UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

MANUAL DE PRÁCTICAS DE LABORATORIO

Programación Orientada a Objetos

Laboratorio de Programación

Programa Académico	**Ingeniero en Software**
Plan de Estudios	**2021**
Fecha de elaboración	**30/06/2025**
Versión del Documento	

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

Dra. Martha Patricia Patiño Fierro
Rectora

Mtra. Ana Lisette Valenzuela Molina
Encargada del Despacho de la Secretaría General Académica

Mtro. José Antonio Romero Montaño
Secretario General Administrativo

Lic. Jorge Omar Herrera Gutiérrez
Encargado de Despacho de Secretario General de Planeación

Tabla de contenido

INTRODUCCIÓN

Como parte de las herramientas esenciales para la formación académica de los estudiantes de la Universidad Estatal de Sonora, se definen manuales de práctica de laboratorio como elemento en el cual se define la estructura normativa de cada práctica y/o laboratorio, además de representar una guía para la aplicación práctica del conocimiento y el desarrollo de las competencias clave en su área de estudio. Su diseño se encuentra alineado con el modelo educativo institucional, el cual privilegia el aprendizaje basado en competencias, el aprendizaje activo y la conexión con escenarios reales.

Con el propósito de fortalecer la autonomía de los estudiantes, su pensamiento crítico y sus habilidades para la resolución de problemas, las prácticas de laboratorio integran estrategias didácticas como el aprendizaje basado en proyectos, el trabajo colaborativo, la experimentación guiada y el uso de tecnologías educativas. De esta manera, se promueve un proceso de enseñanza-aprendizaje dinámico, en el que los estudiantes no solo adquieren conocimientos teóricos, sino que también desarrollan habilidades prácticas y reflexivas para su desempeño profesional.

Señalar en este apartado brevemente los siguientes elementos según corresponda:

- Propósito del manual
- Justificación de su uso en el programa académico
- Competencias a desarrollar
    - **Competencias blandas:** Habilidades transversales que se refuerzan en las prácticas, como la comunicación, el trabajo en equipo, el uso de tecnologías, etc.
    - **Competencias disciplinares:** Conocimientos específicos del área del laboratorio, incluyendo fundamentos teóricos y habilidades técnicas.
    - **Competencias profesionales:** Aplicación de los conocimientos adquiridos en escenarios reales o simulados, en concordancia con el perfil de egreso del programa.

IDENTIFICACIÓN

Nombre de la Asignatura		Programación orientada a objetos	
Clave	061CP041	Créditos	
Asignaturas Antecedentes	061CP040	Plan de Estudios	2021

Área de Competencia	Competencia del curso
Insertar área de competencia a la que pertenece la asignatura	Diseñar aplicaciones computacionales utilizando el lenguaje de programación, a través del paradigma de programación orientada a objetos, con el fin de apoyar a las organizaciones en la toma de decisiones relacionadas con la gestión de la información de forma innovadora, a través del análisis de problemas y en cumplimiento a los enfoques de calidad establecidos.

Carga Horaria de la asignatura

Horas Supervisadas			Horas Independientes	Total de Horas
Aula	Laboratorio	Plataforma		
3	1	1	2	7

Consignación del Documento

Unidad Académica	Navojoa
Fecha de elaboración	30/06/2025
Responsables del diseño	Mtra. Josefina Ortega Ruíz
Validación	
Recepción	Coordinación de Procesos Educativos

MATRIZ DE CORRESPONDENCIA

Señalar la relación de cada práctica con las competencias del perfil de egreso

PRÁCTICA	PERFIL DE EGRESO
1.Clases y objetos	• Desarrollar software para agilizar procesos y toma de decisiones. • Aplicar soluciones e innovaciones tecnológicas. • Desarrollar soporte y asistencia técnica. • Crear bases de datos para gestión eficiente de la información.
2.Encapsulamiento	• Desarrollar software bajo estándares de calidad. • Optimizar y usar responsablemente los recursos. • Garantizar la seguridad de la información. • Aplicar soluciones tecnológicas con gestión efectiva de la información.
3. Herencia	• Desarrollar software bajo estándares de calidad nacional e internacional. • Aplicar soluciones e innovaciones tecnológicas. • Optimizar recursos tecnológicos. • Mantener vigencia tecnológica con responsabilidad.
4.Interfaz gráfica	• Desarrollar software para agilizar procesos y toma de decisiones. • Aplicar soluciones tecnológicas para automatizar procesos. • Trabajo en equipo y planeación. • Desarrollar soporte y asistencia técnica. • Garantizar la usabilidad y accesibilidad de la información.

