecución de un método no termina correctamente, sino que termina de manera excepcional como consecuencia de una situación no esperada.

Cuando se produce una situación anormal durante la ejecución de un programa (por ejemplo, se accede a un objeto que no ha sido inicializado o tratamos de acceder a una posición inválida en un vector), si no manejamos de manera adecuada el error que se produce, el programa va a terminar abruptamente su ejecución. Decimos

que el programa deja de funcionar y es muy probable que el usuario que lo estaba utilizando ni siquiera sepa qué fue lo que pasó.

Cuando durante la ejecución de un método el computador detecta un error, crea un objeto de una clase especial para representarlo (de la clase Exception en Java), el cual incluye toda la información del problema, tal como el punto del programa donde se produjo, la causa del error, etc. Luego, "dispara" o "lanza" dicho objeto (throw en inglés), con la esperanza de que alguien lo atrape y decida como recuperarse del error. Si nadie lo atrapa, el programa termina, y en la consola de ejecución aparecerá toda la información contenida en el objeto que representaba el error.

3.4.5.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema, 5.1 Definición, 5.2 Tipos de excepciones, 5.3 Propagación de excepciones, 5.4 Gestión de excepciones: manejo de excepciones, lanzamiento de excepciones y 5.5 Creación y manejo de excepciones definidas por el usuario.

3.4.5.4 Material Y Equipo Necesario

1. Equipo de cómputo
2. Internet

3.4.5.5 Metodología

Elabora tablas de entrada, proceso y salida, IPO Chart, diagrama de clases y tabla de los métodos set para los atributos de salida, así como el desarrollo, archivos .java, para los casos de estudio planteados.

- Programar clases que generen excepciones comunes como referencias nulas o desbordamientos numéricos para estudiar su naturaleza, comportamiento, prevención y lanzamiento.

3.4.5.6 Sugerencias Didácticas

- Se sugiere que el alumno tenga los conocimientos previos para poder realizar los ejercicios propuestos.

3.4.5.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.5.8 Bibliografías

- **Manejo de las Excepciones · Fundamentos de Programación (gitbooks.io)**

3.4.6 Práctica 6 Flujos y archivos

3.4.6.1 Objetivo

Comprende y aplica la clasificación de archivos y operaciones básicas sobre éstos para manipular su información.

3.4.6.2 Introducción

La información que necesita un programa para su función se obtiene mediante una entrada de datos de una fuente que puede ser de tipos muy variados: desde el teclado, un archivo, una comunicación de red, un objeto en internet, etc. Cuando el programa genera los resultados como salida de la ejecución puede hacerlo de muy diversas maneras: en un archivo, en la pantalla, en una impresora, etc.

En java la entrada de los datos se realiza mediante un flujo de entrada. La salida de datos realiza mediante un flujo de salida.

Existen dos tipos de Flujos:

- Los que trabajan con Bytes
- Los que trabajan con Caracteres

Las clases más importantes a tener en cuenta son las siguientes, donde el sangrado de las líneas indica la herencia, es decir, DataInputStream hereda de FilterInputStream que, a su vez, hereda de InputStream.

3.4.6.3 Especificar La Correlación Con El O Los Temas Y Subtemas Del Programa De Estudio Vigente.

Esta actividad corresponde al subtema, 6.1 Definición, 6.2 Clasificación: Archivos de texto y binarios, 6.3 Operaciones básicas y tipos de acceso, 6.4 Manejo de objetos persistentes.

3.4.6.4 Material Y Equipo Necesario

1. Equipo de cómputo

2. Internet

3.4.6.5 Metodología

Elabora tablas de entrada, proceso y salida, IPO Chart, diagrama de clases y tabla de los métodos set para los atributos de salida, así como el desarrollo, archivos .java, para los casos de estudio planteados.

Implementar aplicaciones que almacenen y recuperen información de diferentes tipos de datos simples a través de un archivo de texto para persistir información.

Programar una clase que tome un objeto de cierto tipo y lo persista en un archivo de texto para ser recuperado posteriormente restableciendo el estado que tenía antes de ser persistido (serializarlo).

3.4.6.6 Sugerencias Didácticas

- Se sugiere que el alumno cuente con los requisitos previos para poder cumplir con los ejercicios propuestos.

3.4.6.7 Reporte Del Alumno

El alumno debe de realizar la actividad detallando paso a paso la elaboración de esta, incluyendo capturas, mediante el formato de un reporte de prácticas dando detalle de los resultados obtenidos, así como su conclusión y aprendizajes obtenidos.

3.4.6.8 Bibliografías

- Unidad 6 Flujos y Archivos | Programacion Orientada a Objetos (fjglez90.blogspot.com)