NORMAS DE SEGURIDAD Y BUENAS PRÁCTICAS

Reglamento general del laboratorio

Acceso a Laboratorio de Programación

- El acceso a las salas de trabajo será sólo a las horas que no haya clases.
- Deberá entrar sólo con sus documentos de trabajo, el resto de sus cosas deberá dejarlo a la entrada del laboratorio de cómputo.
- No deberá entrar a las salas de trabajo con alimentos, bebidas o fumando.
- Queda estrictamente prohibido introducir animales.

Permanencia

- Antes de trabajar se debe revisar el equipo de cómputo donde fue designado y reportar alguna anomalía al encargado de la sala; si posteriormente se detecta alguna falla no reportada, se le responsabilizará sobre ésta.
- No deberá mover el equipo de cómputo ni mobiliario de su lugar bajo ningún motivo; verifique en su apartado el equipo que va a necesitar para su trabajo.
- El usuario que dañe el equipo de cómputo, sus instalaciones o cambie su configuración del software, incluso protectores de pantalla o colores de ventanas quedará responsabilizado de pagar su costo de reparación o adquisición.
- El comportamiento de todo usuario no debe ir en contra de la moral y las buenas costumbres dentro de las diferentes salas de cómputo.
- El usuario deberá mantener limpia su área de trabajo.
- El uso del equipo de cómputo es exclusivamente para fines didácticos y de investigación, por lo que se prohíbe el uso de juegos, trabajos personales ó de terceros con fines de lucro.
- Solo se permitirá el trabajo individual por computadora, a excepciones de aquellos cursos ó clases impartidas por instructores ó profesores.
- Los usuarios no podrán ingresar ningún tipo de radio, grabadora ó elementos que produzcan ruido y/o elementos magnéticos.
- Los estudiantes deben guardar respeto mantener los canales de comunicación y demás normas establecidas con los monitores y responsables del centro de cómputo y estos para con ellos.
- El acceso a los laboratorios es para el usuario que haya solicitado el servicio, bajo ninguna circunstancia podrán estar dos personas trabajando conjuntamente en una computadora.

Uso adecuado del equipo y materiales

- El equipo periférico que se presta, es solo para uso durante la sesión de trabajo por lo que deberá de devolverlo al término de esta.
- Para asesorías ó dudas con respecto al uso del equipo, debe dirigirse al responsable en turno dentro de la sala donde se encuentre ó bien dirigirse al personal encargado del laboratorio
- El usuario no debe desconectar ni destapar los ratones de las computadoras, las impresoras ó cualquier equipo periférico que se tenga instalado.
- Es obligación del usuario saber operar el equipo de cómputo y periféricos; debe consultar con el personal del servicio social o leer los folletos sobre el uso de periféricos distribuidos para tal efecto; ya que el daño por el mal uso será responsabilidad del usuario.
- En caso de necesitar tóner la impresora notifique al responsable en turno ó bien al encargado del laboratorio, para que el la verifique y éste sea remplazado.

Procedimientos en caso de emergencia

- Evacuación ordenada por rutas señalizadas.
- Uso de extintores solo por personal capacitado.
- Reporte inmediato de accidentes al personal responsable.

RELACIÓN DE PRÁCTICAS DE LABORATORIO POR ELEMENTO DE COMPETENCIA

Elemento de Competencia al que pertenece la práctica	**I**
	Analizar la sintaxis básica del lenguaje de programación, con el fin de identificar mediante el análisis de problema, los elementos necesarios para el desarrollo de aplicaciones en las organizaciones, en cumplimiento con el paradigma orientado a objetos.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 1	Clases y Objetos	Crear clases y objetos para modelar entidades del mundo real, siempre que se requiera estructurar soluciones orientadas a objetos, en el desarrollo de programas en Java, demostrando creatividad y pensamiento estructurado.

Elemento de Competencia al que pertenece la práctica	**II**
	Implementar aplicaciones a través de los conceptos de la programación orientada a objetos, con el fin de resolver problemas en las organizaciones a través del análisis de problemas.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 2	Encapsulamiento	Implementar clases con atributos privados y métodos públicos para proteger el acceso a los datos, siempre que se requiera mantener la integridad de los mismos, en el desarrollo de aplicaciones orientadas a objetos, demostrando responsabilidad en la gestión del código.
Práctica No. 3	Herencia	Diseñar clases derivadas a partir de clases base para reutilizar y extender funcionalidades existentes, siempre que se requiera optimizar el código y evitar redundancias, en el desarrollo de

		aplicaciones orientadas a objetos, mostrando pensamiento lógico y capacidad de abstracción.

Elemento de Competencia al que pertenece la práctica	**III** Determinar elementos de la programación orientada a objetos para el desarrollo de sistemas de información a través del análisis de problemas, con el fin de dar solución a las necesidades de una organización, en apego a los estándares de calidad establecidos.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 4	Interfaz gráfica de usuario	Desarrollar interfaces gráficas de usuario para facilitar la interacción entre el usuario y la aplicación, siempre que se requiera mejorar la usabilidad del sistema, en la creación de software con herramientas como Swing, demostrando enfoque en el usuario y atención al detalle.

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

PRÁCTICAS

NOMBRE DE LA PRÁCTICA	1. Clases y Objetos
COMPETENCIA DE LA PRÁCTICA	Diseñar clases y objetos para modelar soluciones orientadas a objetos, considerando los principios básicos, en el contexto de un entorno de desarrollo en Java, demostrando pensamiento lógico y trabajo colaborativo.

FUNDAMENTO TÉORICO

La programación orientada a objetos (POO) es un paradigma de programación que organiza el software como una colección de objetos que interactúan entre sí. Java es un lenguaje de programación fuertemente orientado a objetos, y su estructura básica gira en torno al uso de clases y objetos.

Un objeto es una instancia concreta de una clase, que representa una entidad del mundo real o conceptual. En la programación orientada a objetos (POO), los objetos son los elementos fundamentales que contienen datos (atributos) y comportamientos (métodos).

Una clase es una plantilla o modelo que se utiliza para crear objetos. Define un conjunto de atributos (datos) y métodos (comportamientos) que los objetos creados a partir de ella tendrán en común. En otras palabras, la clase actúa como un molde o plano que describe cómo será un tipo de objeto, pero no representa un objeto en sí hasta que se crea una instancia (objeto).

Para definir una clase se utiliza la siguiente estructura: Public

class MiClase

{

…

}

Creación de un objeto

Tipo nombreObjeto=new Tipo();

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Lenguaje: Java SE 17 o superior

- IDE recomendado: Eclipse / NetBeans
- JDK: Java Development Kit
- Sistema operativo: Windows, Linux o macOS

PROCEDIMIENTO O METODOLOGÍA

1. Diseño de la clase

- Crear un proyecto y nómbralo Calculadora
- Crear una clase llamada clsOperaciones.
- Dentro de esta clase, declarar cuatro métodos:
  - sumar(double num1, double num2)
  - restar(double num1, double num2)
  - multiplicar(double num1, double num2)
  - dividir(double num1, double num2)

2. Creación de la clase principal y objetos (15 minutos)

- Crear una clase Principal con el método main.
- Crear una instancia de la clase clsOperaciones
- Solicitar al usuario dos números y la operación a realizar.
- Llamar al método correspondiente según la operación.

3. Pruebas y validación

- Ejecutar el programa varias veces con diferentes entradas.
- Validar el funcionamiento correcto de cada operación.

RESULTADOS ESPERADOS

 El programa debe permitir ingresar dos números y realizar operaciones aritméticas básicas (sumar, restar, multiplicar, dividir).

 La salida mostrará correctamente el resultado de la operación seleccionada.

ANÁLISIS DE RESULTADOS
Como resultado de la práctica se logra entender cómo la estructura de clases permite organizar y reutilizar código de forma más eficiente y modular. Además, se observa una mejora en la lógica de programación al aplicar estos conceptos en ejercicios prácticos. Finalmente, esta práctica contribuyó al desarrollo del pensamiento abstracto y lógico, además de fomentar la autonomía en la resolución de problemas de programación mediante el uso correcto de clases y objetos.

CONCLUSIONES Y REFLEXIONES
Al implementar una calculadora básica con clases, se aplicaron conceptos fundamentales de la programación orientada a objetos, lo cual es esencial para el desarrollo de software escalable.”

ACTIVIDADES COMPLEMENTARIAS

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	
Rúbricas o listas de cotejo para valorar desempeño	Rubrica de reporte de práctica
Formatos de reporte de prácticas	Reporte de practica

NOMBRE DE LA PRÁCTICA	2. Encapsulamiento
COMPETENCIA DE LA PRÁCTICA	Aplicar el principio de encapsulamiento para proteger los datos y controlar el acceso a los atributos de una clase, utilizando modificadores de acceso y métodos getters y setters, en el contexto de un proyecto orientado a objetos en Java, fomentando la responsabilidad y la atención al detalle.

FUNDAMENTO TÉORICO

El encapsulamiento es uno de los pilares de la programación orientada a objetos (POO). Consiste en ocultar los datos internos de una clase y permitir el acceso a ellos únicamente a través de métodos públicos llamados getters y setters. Esto mejora la seguridad, evita modificaciones indebidas y hace el código más mantenible.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Lenguaje: Java SE 17 o superior
- IDE recomendado: Eclipse / NetBeans
- JDK: Java Development Kit
- Sistema operativo: Windows, Linux o macOS

PROCEDIMIENTO O METODOLOGÍA

Crear la clase Persona

1. Crear una clase Persona con los siguientes atributos **privados**:
    - String nombre
    - int edad
2. Crear un **constructor** que inicialice ambos atributos.
3. Crear los **métodos públicos** (getters y setters) para acceder y modificar los atributos:
    - getNombre(), setNombre(String nombre)
    - getEdad(), setEdad(int edad)
4. Agregar una validación en setEdad(int edad) para que no se permita una edad negativa.

Parte 2: Crear la clase principal Principal

1. Crear una clase Principal con el método main.
2. Crear un objeto de tipo Persona.
3. Asignar valores a sus atributos usando los **métodos set**.
4. Mostrar los datos usando los métodos **get**.

RESULTADOS ESPERADOS

- [ ] El programa debe compilar sin errores.
- [ ] Los atributos solo se pueden acceder/modificar usando getters y setters.
- [ ] El sistema debe impedir que se asigne una edad negativa.
- [ ] Se deben visualizar correctamente los valores antes y después de la modificación.

ANÁLISIS DE RESULTADOS

- [ ] Comprobar que el encapsulamiento protege los atributos.
- [ ] Verificar que la validación de edad funciona correctamente.
- [ ] Reflexionar sobre cómo los métodos get y set mejoran el control del acceso a los datos.

CONCLUSIONES Y REFLEXIONES

Se comprendió el concepto de encapsulamiento en Java, y cómo este permite proteger los datos internos de una clase. El uso de getters y setters facilita la validación y el control de acceso, lo cual es fundamental para construir aplicaciones seguras y bien estructuradas.

ACTIVIDADES COMPLEMENTARIAS

Problemas o ejercicios adicionales

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	
Rúbricas o listas de cotejo para valorar desempeño	Rubrica de reporte de práctica
Formatos de reporte de prácticas	Reporte de practica

NOMBRE DE LA PRÁCTICA	3.Herencia

COMPETENCIA DE LA PRÁCTICA	Implementar el concepto de herencia para reutilizar código y establecer jerarquías entre clases, empleando la palabra clave extends y respetando los principios de la programación orientada a objetos, en el contexto del desarrollo de aplicaciones en Java, promoviendo el pensamiento analítico y la adaptabilidad.

FUNDAMENTO TÉORICO

La herencia es un principio de la programación orientada a objetos que permite a una clase (subclase o clase hija) heredar atributos y métodos de otra clase (superclase o clase padre). Esto promueve la reutilización del código y facilita el mantenimiento y la expansión de programas.

En Java, se usa la palabra clave extends para indicar que una clase hereda de otra.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Lenguaje: Java SE 17 o superior
- IDE recomendado: Eclipse / NetBeans
- JDK: Java Development Kit
- Sistema operativo: Windows, Linux o macOS

PROCEDIMIENTO O METODOLOGÍA

Crear la clase base Persona

1. Crear una clase Persona con los atributos:
    - nombre (String)
    - edad (int)
2. Definir un constructor que inicialice estos atributos.
3. Crear un método mostrarDatos() que imprima los datos de la persona.

Crear la subclase Empleado

1. Crear una clase Empleado que **herede** de Persona usando extends.
2. Agregar un atributo adicional:
    - salario (double)
3. Definir un constructor que reciba nombre, edad y salario.
4. Sobrescribir el método mostrarDatos() para mostrar también el salario.

Clase Principal

1. Crear una clase Principal con el método main.
2. Instanciar un objeto de tipo Empleado.
3. Llamar al método mostrarDatos() para mostrar todos los datos.

RESULTADOS ESPERADOS

- El programa debe compilar y ejecutarse sin errores.
- El método mostrarDatos() de la clase Empleado debe mostrar el nombre, la edad y el salario.
- Se debe observar claramente que la subclase hereda atributos y métodos de la superclase.

ANÁLISIS DE RESULTADOS

- Se comprueba que la clase Empleado hereda los atributos nombre y edad de Persona.
- El método super.mostrarDatos() permite reutilizar el código de la clase base.
- La herencia permite extender clases sin repetir código, facilitando la escalabilidad.

CONCLUSIONES Y REFLEXIONES

Se comprende el concepto de herencia como un mecanismo fundamental de la programación orientada a objetos que permite crear nuevas clases a partir de otras existentes, promoviendo la reutilización de código y reduciendo la redundancia.

El uso de la palabra clave extends se usa para establecer la relación entre la clase padre y la clase hija.

Se reforzó el pensamiento lógico y la capacidad de abstracción, al identificar qué atributos y métodos debían estar en la clase base y cuáles debían pertenecer a la clase hija.

ACTIVIDADES COMPLEMENTARIAS

Problemas o ejercicios adicionales

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	De acuerdo a rúbrica
Rúbricas o listas de cotejo para valorar desempeño	Solución Individual de Ejercicios
Formatos de reporte de prácticas	Reporte de practica

NOMBRE DE LA PRÁCTICA	Interfaz gráfica usando Swing
COMPETENCIA DE LA PRÁCTICA	Desarrollar interfaces gráficas de usuario para facilitar la interacción entre el usuario y la aplicación, siempre que se requiera mejorar la usabilidad del sistema, en la creación de software con herramientas como Swing, demostrando enfoque en el usuario y atención al detalle.

FUNDAMENTO TÉORICO

Java proporciona la biblioteca Swing para construir interfaces gráficas (GUI). Swing ofrece componentes como JFrame, JLabel, JButton, JTextField, etc., que permiten crear ventanas interactivas.

A diferencia de la consola, las interfaces gráficas ofrecen una mejor experiencia de usuario y permiten desarrollar aplicaciones visualmente más amigables.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Computadora con JDK instalado
- IDE (NetBeans, Eclipse)
- Librería Swing (incluida en el JDK)

PROCEDIMIENTO O METODOLOGÍA

Crear la estructura de la ventana

1. Crear un proyecto llamado Calculadora
2. Agregar un Jframe Form al proyecto.

Crear una clase llamada clsOperaciones.

Dentro de esta clase, declarar cuatro métodos:

1. sumar(double num1, double num2)
2. restar(double num1, double num2)
3. multiplicar(double num1, double num2)
4. dividir(double num1, double num2)

Agregar componentes al JFrame Form

3. Agregar dos JTextField para ingresar los números.
4. Agregar cuatro JButton para realizar las operaciones: sumar, restar, multiplicar y dividir.
5. Agregar un JTextField para mostrar el resultado.

Parte 3: Programar los eventos

6. Programar cada botón del Jframe Form para invocar el método correspondiente de la clase clsOperaciones.
7. Validar que los campos no estén vacíos y que se puedan convertir a números.
8. Verificar que el resultado arrojado sea correcto.

RESULTADOS ESPERADOS

- Se muestra una ventana con dos campos de texto para introducir los números , cuatro botones para ejecutar las operaciones(Sumar, Restar, Dividir,Multiplicar) y un campo de texto para mostrar el resultado.

- Al ingresar dos números y presionar un botón, se muestra el resultado correcto.

- Se valida la división por cero y la entrada de datos no numéricos.

ANÁLISIS DE RESULTADOS
La interfaz gráfica mejora la experiencia del usuario. La separación entre componentes y lógica mejora la organización del código.

CONCLUSIONES Y REFLEXIONES
Esta práctica permite conocer y aplicar los componentes básicos de Swing para desarrollar interfaces gráficas simples en Java. Se logra crear una aplicación funcional que interactúa con el usuario de manera visual.

ACTIVIDADES COMPLEMENTARIAS
Problemas o ejercicios adicionales

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	
Rúbricas o listas de cotejo para valorar desempeño	Rubrica de reporte de práctica
Formatos de reporte de prácticas	Reporte de práctica

FUENTES DE INFORMACIÓN

1. Blasco, F. (2019). Programación orientada a objetos en Java. Ediciones de la U. https://elibro.net/es/lc/ues/titulos/127125
2. Ceballos, F. J. (2015). Java 2: lenguaje y aplicaciones. RA-MA Editorial. https://elibro.net/es/lc/ues/titulos/62458
3. Flórez , H. A. (2012). Programación orientada a objetos usando java. Ecoe Ediciones. https://elibro.net/es/lc/ues/titulos/69236
4. Moreno, J. (2015) Programación orientada a objetos.: RA-MA Editorial. https://elibro.net/es/ereader/ues/106461?page=1
5. Oviedo, E. (2015). Lógica de programación orientada a objetos. Ecoe Ediciones. https://elibro.net/es/lc/ues/titulos/70431


UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu