Programación Orientada a Objetos

Guía para el profesor
Clave LSTI2307

TECMILENIO

Contenido

Datos generales

Nombre del certificado: Programación Orientada a Objetos

Nivel: Profesional

Modalidad: Presencial

Clave: **LSTI2307**

Competencia global

Aplica los conceptos de orientación a objetos para resolver problemas, utilizando un lenguaje de programación que emplee dicho paradigma.

Competencias esenciales

- Pensamiento crítico
- Resolución de problemas
- Capacidad de análisis
- Capacidad de trabajo en equipo
- Creatividad

Introducción

¡Bienvenido a esta experiencia educativa! En ella, te sumergirás en el universo de la programación orientada a objetos (POO) en lenguaje Java.

La POO es una metodología de programación que ha revolucionado la forma en que se desarrollan aplicaciones y sistemas de *software*, ya que se basa en la idea de que los programas deben estar organizados en torno a "objetos", es decir, a unidades autónomas que encapsulan datos y funcionalidades. Esta metodología se ha convertido en el fundamento de muchas aplicaciones que utilizas diariamente, desde aquellas empleadas en móviles y software de oficina hasta sistemas de control de tráfico aéreo y aplicaciones médicas.

Aprender POO te permitirá comprender y crear sistemas de software más efectivos, mantenibles y escalables; asimismo, te brindará una ventaja competitiva en el mundo laboral, pues muchas empresas buscan programadores con habilidades en POO para desarrollar software de alta calidad.

Por otra parte, Java es uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo, caracterizado por su portabilidad, seguridad y versatilidad, de tal manera que se convierte en una elección ideal para desarrolladores de software. Esta experiencia educativa se centra en enseñarte a programar en Java, desde los conceptos básicos hasta los más avanzados.

A medida que adquieras experiencia en el uso de este lenguaje, te prepararás para abordar proyectos de desarrollo de software en el mundo real, así como para colaborar con equipos de programadores en diversas disciplinas. Con una base sólida en POO con Java, enfrentarás desafíos, resolverás problemas y ganarás experiencia práctica.

Al finalizar tu viaje de aprendizaje, estarás listo para enfrentar proyectos complejos y para contribuir de manera significativa en cualquier equipo de desarrollo de software. Finalmente, es importante mencionar que esta experiencia educativa te brindará los conocimientos necesarios para presentar con confianza el examen final de Java Fundamentals, de Oracle, el cual te abrirá puertas en el mundo laboral y fortalecerá tu carrera profesional.

Metodología

El modelo académico **MAPS** se caracteriza por ser modular, apilable y personalizable con un enfoque flexible y centrado en el estudiante. Implementamos técnicas didácticas que buscan no solo la adquisición de conocimientos teóricos, sino también la aplicación práctica y el desarrollo de competencias profesionales altamente valoradas por los empleadores. A continuación, se detallan las técnicas didácticas y características principales de nuestro modelo académico.

Técnicas didácticas

Aprendizaje basado en retos. El alumno demuestra la adquisición de los conocimientos y los aplica por medio de retos propuestos.

Aprendizaje basado en proyectos. El alumno demuestra la adquisición de los conocimientos y los aplica en la práctica, por medio de proyectos que impacten de manera positiva a las organizaciones.

Aula invertida*. Esta metodología promueve el autoestudio fuera de las clases, para que, una vez que los alumnos se encuentren en el aula virtual, se promueva la interacción, la construcción conjunta del conocimiento, la generación de ideas y el desarrollo de las competencias, gracias al acompañamiento de docentes expertos.

El **aprendizaje basado en retos** se implementa del primero al quinto semestre, el **aprendizaje basado en proyectos** se aplica del sexto semestre en adelante y la metodología de **aula invertida** está presente en todos los certificados.

*En las Semanas de Desarrollo Integral (SeDI) y los certificados de idioma, solamente aplica la metodología de aula invertida.

Características

1. Certificados
   a. El modelo está formado por certificados de especialidad, los cuales buscan el desarrollo y la adquisición de competencias requeridas por los principales empleadores de nuestro país a través del aprendizaje activo.
   b. Todos los certificados son creados en alianza y colaboración con empresas de prestigio nacional e internacional y/o con expertos que cuentan con conocimiento técnico actual y académico que se requiere en las distintas industrias, con lo que se garantiza el desarrollo de competencias profesionales.
   c. En cada período, el estudiante lleva un máximo de dos certificados simultáneos, con ello los estudiantes tienen la oportunidad de profundizar más en cada tema. Esto es especialmente valioso en cursos que requieren una comprensión detallada de teorías complejas, aplicaciones prácticas y habilidades analíticas avanzadas.

2. Duración
   Licenciatura dependiendo del formato elegido. Los programas ejecutivos se cursan en 15 bimestres, mientras que los programas semestrales se cursan en 8 semestres. Ambos están compuestos por los mismos certificados en sus mapas curriculares, lo que permite transitar entre ambas modalidades dependiendo de las necesidades de los estudiantes.

3. Flexibilidad
   Este modelo promueve la participación de los estudiantes al permitirles personalizar su experiencia de aprendizaje de acuerdo con sus intereses y necesidades individuales. Esta personalización no solo facilita un mayor compromiso y motivación, sino que también prepara a los estudiantes para enfrentar retos específicos de su futuro campo profesional, aumentando así su empleabilidad y éxito académico.

4. Credenciales apilables
   La idea atrás de estas credenciales es proveer un esquema de capacitación y aprendizaje para los aprendedores, de tal forma que puedan moverse rápidamente en el proceso educativo, aprendiendo habilidades que son aplicables en el trabajo. Las credenciales, por lo tanto, pueden ser apiladas para cumplir con el estándar de un programa de grado tradicional.

5. Insignias digitales
   Las insignias digitales permiten documentar la educación de los estudiantes, así como sus logros. Una de las ventajas de las insignias digitales es que, a través de la metadata, se pueden obtener los detalles de las competencias adquiridas, la institución que otorga la insignia, así como un reconocimiento visual que puede ser compartido en redes sociales o redes profesionales.

6. Diferenciadores del modelo
   a. Certificados de lengua extranjera: se cuenta con certificados para adquirir o reforzar el dominio de lengua extranjera y con certificados impartidos en una lengua extranjera específicos de la disciplina, todo con el objetivo de atender las demandas de los empleadores.
   b. Semanas de Desarrollo Integral: unidades de aprendizaje transversal, diseñadas para vivir una experiencia inmersiva, desarrollando las competencias humanas, profesionales y de bienestar.
   d. Períodos de *Skilling*: período complementario donde el alumno puede llevar a cabo actividades que suman a su formación académica. Son opcionales y personalizadas, ya que el estudiante las selecciona con base en sus intereses profesionales y personales.
   e. Estancia empresarial al final del programa de estudios: los estudiantes tendrán a su disposición tres opciones en función de la estancia empresarial que vayan a realizar, entre las cuales se encuentran: gestión de proyectos, emprendimiento y desarrollo sostenible.

Bibliografía y software

Bibliografía de apoyo

• Alankus, G., De Brito, R., Ahamed, B., Isola, V., y Obare, M. (2019). *Java Fundamentals. A fast-paced and pragmatic introduction to one of the world's most popular programming languages*. Reino Unido: Packt. Recuperado de https://eds.p.ebscohost.com/eds/ebookviewer/ebook/ZTAwMHh3d19fMjA4NzU4NV9fQU41?sid=dcb43961-1574-45d1-9e2e-d868115c9b9b@redis&vid=7&format=EB&rid=1

Software

• Oracle. (s.f.). *Java Downloads*. Recuperado de https://www.oracle.com/java/technologies/downloads/

• The Apache Software Foundation. (s.f.). *Downloading Apache NetBeans 19*. Recuperado de https://netbeans.apache.org/download/nb19/index.html

Evaluación

La evaluación es una combinación de los siguientes elementos:

- Actividades que retoman el contenido conceptual de los temas de la semana.
- Proyecto, dividido en dos fases, con el que el participante demostrará las habilidades y conocimientos requeridos para acreditar el certificado.
- Presentación del proyecto.

A continuación, puedes revisar el detalle de la evaluación:

Semana	Evaluable	Ponderación
1	Actividad I	6%
2	Actividad II	6%
3	Avance del proyecto	30%
4	Actividad III	6%
5	Actividad IV	6%
6	Actividad V	6%
7	Entrega final del proyecto	35%
8 Semana de Assesment	Presentación del proyecto	5%

Actividades y fases del proyecto

El avance y entrega final del proyecto se han diseñado para realizarse de manera individual.

Como una forma de promover el dinamismo y la interacción de los participantes en distintos formatos, durante las sesiones, el profesor alterna intervenciones individuales, plenarias y grupales que enriquecen tus puntos de vista y, al mismo tiempo, te dan la oportunidad de presentar tus ideas y posturas en torno a los temas de clase.

El resultado del avance y entrega final del proyecto realizadas deberá entregarse a través de la plataforma tecnológica para su revisión y evaluación por parte del docente. Es muy importante que revises el esquema de evaluación y los criterios que utilizará el docente para otorgarte una calificación. Lo anterior con la intención de que desde el inicio de la semana tengas claro el nivel de complejidad y esfuerzo que requieres para realizar las entregas semanales y garantizar tu éxito.

En caso de tener dudas sobre el avance y entrega final del proyecto o contenido, puedes contactar a tu docente a través de los medios que te indique.

Calendario de entregas

Semana	Evaluable
1	Actividad I
2	Actividad II
3	Avance del reto
4	Actividad III
5	Actividad IV
6	Actividad V
7	Entrega final del proyecto
8 Semana de Assesment	Presentación del proyecto

Temario

Tema 1. Fundamentos de programación orientada a objetos
- 1.1 Introducción a la POO
- 1.2 Conceptos clave
- 1.3 Atributos y comportamientos de objetos
- 1.4 Características de POO

Tema 2. Modelado y UML
- 2.1 Ingeniería de *software*
- 2.2 Fundamentos de UML
- 2.3 Relaciones entre clases en UML
- 2.4 Diagramas de clases

Tema 3. Lenguaje orientado a objetos
- 3.1 Elementos básicos del lenguaje Java
- 3.2 Cómo obtener y utilizar Java
- 3.3 Estructura de un programa
- 3.4 Escritura y obtención de datos (operaciones I/O estándar)

Tema 4. Tipos de datos y operadores
- 4.1 Tipos de datos
- 4.2 Declaración de variables y uso de literales
- 4.3 Operadores booleanos, aritméticos y de asignación
- 4.4 Conversión de tipos de datos (*casting*)

Tema 5. *Strings* (cadenas de texto)
- 5.1 Concepto de string
- 5.2 Creación de una instancia de string
- 5.3 Referencias de string
- 5.4 Operaciones con strings

Tema 6. Estructuras de selección
- 6.1 Introducción a las estructuras de selección
- 6.2 La sentencia if
- 6.3 La sentencia if-else
- 6.4 La sentencia switch

Tema 7. Estructura de repetición
- 7.1 Conceptos básicos de las estructuras de repetición
- 7.2 La sentencia while
- 7.3 La sentencia do-while
- 7.4 La sentencia for

Tema 8. Arreglos
- 8.1 Declaración de arreglos
- 8.2 Recorrido de arreglos
- 8.3 Arreglos de objetos
- 8.4 Arreglos bidimensionales

Tema 9. Manejo de errores
- 9.1 Tipos de errores
- 9.2 Categorías de las excepciones
- 9.3 Manejo de excepciones controladas (*checked*)
- 9.4 Lanzamiento y captura de excepciones (*try/catch*)

Tema 10. Métodos y parámetros
- 10.1 Definición y uso de métodos
- 10.2 Parámetros en métodos
- 10.3 Métodos con valor de retorno
- 10.4 Sobrecarga de métodos

Tema 11. Clases y objetos
- 11.1 Elementos de una clase
- 11.2 Declaración de objetos
- 11.3 Objetos como parámetros
- 11.4 Clases y métodos estáticos

Tema 12. Métodos avanzados
- 12.1 Métodos estáticos
- 12.2 Métodos recursivos
- 12.3 Paso de objetos como parámetros
- 12.4 Métodos *final* y *abstract*

Tema 13. Constructores y destructor
- 13.1 Características de constructores
- 13.2 Tipos de constructores
- 13.3 Invocación de métodos
- 13.4 El destructor en POO

Tema 14. Relaciones avanzadas entre clases
- 14.1 Composición y agregación
- 14.2 Objetos contenedores
- 14.3 Relación de dependencia
- 14.4 Asociación entre objetos

Tema 15. Abstracción y modelado
- 15.1 Concepto de abstracción
- 15.2 Aplicación práctica
- 15.3 Modelado de abstracciones
- 15.4 Diagramas de secuencia

Tema 16. Encapsulamiento y modificadores
- 16.1 Definición y significado
- 16.2 Implementación efectiva
- 16.3 Modificadores de acceso
- 16.4 Prácticas de encapsulamiento

Tema 17. Fundamentos de herencia
- 17.1 Introducción a la herencia
- 17.2 Clases base y derivadas
- 17.3 Ventajas de la herencia
- 17.4 Ejemplos de herencia

Tema 18. Modificadores y herencia
- 18.1 Modificadores de acceso y herencia
- 18.2 Constructores en herencia
- 18.3 Herencia múltiple
- 18.4 Consideraciones y mejores prácticas

Tema 19. Conceptos de polimorfismo
- 19.1 Introducción al polimorfismo
- 19.2 Métodos polimórficos
- 19.3 Interfaces y polimorfismo
- 19.4 Ejemplos de polimorfismo

Tema 20. Conversión de objetos
- 20.1 Conversión entre objetos de clase base y derivada
- 20.2 *Casting* y conversiones explícitas
- 20.3 Casos de uso de conversiones
- 20.4 Precauciones en la conversión

Preguntas más frecuentes

¿En dónde o a quién le reporto un error detectado en el contenido?

Lo puedes reportar a través del botón “Mejora tu curso”, también puedes compartir sugerencias para el contenido y actividades del certificado.

¿Quién me informa de la cantidad de sesiones y el tiempo de cada sesión en las semanas?

El coordinador docente te debe proporcionar esta información.

¿Tengo que capturar las calificaciones en Banner y en la plataforma educativa?

Sí, es importante que captures las calificaciones en la plataforma para que los participantes estén informados de su avance y reciban retroalimentación de parte tuya de todo lo que realizan en esta experiencia educativa. En Banner es el registro oficial de las calificaciones de los participantes.

Recomendaciones para la explicación de los temas, actividades y proyecto.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 1.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar con una explicación sobre la esencia de la POO como un paradigma basado en la creación de objetos; después, ilustrar las características clave de estos últimos, como atributos y comportamientos, utilizando ejemplos concretos.

- Desglosar los conceptos fundamentales de la POO, incluyendo clases, objetos, encapsulación, herencia y polimorfismo; además, se recomienda utilizar ejemplos simples para ayudar a los aprendedores a comprender cada una de estas nociones. También es importante destacar cómo la POO permite modelar el mundo real de manera más natural y organizada.

- Resaltar la importancia de la reutilización de código y de la modularidad en la POO; para ello, se sugiere explorar detalladamente cómo los objetos poseen atributos que representan características y comportamientos, los cuales definen lo que pueden hacer.

- Contrastar la POO con otros paradigmas de programación para resaltar sus diferencias y beneficios.

- Destacar la importancia de la planificación y el diseño en la POO.

- Animar a los aprendedores a que revisen el capítulo 1 del libro de apoyo, “Chapter 1. Introduction to Java”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 2.

Al profesor impartidor, se le recomienda lo siguiente:

- Presentar el concepto de ingeniería de software como una disciplina que se ocupa de diseñar, construir y mantener software de manera sistemática y eficiente; además, explicar por qué dicho campo resulta indispensable en el desarrollo de aplicaciones informáticas.

- Introducir el lenguaje de modelado unificado (UML) como una herramienta esencial en la ingeniería de software para modelar y representar sistemas; en este sentido, se sugiere explicar los diagramas de clases y cómo se utilizan para representar la estructura y las relaciones de las clases en un sistema.

- Explorar detalladamente las relaciones entre clases en UML, como la asociación, agregación y composición; asimismo, ilustrar estas relaciones con ejemplos concretos, es decir, que muestren cómo se representan gráficamente en diagramas de clases.

- Enseñar a los aprendedores a crear y leer diagramas de clases en UML, así como demostrarles de qué manera se representan las clases, atributos, métodos y relaciones en estos diagramas.

- Realizar ejercicios prácticos para que los aprendedores ejerciten la creación y lectura de diagramas de clases.

- Animar a los aprendedores a que revisen el capítulo 1 del libro de apoyo, “Chapter 1. Introduction to Java”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 3.

Al profesor impartidor, se le recomienda lo siguiente:

- Introducir los conceptos fundamentales de Java, como clases, objetos, métodos y atributos, para, después, explicar cómo se define una clase en Java y cómo se crean objetos a partir de ella.

- Mostrar ejemplos de métodos y atributos en clases Java.

- Explicar a los aprendedores cómo pueden obtener e instalar el entorno de desarrollo de Java (IDE) en sus sistemas; asimismo, se sugiere guiarlos a través de la creación de un proyecto Java simple y de la escritura de su primer programa.

- Enseñar la estructura básica de un programa Java, incluyendo la función **main()**, declaraciones de variables y la ejecución secuencial de instrucciones.

- Demostrar cómo se comenta el código y cuál es la importancia de las observaciones.

- Introducir las operaciones de entrada/salida estándar en Java para que los aprendedores puedan interactuar con el usuario y mostrar resultados.

- Proporcionar ejemplos de cómo se leen los datos del usuario y mostrar mensajes en la consola.

- Animar a los aprendedores a que revisen el capítulo 1 del libro de apoyo, “Chapter 1. Introduction to Java”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 4.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar la clase con una explicación sobre qué son los tipos de datos en programación y por qué son importantes.

- Introducir los tipos de datos primitivos en Java, como ***int***, ***double***, ***char***, ***boolean***, etcétera.

- Proporcionar ejemplos de cómo declarar variables con diferentes tipos de datos y asignarles valores; además, hay que aclarar cuál es la sintaxis adecuada para cada variable.

- Explicar el concepto de literales y cómo se utilizan para asignar valores a las variables, así como proporcionar ejemplos de declaraciones de variables y asignación de literales.

- Introducir los diferentes operadores (booleanos, aritméticos y de asignación) y comentar cuándo se utiliza cada uno.

- Explicar la conversión de tipos de datos en Java, incluyendo la implícita y explícita.

- Proporcionar ejemplos de cómo realizar casting de tipos de datos y mencionar en qué casos es necesario.

- Animar a los aprendedores a que revisen el capítulo 2 del libro de apoyo, “Chapter 2. Variables, Data Types, and Operators”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 5.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar con una lluvia de ideas para conocer qué entienden los aprendedores por una cadena de texto (string) en programación y por qué creen que es importante; al terminar con este ejercicio, hay que brindar una explicación del concepto.

- Introducir el concepto de secuencia de caracteres y cómo se utiliza para representar texto.

- Enseñar a los aprendedores a declarar y crear objetos de cadena de texto en Java; después, se sugiere mostrar las diferentes formas de inicializar una cadena de texto, ya sea mediante literales o mediante el constructor **String()**.

- Hablar sobre cómo funcionan las referencias de cadena de texto en Java, incluyendo el concepto de inmutabilidad.

- Explicar de qué manera se pueden concatenar cadenas de texto, cómo funcionan los operadores de concatenación e introducir las operaciones comunes que se pueden realizar con cadenas de texto, como longitud, búsqueda de subcadenas, reemplazo y división; además, se sugiere proporcionar ejemplos de cómo se realizan estas operaciones con los métodos de la clase String en Java.

- Animar a los aprendedores a que revisen el capítulo 6 del libro de apoyo, “Chapter 6. Data Structures, Arrays, and Strings”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 6.

Al profesor impartidor, se le recomienda lo siguiente:

- Explicar qué son las estructuras de selección y por qué son fundamentales en programación; además, es importante destacar cómo las estructuras de selección permiten tomar decisiones en función de condiciones lógicas.

- Explicar detalladamente cómo funciona la sentencia ***if*** en Java y proporcionar ejemplos de cómo usarla para realizar acciones condicionales.

- Introducir la sentencia ***if-else*** y mostrar cómo se utiliza para manejar casos alternativos; adicionalmente, se aconseja proporcionar ejemplos concretos de situaciones en las que dicha sentencia es útil.

- Describir la sentencia ***switch*** y de qué manera se utiliza para manejar múltiples casos; luego de comprender su empleo, se sugiere proporcionar algunos ejemplos de cómo implementar un **switch** en Java.

- Animar a los aprendedores a que revisen el capítulo 3 del libro de apoyo, “Chapter 3. Control Flow”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 7.

Al profesor impartidor, se le recomienda lo siguiente:

- Introducir las estructuras de repetición y explicar su importancia en la automatización de tareas.

- Enseñar cómo las estructuras de repetición permiten ejecutar un bloque de código múltiples veces.

- Explicar la sentencia **while** en Java y cómo se utiliza para repetir un bloque de código mientras se cumple una condición; asimismo, resulta esencial proporcionar ejemplos prácticos de cómo implementar dicha sentencia.

- Describir la sentencia **do-while** y cómo se diferencia de **while**; además, se recomienda ilustrar con ejemplos que demuestren cuándo es apropiado usarla.

- Introducir la sentencia **for** y explicar cómo simplifica las estructuras de repetición; después, se aconseja proporcionar ejemplos de bucles **for** y cómo se controlan con variables de iteración.

- Animar a los aprendedores a que revisen el capítulo 3 del libro de apoyo, “Chapter 3. Control Flow”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 8.

Al profesor impartidor, se le recomienda lo siguiente:

- Introducir el concepto de arreglos y su importancia en la gestión de datos; de igual forma, es importante explicar cómo declarar arreglos en Java, incluyendo su tamaño y tipo de dato.

- Proporcionar algunos ejemplos de declaraciones de arreglos y explicar cómo se inicializan.

- Enseñar cómo se accede a los elementos individuales de un arreglo mediante índices; para ello, se sugiere describir cómo funcionan estos últimos en los arreglos y de qué manera se pueden evitar errores de desbordamiento.

- Brindar algunos ejemplos de bucles para recorrer arreglos y realizar operaciones en sus elementos.

- Explicar que los arreglos no se limitan a tipos de datos primitivos, ya que pueden contener objetos; además, es necesario proporcionar ejemplos sobre cómo declarar, inicializar y utilizar arreglos de objetos.

- Ilustrar la ventaja de utilizar arreglos de objetos en situaciones específicas.

- Introducir el concepto de arreglos bidimensionales, los cuales se definen como “arreglos de arreglos”.

- Explicar cómo declarar y trabajar con arreglos bidimensionales en Java, además de proporcionar ejemplos de matrices bidimensionales y de qué manera acceder a sus elementos.

- Animar a los aprendedores a que revisen el capítulo 6 del libro de apoyo, “Chapter 6. Data Structures, Arrays, and Strings”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 9.

Al profesor impartidor, se le recomienda lo siguiente:

- Comentar los tres principales tipos de errores en el desarrollo de software: de sintaxis, lógicos y excepciones. Además, se recomienda ilustrar cada tipo con ejemplos claros y prácticos para facilitar la comprensión.

- Enfatizar la importancia de la revisión exhaustiva del código para detectar y corregir errores de sintaxis antes de la compilación, así como la necesidad de realizar pruebas rigurosas para identificar y resolver defectos lógicos, los cuales pueden pasar desapercibidos en el proceso de desarrollo. Asimismo, se sugiere promover la resolución de problemas prácticos en equipo, donde los aprendedores analicen y corrijan ejemplos de lógicas comunes.

- Explicar claramente la diferencia entre las excepciones controladas y no controladas, así como las estrategias para manejarlas; en este caso, resulta fundamental ilustrar casos de excepciones comunes, como **IndexOutOfBoundsException**, **NullPointerException** y **ArithmeticException**, en conjunto con ejemplos prácticos que demuestren cómo administrarlas en el código.

- Animar a los aprendedores a que revisen el capítulo 9 del libro de apoyo, “Chapter 9. Expection Handling”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 10.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar la explicación con una lluvia de ideas donde los alumnos compartan sus impresiones sobre los métodos; finalizado el ejercicio, se recomienda destacar su relevancia como bloques de código reutilizables que estructuran los programas en Java. Asimismo, se aconseja resaltar su funcionalidad, enfocándose en su estructura básica: modificadores de acceso, tipo de retorno, nombre, parámetros y cuerpo.

- Hacer hincapié en la forma como se nombran los métodos, es decir, usando las convenciones de Camel Case.

- Explicar detalladamente en qué consiste la sobrecarga de métodos, ya que posiblemente se trata del subtema que genera más dudas; para esto, se recomienda mostrar algunos ejemplos y situaciones donde la sobrecarga de métodos resulta útil.

- Animar a los aprendedores a que revisen el capítulo 4 del libro de apoyo, “Chapter 4. Object-Oriented Programming”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 11.

Al profesor impartidor, se le recomienda lo siguiente:

- Enfatizar y explicar detalladamente los elementos que conforman una clase (atributos y métodos), resaltar su relación con los objetos y exponer cómo modelan la solución de un problema.

- Realizar ejercicios interactivos que impliquen la creación de clases simples y la instanciación de objetos a partir de ellas; además, se sugiere incluir dinámicas en equipo para diseñar clases y objetos basados en situaciones del mundo real, así como elaborar mapas mentales o diagramas de flujo para visualizar la relación entre clases y objetos. Estas herramientas son eficaces para aclarar dudas que pueden surgir al respecto de la relación entre clases y objetos, o bien, acerca de la manipulación de atributos y métodos.

- Explicar detalladamente que las clases deben incluir métodos de acceso (**getters**) y establecedores (**setters**) para leer y asignar valores a los atributos; de igual manera, se recomienda ilustrar con ejemplos prácticos.

- Describir los conceptos de encapsulación, herencia, uso de clases y métodos estáticos.

- Animar a los aprendedores a que revisen el capítulo 4 del libro de apoyo, “Chapter 4. Object-Oriented Programming”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 12.

Al profesor impartidor, se le recomienda lo siguiente:

- Resaltar la importancia de los métodos estáticos, enfatizando su utilidad en la creación de rutinas y funciones independientes de objetos.

- Proporcionar ejemplos claros y concretos para mostrar cómo se definen y utilizan los métodos estáticos; además, hay que aclarar los posibles riesgos al abusar de ellos.

- Involucrar a los aprendedores en ejercicios prácticos que incluyan la creación de algoritmos recursivos, por ejemplo, la lectura de sistemas de archivos con carpetas y subcarpetas.

- Fomentar una discusión sobre los pros y contras de los métodos recursivos, especialmente en relación con el uso eficiente de la memoria y la prevención de sobrecargas.

- Animar a los aprendedores a que revisen el capítulo 4 del libro de apoyo, “Chapter 4. Object-Oriented Programming”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 13.

Al profesor impartidor, se le recomienda lo siguiente:

- Hacer una introducción clara, donde se destaque la importancia de los constructores en la creación de objetos y por qué resultan esenciales para inicializar los atributos de una clase.

- Proporcionar ejemplos que demuestren la utilidad y la forma de definir constructores por defecto, con parámetros y de copia en Java; al presentarlos, se recomienda enfatizar cómo la definición de los más adecuados puede prevenir errores y permitir un control más preciso del comportamiento de los objetos.

- Mencionar y explicar, con ejemplos prácticos, los diferentes tipos de constructores y cuándo es más conveniente utilizar cada uno.

- Animar a los aprendedores a que revisen el capítulo 4 del libro de apoyo, “Chapter 4. Object-Oriented Programming”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 14.

Al profesor impartidor, se le recomienda lo siguiente:

- Destacar las diferencias clave entre composición y agregación; para ello, se sugiere emplear casos de la vida real, como la relación entre un automóvil y sus componentes, para fortalecer la comprensión entre relaciones fuertes y débiles entre objetos.

- Ejemplificar la diferencia entre composición y agregación a partir de ejemplos de implementación claros y detallados.

- Formar equipos para simular las relaciones entre objetos, asignando roles específicos para ejemplificar, de manera práctica, cómo interactúan en un sistema.

- Prestar atención a la explicación detallada de la relación de dependencia, así como a la asociación entre clases, ya que estos conceptos pueden necesitar una mayor clarificación y más ejemplos prácticos para ser comprendidos.

- Animar a los aprendedores a que revisen el capítulo 5 del libro de apoyo, “Chapter 5. OOP in Depth”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 15.

Al profesor impartidor, se le recomienda lo siguiente:

- Explicar a los aprendedores de qué forma la abstracción permite identificar y separar los aspectos esenciales de un objeto, simplificando su representación, sin preocuparse por detalles irrelevantes.

- Abordar la creación de clases abstractas, el uso de interfaces y la implementación de herencia, proporcionando ejemplos claros y prácticos que ilustren cómo se aplican dichos conceptos en situaciones reales.

- Realizar una lluvia de ideas para fomentar la participación de los aprendedores, de tal manera que se les permita explorar cómo pueden utilizar la abstracción en diferentes contextos de desarrollo de software.

- Animar a los aprendedores a que revisen el capítulo 5 del libro de apoyo, “Chapter 5. OOP in Depth”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 16.

Al profesor impartidor, se le recomienda lo siguiente:

- Explicar la importancia del encapsulamiento y de los modificadores de acceso en la construcción de un software robusto y mantenible; asimismo, se sugiere comentar por qué el encapsulamiento permite ocultar la complejidad interna de una clase, mientras proporciona una interfaz pública clara.

- Fomentar la comprensión de que una clase debe actuar como una "caja negra", es decir, que revela solo lo necesario para su uso externo; de esta manera, se busca promover tanto la modularidad como la evolución flexible del software.

- Agrupar a los aprendedores en equipos para que discutan y propongan ejemplos prácticos sobre las diferencias en el uso de los modificadores *public*, *private* y *protected*; después, se recomienda proporcionar algunos ejemplos concretos, así como plantear ejercicios que involucren la identificación de aquellas partes de una clase que deben ser públicas, privadas o protegidas. De esta manera, se intenta fomentar la comprensión práctica de dichos conceptos.

- Describir detalladamente de qué manera los aprendedores pueden identificar aquellos elementos que deben ser visibles desde fuera de una clase, así como usar **getters** y **setters** para interactuar con los atributos; para esta finalidad, se sugiere la creación de mapas mentales que visualicen la estructura de una clase y sus interacciones.

- Animar a los aprendedores a que revisen el capítulo 5 del libro de apoyo, “Chapter 5. OOP in Depth”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 17.

Al profesor impartidor, se le recomienda lo siguiente:

- Basar la explicación del tema en el recurso de la ruta de aprendizaje de Oracle Academy, Java Fundamentals 7-4: Herencia.

- Comenzar con una introducción clara y concisa sobre el concepto de herencia en POO, así como destacar su importancia en la reutilización de código y en la creación de jerarquías de clases.

- Proporcionar ejemplos prácticos que ilustren situaciones del mundo real donde la herencia resulta útil; además, se aconseja crear una jerarquía simple de clases para demostrar cómo se heredan los atributos y métodos.

- Explicar claramente la diferencia entre clases base (superclases) y clases derivadas (subclases); asimismo, hay que mostrar cómo se heredan los miembros de la superclase y cómo se pueden modificar o extender en la subclase.

- Animar a los aprendedores a que revisen el capítulo 4 del libro de apoyo, “Chapter 4. Object-Oriented Programming”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 18.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar con una explicación sobre la importancia de los modificadores de acceso en la herencia; además, es necesario destacar los cuatro modificadores de acceso en Java: public, protected, *default* (sin modificador) y private. En este sentido, hay que asegurarse de explicar y enfatizar las diferencias entre cada uno de ellos.

- Proporcionar ejemplos prácticos que muestren cómo se aplican los diferentes modificadores de acceso en situaciones de herencia, así como impulsar a los aprendedores a experimentar con estos conceptos a través de la escritura de código.

- Presentar desafíos para que los aprendedores piensen críticamente acerca de cómo se aplican los modificadores de acceso en situaciones específicas.

- Animar a los aprendedores a que revisen el capítulo 5 del libro de apoyo, “Chapter 5. OOP in Depth”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 19.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar con una explicación sobre el concepto fundamental del polimorfismo, el cual se define como la capacidad de objetos de diferentes clases de responder a un mismo mensaje; por ello, se recomienda ejemplificar con situaciones del mundo real para facilitar la comprensión.

- Basar las explicaciones en el tema 7-5 de la ruta de aprendizaje de Java Fundamentals, de la Oracle Academy.

- Enseñar el polimorfismo de subtipos, es decir, donde el objeto de una subclase puede ser tratado como uno de la superclase; además, se sugiere proporcionar ejemplos prácticos que permitan entender por qué esto facilita la flexibilidad en el diseño y la reutilización de código.

- Explorar de qué forma los métodos polimórficos permiten que las clases derivadas proporcionen implementaciones específicas mientras se mantienen dentro del contrato de la interfaz común; además, se recomienda demostrar cómo las subclases pueden sobrescribir métodos de la superclase para lograr el polimorfismo.

- Animar a los aprendedores a que revisen el capítulo 7 del libro de apoyo, “Chapter 7. The Java Collections Frameword and Generics”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación del tema 20.

Al profesor impartidor, se le recomienda lo siguiente:

- Iniciar con una explicación sobre el concepto de conversión de objetos, el cual implica cambiar el tipo de un objeto a otro; de igual forma, se aconseja ejemplificar situaciones en las que la conversión de objetos resulta necesaria para aprovechar la herencia y la interfaz común.

- Enseñar la diferencia entre el casting implícito (*upcasting*), donde un objeto se convierte a su tipo base automáticamente, y el casting explícito (*downcasting*), el cual requiere una declaración explícita; por este motivo, se sugiere proporcionar ejemplos para ilustrar ambos tipos.

- Destacar la importancia de verificar la compatibilidad de tipos antes de realizar un casting; de hecho, se aconseja discutir cómo las excepciones de casting (**ClassCastException**) pueden ocurrir si la conversión no es válida.

- Explicar por qué el casting resulta especialmente útil cuando se trabaja con clases relacionadas por herencia; en este sentido, se puede brindar un ejemplo como el siguiente:
  - “El casting resulta especialmente útil cuando se trabaja con clases relacionadas por herencia, ya que permite tratar un objeto como si fuera una instancia creada a partir de su clase padre en la jerarquía y, luego, volver a tratarlo como la clase hija; por ejemplo, si tienes una clase **Animal** y una subclase **Perro**, puedes hacer algo así:
    - Animal a = new Perro(); // Creamos un objeto de tipo.
    - Perro a.ladrar(); // Llamamos al método ladrar del objeto.
    - Perro a.morder(); // Llamamos al método morder del objeto Animal.

- Animar a los aprendedores a que revisen el capítulo 5 del libro de apoyo, “Chapter 5. OOP in Depth”, para profundizar en el aprendizaje.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad I:

Al profesor impartidor, se le recomienda lo siguiente:

- Asegurarse de revisar las declaraciones de variables de los aprendedores para verificar que hayan utilizado los tipos de datos adecuados, según las instrucciones, y evaluar si han declarado las variables correctamente en términos de sintaxis.

- Verificar que utilizaron literales, de manera apropiada, para asignar valores a las variables.

- Cerciorarse de que las asignaciones de valores se han realizado correctamente, según las instrucciones.

- Revisar los ejercicios relacionados con operadores booleanos y aritméticos para asegurarse de que los aprendedores los aplicaron correctamente en sus expresiones.

- Comprobar si los resultados de las operaciones son correctos y coinciden con las expectativas.

- Evaluar los ejemplos de conversión de tipos de datos para determinar si los aprendedores comprenden la conversión implícita y explícita.

- Comprobar si utilizaron casting de manera adecuada y en situaciones pertinentes.

- Prestar atención a la claridad de las explicaciones proporcionadas por los aprendedores junto con su código.

- Evaluar si explicaron sus pasos de manera coherente y fácil de entender.

- Asegurarse de que los aprendedores cumplieron con todos los requisitos de la actividad, incluyendo la creación de la clase "Coche", el diagrama de clases UML y la implementación del programa en Java.

- Verificar que hayan incluido entradas y salidas de usuario, según las instrucciones.

- Revisar si los aprendedores proporcionaron comentarios adecuados en su código para explicar sus decisiones de diseño y lógica.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad II:

Al profesor impartidor, se le recomienda lo siguiente:

- Asegurarse de que los aprendedores construyan un programa en Java que solicite y reciba correctamente el nombre del usuario.

- Evaluar si han utilizado estructuras de selección (if o switch), de manera efectiva, para verificar si el nombre ingresado coincide con uno en específico.

- Verificar que los aprendedores aprovechen las operaciones de manejo de cadenas de texto para realizar comparaciones con sensibilidad a mayúsculas o minúsculas, así como para proporcionar mensajes adecuados.

- Comprobar la precisión de las comparaciones de cadenas y los mensajes personalizados.

- Asegurarse de que los aprendedores desarrollen un programa que genere y almacene, con precisión, una lista de números enteros pares del dos al 100, utilizando estructuras de repetición (for o while) y arreglos.

- Cerciorarse de que utilicen una estructura de repetición para recorrer el arreglo y mostrar correctamente los números almacenados.

- Comprobar la exactitud en la generación y almacenamiento de números enteros pares.

- Prestar atención a la claridad de las explicaciones proporcionadas por los aprendedores, junto con su código, en ambas partes del ejercicio.

- Evaluar si explicaron, de manera coherente y fácil de entender, cómo emplearon Strings, estructuras de selección y de repetición, así como arreglos, en su programa.

- Asegurarse de que los aprendedores cumplieron con todos los requisitos de la actividad, incluyendo la descripción del código desarrollado en ambas partes del ejercicio.

- Verificar que adjuntaron una captura de pantalla de la ejecución del programa.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad III:

Al profesor impartidor, se le recomienda lo siguiente:

- Asegurarse de que el aprendedor tenga claro cómo incluir los métodos **setters** y **getters** para cada atributo.

- Explicar claramente a los aprendedores cómo se calcula el precio de venta del producto para que, así, puedan desarrollar el método **calcularPrecio** de manera correcta.

- Confirmar que los aprendedores comprenden bien la diferencia entre los métodos de clase y los estáticos.

- Asegurarse de que los aprendedores comprendan de qué forma se envían objetos a manera de parámetros y, además, que tengan clara la funcionalidad del método **comparaProductos**.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad IV:

Al profesor impartidor, se le recomienda lo siguiente:

- Asegurarse de que el aprendedor comprenda el caso planteado.

- Orientar al aprendedor en el esbozo de la estructura de clases, pero motivarlo para que diseñe su propia estructura.

- Usar ejemplos con situaciones parecidas al caso planteado para que el aprendedor elabore su propio diseño.

- Motivar a los aprendedores para que enriquezcan su diseño con sus propias ideas y aportaciones.

- Validar el diseño del aprendedor antes de que construya su programa, de tal manera que no arrastre ningún error.

- Motivar al aprendedor para que documente, de manera correcta, el desarrollo de su actividad.

Notas para el profesor impartidor. Estas corresponden a la explicación de la actividad V:

Al profesor impartidor, se le recomienda lo siguiente:

- Explicar que esta actividad consiste en presentar el examen final del curso Java Fundamentals, de Oracle Academy.

- Antes de que presenten el examen, hay que preguntar a los aprendedores si hay alguna duda y, sobre todo, animarlos a repasar el contenido de la plataforma para ampliar y ahondar en su aprendizaje sobre POO.

- Guiar a los aprendedores por la plataforma de Oracle Academy para que localicen el apartado donde se encuentra el examen final.

Notas para el profesor impartidor. Estas corresponden a la explicación del proyecto fase I:

Antes de comenzar con este avance, hay que revisar exactamente en qué consiste, ya que esto te permitirá atender las dudas de los aprendedores; en este sentido, es muy importante que los temas vistos durante las primeras semanas queden muy claros, pues esta fase del proyecto se diseñó con base en ellos.

Por otro lado, resulta fundamental que los aprendedores comprendan a cabalidad los requisitos del sistema de gestión de biblioteca, ya que deben ser capaces de identificar sus funcionalidades clave, como registrar, consultar, modificar y eliminar libros, usuarios y préstamos, así como generar reportes.

Al profesor impartidor, se le recomienda lo siguiente:

- Comentar que, para este proyecto, deben identificar clases y objetos; esto implica determinar entidades como Libro, Usuario, Préstamo y Biblioteca, además de comprender cómo se relacionan entre sí.

- Recordarles a los aprendedores que, para esta fase del proyecto, deben utilizar diagramas UML que representen las relaciones entre las clases; asimismo, deben definir asociaciones, agregaciones, composiciones, herencias y demás vínculos, según corresponda al diseño del sistema. En este sentido, es muy importante mencionar que, como no se ha dado una formación previa de UML, el diagrama no tiene que ser tan exacto o estricto.

- Subrayar que, para cada clase, deben especificarse los atributos y métodos pertinentes; esto incluye la definición de campos de datos, como título del libro o fecha de préstamo, y métodos que realizan acciones relacionadas con la clase.

- Hacer hincapié en que la documentación es clave en el diseño de sistemas y, por tanto, deben crear una lo más detallada posible, es decir, que explique sus decisiones de diseño,

cómo funcionan las clases y de qué manera se relacionan entre sí. Esto facilita la comprensión y revisión del diseño.

- Antes de avanzar hacia la implementación, es importante que los aprendedores validen su diseño; esto lo pueden hacer mediante revisiones de pares, o bien, con herramientas UML para verificar la coherencia y corrección de los diagramas.

- Hacer revisiones periódicas del progreso de los aprendedores, así como proporcionar retroalimentación constructiva, ya que esto les ayuda a mejorar sus diseños y a aprender de sus errores.

Notas para el profesor impartidor. Estas corresponden a la explicación del proyecto fase II:

Antes de comenzar la implementación (codificación), los aprendedores deben revisar detenidamente su diseño UML de la fase anterior, pues esto les ayudará a entender la estructura del sistema y cómo deben traducirlo a código.

Al profesor impartidor, se le recomienda lo siguiente:

- Alentar a los aprendedores para que apliquen los conceptos de la POO, como la abstracción, el encapsulamiento, el polimorfismo y la herencia, en la implementación de las clases y objetos del sistema.

- Fomentar la creación de un plan de desarrollo que incluya una lista de tareas y un cronograma; esto les ayudará a mantenerse organizados y a asegurarse de que avanzan de manera efectiva.

- Instruir a los aprendedores para que construyan las clases y objetos que representen el sistema, asignando los atributos y los métodos correspondientes, según el diseño UML. En este sentido, resulta muy importante que se mantenga una nomenclatura y estructura de código coherentes.

- Ayudar a los aprendedores para que implementen las relaciones entre clases y objetos a partir de los mecanismos más apropiados del lenguaje de programación Java; esto puede incluir la herencia, la implementación de interfaces, el uso de clases abstractas y el polimorfismo.

- Guiar a los aprendedores en la implementación de las funcionalidades del sistema, por ejemplo, en agregar, modificar, eliminar y consultar libros, usuarios y préstamos; para lograrlo, deben utilizar las estructuras de datos y los operadores adecuados en Java.

- Animar a los aprendedores a que realicen pruebas exhaustivas del sistema; esto incluye la ejecución del programa con datos de entrada para probar los métodos y, de manera opcional, la creación de casos de prueba.

- Resaltar cuán importante resulta documentar la implementación y pruebas, lo cual incluye comentarios en el código fuente explicando su lógica, así como la descripción de los resultados obtenidos durante las pruebas. La documentación debe ser clara y concisa.

- Enfatizar que, al final, los aprendedores deben elaborar una conclusión personal en la que destaquen los aprendizajes obtenidos, los desafíos enfrentados y las mejoras que pueden realizar en futuros proyectos similares.

- Animar a los aprendedores a que revisen y colaboren entre ellos para obtener diferentes perspectivas y sugerencias. La revisión por pares puede ayudar a identificar errores y mejorar la calidad del código.

- En la medida de lo posible, se recomienda establecer un proceso de retroalimentación y resolución de las dudas que los aprendedores puedan tener durante el proceso de crear un sistema de gestión de biblioteca.

Rúbrica del avance del proyecto (fase I)

Criterios de evaluación	Nivel de desempeño			%
	Altamente competente 100%-86%	Competente 85%-70%	Aún sin desarrollar la competencia 69%-0%	
1. Análisis de requisitos funcionales.	20 – 18 puntos Identifica claramente los requisitos funcionales. Detalla adecuadamente cómo se abordarán las operaciones de registro y consulta de libros, usuarios y préstamos. Integra la explicación de cómo se implementarán las validaciones de datos y las restricciones en los préstamos.	17 – 15 puntos Identifica los requisitos funcionales. Detalla cómo se abordarán las operaciones de registro y consulta de libros, usuarios y préstamos. No integra la explicación de cómo se implementarán las validaciones de datos y las restricciones en los préstamos.	14 – 0 puntos No logra identificar los requisitos funcionales. No detalla cómo se abordarán las operaciones de registro y consulta de libros, usuarios y préstamos. No integra la explicación de cómo se implementarán las validaciones de datos y las restricciones en los préstamos.	20
2. Diseño de clases y objetos.	20 – 18 puntos Identifica claramente las clases y objetos relevantes en el sistema. Define atributos y métodos para cada clase y objeto. Considera la coherencia y la modularidad en el diseño.	17 – 15 puntos Identifica las clases y objetos relevantes en el sistema. Define atributos y métodos para cada clase y objeto. No considera la coherencia ni la modularidad en el diseño.	14 – 0 puntos Muestra dificultades para identificar las clases y objetos relevantes en el sistema. No define atributos ni métodos para cada clase y objeto. No considera la coherencia ni la modularidad en el diseño.	20
3. Relaciones entre clases.	15 – 13 puntos Establece adecuadamente las distintas relaciones, como asociación, agregación, composición y herencia, entre las clases y objetos identificados. Explica las decisiones tomadas para cada tipo de relación.	12 – 10 puntos Establece adecuadamente las distintas relaciones, como asociación, agregación, composición y herencia, entre las clases y objetos identificados. No explica las decisiones tomadas para cada tipo de relación.	9 – 0 puntos Establece incorrectamente las distintas relaciones, como asociación, agregación, composición y herencia, entre las clases y objetos identificados. No explica las decisiones tomadas para cada tipo de relación.	15
	20 – 18 puntos	17 – 15 puntos	14 – 0 puntos	

4. Diagramas UML.	Elabora los diagramas de UML con claridad y, además, incluye todos los otros solicitados, según sean necesarios. Muestra coherencia entre los diferentes diagramas y, por tanto, asegura una representación unificada del sistema.	Crea los diagramas de UML con claridad y, además, incluye todos los otros solicitados, según sean necesarios. No muestra coherencia entre los diferentes diagramas.	Crea los diagramas de UML de manera incorrecta. No muestra coherencia entre los diferentes diagramas.	20
5. Documentación del diseño.	15 – 13 puntos	12 – 10 puntos	9 – 0 puntos	15
	Presenta la documentación clara y completa del diseño, explicando cada elemento de los diagramas UML. Describe detalladamente el significado y la función de cada clase, objeto, relación, atributo y método.	Presenta la documentación del diseño, explicando cada elemento de los diagramas UML. No describe el significado y la función de cada clase, objeto, relación, atributo y método.	Presenta la documentación incompleta del diseño. No describe el significado y la función de cada clase, objeto, relación, atributo y método.	
6. Consideración de requisitos no funcionales.	10 – 8 puntos	7 – 5 puntos	4 – 0 puntos	10
	Evalúa el diseño en términos de facilidad de uso, robustez, eficiencia y escalabilidad. Explica cómo se abordan los aspectos no funcionales del sistema.	Evalúa el diseño en términos de facilidad de uso, robustez, eficiencia y escalabilidad. No explica cómo se abordan los aspectos no funcionales del sistema.	No evalúa el diseño en términos de facilidad de uso, robustez, eficiencia y escalabilidad. No explica cómo se abordan los aspectos no funcionales del sistema.	
			TOTAL	**100%**

Rúbrica del proyecto final (fase II)

	Nivel de desempeño			
Criterios de evaluación	Altamente competente 100%-86%	Competente 85%-70%	Aún sin desarrollar la competencia 69%-0%	%
1. Creación de clases y objetos.	20 - 18 Crea adecuadamente las clases que representan los elementos del sistema (libros, usuarios y préstamos), con una asignación correcta de atributos y métodos a cada clase; además, hace un uso efectivo de los conceptos de abstracción y encapsulamiento.	17 - 15 Crea clases que representan los elementos del sistema (libros, usuarios y préstamos), con una asignación correcta de atributos y métodos a cada clase; además, hace un uso efectivo de algunos conceptos, como abstracción y encapsulamiento, aunque con algunos errores.	14 - 0 No crea las clases que representan los elementos del sistema (libros, usuarios y préstamos), con una asignación correcta de atributos y métodos a cada clase; sin embargo, hace un uso efectivo de algunos conceptos, como abstracción y encapsulamiento, aunque con algunos errores.	20
2. Implementación de relaciones.	20 – 18 Implementa correctamente las relaciones entre clases a partir de los mecanismos de la POO (herencia, polimorfismo y conversión de objetos); asimismo, demuestra una comprensión y aplicación efectivas de los conceptos de la POO.	17 - 15 Implementa correctamente las relaciones entre clases a partir de los mecanismos de la POO (herencia, polimorfismo y conversión de objetos); asimismo, demuestra una comprensión y aplicación efectivas de los conceptos de la POO, aunque con algunos errores.	14 - 0 No implementa correctamente las relaciones entre clases a partir de los mecanismos de la POO (herencia, polimorfismo y conversión de objetos); sin embargo, demuestra una comprensión y aplicación efectivas de los conceptos de la POO.	20
	20 – 18	17 - 15	14 - 0	

3.	Desarrollo de funcionalidades.	Implementa, de manera efectiva, las funcionalidades del sistema mediante las estructuras de datos y operadores apropiados del lenguaje Java, es decir, con el uso adecuado de arreglos, cadenas de texto, estructuras de selección y bucles de repetición.	Implementa, de manera efectiva, las funcionalidades del sistema mediante estructuras de datos y operadores apropiados del lenguaje Java, es decir, con el uso adecuado de arreglos, cadenas de texto, estructuras de selección y bucles de repetición, aunque comete algunos errores.	No implementa, de manera efectiva, las funcionalidades del sistema mediante estructuras de datos y operadores apropiados del lenguaje Java, es decir, con el uso adecuado de arreglos, cadenas de texto, estructuras de selección y bucles de repetición.	20
4.	Pruebas del sistema.	15 – 13 Realiza pruebas exhaustivas, es decir, ejecuta el sistema con datos de entrada y prueba los métodos; opcionalmente, plantea casos de prueba para validar, de manera efectiva, diferentes escenarios donde se aplique la depuración de errores y, por ende, se asegura de su correcto funcionamiento.	12 - 10 Realiza pruebas exhaustivas, es decir, ejecuta el sistema con datos de entrada y prueba los métodos; opcionalmente, plantea casos de prueba para validar, de manera efectiva, diferentes escenarios donde se aplique la depuración de errores y, por ende, se asegura de su correcto funcionamiento, pero comete algunos errores.	9 - 0 No realiza pruebas exhaustivas, es decir, no ejecuta el sistema con datos de entrada ni prueba los métodos; opcionalmente, plantea casos de prueba para validar, de manera efectiva, diferentes escenarios con la depuración de errores y, por ende, se asegura de su correcto funcionamiento.	15
5.	Documentación del código.	15 - 13 Documenta clara y detalladamente el código fuente, de tal manera que explica la lógica detrás de las	12 - 10 Documenta clara y detalladamente el código fuente, de tal manera que explica la lógica detrás de	9 – 0 No documenta clara y detalladamente el código fuente, de tal manera que no explica la lógica	

	implementaciones y, además, incluye comentarios que facilitan la comprensión del código por parte de otros programadores.	las implementaciones y, además, incluye comentarios que facilitan la comprensión del código por parte de otros programadores, pero comete algunos errores.	detrás de las implementaciones, o bien, no incluye comentarios que faciliten la comprensión del código por parte de otros programadores.	15
6. Conclusión personal.	10-9	8-7	6-0	
	Elabora una conclusión personal reflexiva sobre el proceso de creación del sistema, donde incluye la explicación de los desafíos enfrentados, lecciones aprendidas y áreas de mejora identificadas.	Elabora una conclusión personal reflexiva sobre el proceso de creación del sistema, donde incluye la explicación de los desafíos enfrentados, lecciones aprendidas y áreas de mejora identificadas, pero comete algunos errores.	No elabora una conclusión personal reflexiva sobre el proceso de creación del sistema, donde incluya la explicación de los desafíos enfrentados, lecciones aprendidas y áreas de mejora identificadas.	10
			TOTAL	**100%**

Prácticas de bienestar

Práctica 1

Nombre de la práctica	Un momento para respirar.
Descripción de la práctica	Aprender a respirar por nariz y a tranquilizar tu mente.
Palabras clave	Fortalezas de carácter, autorregulación.
Instrucciones para el aprendedor	La autorregulación, también percibida como control, es una fortaleza de carácter muy importante dentro de la psicología positiva. Este concepto implica regular lo que uno siente y hace, ser disciplinado, así como mantener un control sobre los apetitos y, especialmente, sobre las emociones. En la actualidad vivimos situaciones muy estresantes que provocan que nuestra reacción instintiva y natural ante ellas sea estallar en ira, pero las consecuencias de este comportamiento no solo se quedan en nosotros, sino que también pueden llegar a afectar a terceros. A continuación, se presenta un ejercicio que te ayudará a cultivar la fortaleza de autorregulación: 1. Toma dos minutos de tu tiempo, siéntate en un lugar cómodo, donde no haya mucho ruido que te pueda distraer. 2. Escucha música de relajación (crea tu propio ambiente de meditación). 3. Comienza a respirar y exhalar por la nariz. Trata de que tu respiración y exhalación dure el mismo tiempo. 4. Fija tu mente en tu respiración, en cómo entra y sale el aire de tu cuerpo. Así durante dos minutos. Te recomendamos que, si durante este periodo algún pensamiento de olvide algo en la oficina, más tarde tengo que hacer tal actividad, etc. llega a tu mente, solo déjalo

	pasar y regresa a tu concentración en tu respiración. Al finalizar los dos minutos sentirás paz en tu ser, comienza a hacer este ejercicio de respiración y meditación todos los días, y poco a poco vas aumentando los minutos de este.
Fuente	Conferencia Rosalinda Ballesteros.

Práctica 2

Nombre de la práctica	Fomentando la atención plena.
Descripción de la práctica	Llevarás a cabo breves ejercicios de meditación para fomentar la atención plena en tus actividades diarias.
Palabras clave	Atención plena, fortalezas de carácter, autorregulación.
Instrucciones para el aprendedor	La meditación es una herramienta que ayuda a mejorar el desempeño de cualquier persona, ya que fomenta el desarrollo de la atención plena en una sola actividad. Para fomentar la atención plena y lograr cada vez más estar en una zona de concentración mientras realizas tus actividades cotidianas, puedes llevar a cabo los siguientes ejercicios de meditación: Encuentra en algún momento del día cinco minutos para ti, siéntate en un lugar cómodo, donde no tengas distracciones. 1. Haz tres respiraciones profundas por la nariz y exhala por la nariz. 2. Comienza a hacer un repaso de tu día, de lo que más te acuerdes, Ej. Te levantaste, ¿qué hiciste? ¿Desayunaste? ¿Te bañaste? ¿Diste los buenos días?, etcétera. Si desayunaste, ¿qué fue lo que desayunaste? ¿Te gustó? ¿Tomaste tu alimento despacio o apurado?, si estabas apurado, ¿qué era lo que te tenía en esa situación? 3. Sigue meditando en lo que te acuerdes: ¿te molestaste con alguien? ¿Por qué? ¿Qué fue lo que pasó? ¿Crees que era posible haber

	reaccionado de alguna manera más pacifica? Con este ejercicio te darás cuenta de que reaccionamos o hacemos cosas de manera automática, algunas veces si estamos más conscientes y presentes, podemos tener otra actitud sin que alguna situación nos afecte demasiado.
Fuente	Eby, D. (s.f.) *Creativity and Flow Psychology*. Talent Development Resources. http://talentdevelop.com/articles/Page8.html

Práctica 03

Nombre de la práctica	Experiencias difíciles.
Descripción de la práctica	En esta práctica podrás analizar las estrategias que seguiste para afrontar problemáticas y cómo aprendiste de tales sucesos.
Palabras clave	Resiliencia.
Instrucciones para el aprendedor	Todos hemos pasado por situaciones complejas, no solo en lo laboral, sino también en el ámbito familiar y personal. La manera en que enfrentamos dichos obstáculos es muy diferente, algunas personas continúan con su vida sin problema alguno, a otras tantas se les complica esa transición, e incluso hay quienes no pueden sobreponerse a las experiencias difíciles. De acuerdo con Margarita Tarragona (2012), La resiliencia es la capacidad de reponerse tras la adversidad, de recuperarse después de vivir experiencias difíciles, dolorosas o traumáticas. Para algunos, la resiliencia implica no solo salir adelante después de una situación muy dura, sino incluso crecer o ser mejor a raíz de esta experiencia. La siguiente práctica te ayudará a fomentar esta importante cualidad: 1. Crea una tabla con **tres** columnas y **cinco** filas.

	2. En la primera columna escribe un evento difícil o desagradable al que te hayas enfrentado en tu vida. 3. En la segunda columna menciona cuáles son tus creencias sobre esa adversidad. 4. En la tercera columna describe las consecuencias que tiene esa creencia. 5. Cuando termines, lee toda la tabla y reflexiona sobre cómo te ha cambiado cada evento y cómo lo enfrentaste. 6. Escribe al final, ¿cómo enfrentarías cada evento hoy en día?
Fuente	Metodología ABC. Fundamentos de psicología positiva.

Práctica 04

Nombre de la práctica	Concentrarse en lo positivo.
Descripción de la práctica	Analizarás sucesos que te hayan ocurrido recientemente, buscando orientar el análisis hacia las consecuencias positivas.
Palabras clave	Resiliencia, esperanza.
Instrucciones para el aprendedor	¿Qué es lo primero que piensas cuando recibes una noticia inesperada? O bien, ¿qué te imaginas cuando un acontecimiento complejo se presenta ante ti? La mayoría de las personas automáticamente se concentra en el peor de los escenarios independientemente del tipo de noticia que reciban. Martin Seligman (2011) sugiere hacer un breve ejercicio para fomentar la resiliencia y la esperanza con base en la premisa antes señalada: 1. Piensa en una noticia reciente que hayas recibido y que creas es negativa para ti.

	2. Luego de analizarla, haz una tabla con tres columnas, en la primera señala cuál sería el peor de los escenarios posibles que pudieran resultar de esa noticia, en la segunda columna señala cuál sería el mejor de los escenarios posibles, y en la última cuál es el escenario que realmente tiene mayor probabilidad de ocurrir. 3. Reflexiona sobre los tres escenarios, ¿cómo enfrentarías a cada uno de ellos? Procura repetir este ejercicio cada vez que sientas que te enfrentas a una situación complicada. Hacerlo te dará perspectiva y te ayudará a cultivar tu resiliencia.
Fuente	Seligman, M. (2011). *Building Resilience*. Recuperado de https://hbr.org/2011/04/building-resilience

Práctica 05

Nombre de la práctica	Crecimiento postraumático.
Descripción de la práctica	En esta práctica harás un recuento de las situaciones difíciles a las que te has enfrentado y reflexionarás sobre lo positivo que surgió de ellas.
Palabras clave	Resiliencia.
Instrucciones para el aprendedor	De acuerdo con Margarita Tarragona (2012) La resiliencia es la capacidad de reponerse tras la adversidad, de recuperarse después de vivir experiencias difíciles, dolorosas o traumáticas. Para algunos la resiliencia implica no solo salir adelante después de una situación muy dura, sino incluso crecer o ser mejor a raíz de esta experiencia. La siguiente práctica te ayudará a fomentar esta importante cualidad:

	1. Escribe acerca de un momento en el que enfrentaste una adversidad significativa o pérdida. 2. Primero escribe acerca de las puertas que se te cerraron debido a esa adversidad o pérdida, ¿qué perdiste? 3. Después escribe acerca de las puertas que se abrieron al término o como secuela de esa adversidad o pérdida. 4. ¿Hay nuevas maneras de actuar, pensar, o relacionarse que son más probables de suceder ahora?
Fuente	Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro A Primer in Positive Psychology de Christopher Peterson.

Práctica 06

Nombre de la práctica	La mejor versión de ti mismo.
Descripción de la práctica	Escribe durante por lo menos 20 minutos acerca de la mejor versión posible de ti mismo.
Palabras clave	Emociones positivas, fortalezas de carácter, autorregulación, esperanza.
Instrucciones para el aprendedor	Imagina que dentro de 20 años has crecido en todas las áreas o maneras que te gustaría crecer y las cosas te han salido tan bien como te las imaginaste. ¿Cómo es esa mejor versión de ti mismo? ¿Qué hace él o ella cotidianamente? ¿Qué dicen los demás acerca de él o ella? No es necesario que compartas este escrito, ya que el objetivo de esta reflexión es enfocarse en la experiencia que viviste mientras reflexionabas en esa mejor versión posible de ti mismo.
Fuente	Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive

	Psychology Center, y basado en el libro A Primer in Positive Psychology de Christopher Peterson.

Práctica 07

Nombre de la práctica	Obtener lo que quieres.
Descripción de la práctica	Reflexionarás sobre alguna meta que desees alcanzar y propondrás una forma de conseguirla.
Palabras clave	Logro, involucramiento, fortalezas de carácter, esperanza, autorregulación, metas, objetivos a largo plazo.
Instrucciones para el aprendedor	Tener una idea clara de lo que deseas lograr a corto, mediano y largo plazo es de suma importancia, pues te ayuda a seguir un camino trazado previamente. Para que puedas generar esta guía, responde las siguientes preguntas: 1. ¿Qué quieres lograr? Al trazar tu meta, procura que esta sea específica, medible, alineada, realista, retadora y con una fecha para lograrla. Piensa en algo y utiliza el método SMART para definirla. 2. ¿Qué te impide que lo tengas en este momento? 3. ¿Qué sufrimiento estás experimentando en tu vida por no tenerlo en este momento? 4. ¿Qué placer, involucramiento, relación, significado o logro tendrías en tu vida si tuvieras eso en este momento? 5. ¿Qué hábitos te detienen o no te dejan avanzar hacia eso que quieres? 6. ¿Qué nuevos hábitos podrías generar para ayudarte a obtener lo que quieres? 7. ¿Qué dos cosas podrías hacer para romper con los hábitos que no te permiten

	avanzar hacia lo que quieres y generar hábitos nuevos? 8. ¿Te comprometes a hacer esas dos cosas? Si es así, ¿cuándo las harás? Escribe tus resultados en un sitio donde puedas verlos constantemente.
Fuente	Ejercicio contribuido por Taylor Kreiss de University of Pennsylvania Positive Psychology Center, y basado en el libro A Primer in Positive Psychology de Christopher Peterson.

Práctica 08

Nombre de la práctica	Felicidad en el trabajo.
Descripción de la práctica	Reflexionarás sobre las distintas dimensiones de tu vida cotidiana, enfocando el análisis a cómo fomentar un estado de ánimo y relaciones positivas en el ámbito laboral.
Palabras clave	Involucramiento, emociones positivas, relaciones positivas.
Instrucciones para el aprendedor	Elegir conscientemente maneras de incrementar la felicidad en el trabajo puede hacer la diferencia en cómo nosotros nos sentimos y que tan bien nos desempeñamos. En lugar de quejarnos del trabajo, ¿por qué no pensar en cómo podemos obtener mayor felicidad de lo que hacemos? Estar más involucrados en lo que hacemos, contribuye a nuestra felicidad y bienestar y nos lleva a un mejor desempeño y productividad. A manera de reflexión, responde las siguientes preguntas que están enfocadas a distintas dimensiones de tu vida: **DAR**: ¿Cómo estoy apoyando a mis colaboradores, compañeros, líderes, proveedores y clientes?

	RELACIONES: ¿Cómo puedo mejorar mis relaciones en el trabajo? ¿Cómo logro un balance entre la vida laboral y familiar? **EJERCICIO**: ¿Cómo puedo integrar la actividad física dentro de mis actividades diarias? ¿Cómo aseguro que estoy comiendo bien y descansando lo suficiente? **CONCIENCIA**: ¿Cómo puedo construir momentos de atención plena en mi día laboral? **ENSAYO**: ¿Qué habilidades estoy construyendo? ¿Qué cosas nuevas he experimentado? **DIRECCIÓN**: ¿Cuáles son mis metas laborales hoy, esta semana, este año? ¿Cómo caben y contribuyen estas con mis metas de vida y me ayudan a desarrollar mis competencias en la construcción de mis relaciones y cómo contribuyo con lo anterior a ayudar a otros? ¿Cómo se pueden alinear mis metas laborales con las de mi equipo y la organización? **RESILIENCIA**: ¿Cuáles son mis tácticas para lidiar con los retos difíciles en el trabajo? ¿Me estoy enfocando en lo que puedo controlar? ¿Necesito pedir ayuda a otros? ¿Hay alguien a mi alrededor que requiere de mi ayuda? **EMOCIÓN**: ¿Qué cosas, aunque sean pequeñas, puedo encontrar que me pueden hacer sentir bien en mi trabajo hoy? ¿Qué me ha hecho sonreír?
Fuente	Tomado de catálogo de actividades para profesores.

Práctica 9

Nombre de la práctica	Interacciones positivas.
Descripción de la práctica	Reflexionarás sobre las cualidades positivas que aprecias de las personas con las que interactúas diariamente.
Palabras clave	Relaciones positivas.
Instrucciones para el aprendedor	Puedes obtener mayor gozo de los momentos que compartes con tus colegas si te tomas el tiempo para pensar en lo que valoras y aprecias de ellos. Diversas investigaciones muestran que enfocarse en

lo positivo que sucede diariamente, ayuda a incrementar nuestra felicidad, y lo mismo aplica a todas nuestras relaciones cercanas.

El psicólogo John Gottman sugiere que, para tener relaciones felices con alguna persona, es necesario aspirar a tener cinco interacciones positivas por cada interacción negativa que se tenga con ella. Enfócate en tus compañeros y/o colegas y piensa en las siguientes preguntas; en cada caso, anota ejemplos específicos.

1. ¿Qué te atrajo de tus compañeros cuando se conocieron?

2. ¿Qué cosas han disfrutado al hacerlas juntos?

3. ¿Qué cosas realmente aprecias de ellos en este momento?

4. ¿Cuáles son sus fortalezas?

Ahora, lo más importante es que cuando estés con tus compañeros, te tomes el tiempo para darte cuenta y reconocer estas cualidades, sus fortalezas, las cosas que ellos hacen que realmente aprecies, así como los momentos agradables que han compartido.

Piensa en estas declaraciones:

“Realmente me encanta cuando ellos…”
“Son tan buenos para…”
“Viéndolos hacer…, me recuerda ese fantástico día cuando nosotros…”

Aunque realizar dicho análisis con todas las personas que conoces resulta poco práctico, puedes usar los mismos principios para mejorar tus relaciones en general. Por ejemplo, antes de pasar tiempo con alguien tómate un momento para pensar en aquellas cosas que te gustan, aprecias o admiras de esa persona o cómo te hacen sentir bien. Así mismo, después de pasar tiempo con esa persona, piensa en las cosas que apreciaste o lo que disfrutaste del tiempo que pasaron juntos.

Fuente	Basado en catálogo de actividades para profesores.

Práctica 10

Nombre de la práctica	Las fortalezas se muestran en nuestras historias.
Descripción de la práctica	Reflexionarás sobre las fortalezas de carácter que aplicaste en una situación.
Palabras clave	Fortalezas de carácter.
Instrucciones para el aprendedor	Antes de comenzar el ejercicio, ¿sabes cuáles son las fortalezas de carácter? Consulta la descripción de las 24 fortalezas de carácter en la siguiente liga: El **siguiente enlace es externo a la Universidad Tecmilenio, al acceder a este considera que debes apegarte a sus términos y condiciones.** http://www.viacharacter.org/www/Character-Strengths/VIA-Classification Luego de que leas cuáles son las fortalezas de carácter, realiza lo que se pide a continuación: 1. Describe detalladamente mediante un texto una anécdota en la que hayas llevado a cabo alguna acción de la mejor manera posible, o bien, que hayas actuado por encima de lo ordinario. Procura enfocarlo al entorno laboral. 2. Puede ser cualquier suceso que te haya marcado por la manera en que te desenvolviste. 3. Señala en tu descripción, ¿qué ocurrió?, ¿qué papel jugaste en el suceso?, ¿qué acciones llevaste a cabo que fueron de utilidad para ti y para los demás? 4. Luego de que hayas terminado de escribir, lee tu texto y subraya las palabras y oraciones que te den una idea sobre cómo usaste cualquiera de las 24 fortalezas de carácter. 5. Observa y clasifica cuáles son las fortalezas que usaste en tu anécdota. Reflexiona sobre el

	impacto que estas pueden tener en tu desempeño cotidiano.
Fuente	Niemiec, R. (2016). *How to Assess Your Strengths: 5 Tactics for Self-Growth*. Recuperado de https://www.psychologytoday.com/us/blog/what-matters-most/201603/how-assess-your-strengths-5-tactics-self-growth

Práctica 11

Nombre de la práctica	Tus fortalezas en los ojos del otro.
Descripción de la práctica	En la práctica podrás reflexionar sobre la percepción que otros tienen sobre tus fortalezas de carácter.
Palabras clave	Fortalezas de carácter
Instrucciones para el aprendedor	¿Recuerdas alguna ocasión en la que hablaste con algún colega y este te reveló algo positivo que piensa de ti? Cuando esto ocurre usualmente deja huella en nuestros comportamientos y acciones, pues nos damos cuenta de que las personas tienen percepciones sobre nuestras fortalezas que nosotros mismos no vislumbramos. Haz lo siguiente: 1. Piensa sobre alguna vez que algún compañero de trabajo te compartió lo que piensa de ti y que te haya sorprendido. 2. Piensa, ¿qué fue lo que te llamó más la atención?, ¿qué fortalezas vio en ti que pensaste no tenías tan desarrolladas? 3. Por último, señala en un texto por qué consideras que esta revelación te causó tanto impacto, así como la manera en que te ayudó a cultivar tus fortalezas de carácter.
Fuente	Niemiec, R. (2016). *How to Assess Your Strengths: 5 Tactics for Self-Growth*. Recuperado de https://www.psychologytoday.com/us/blog/what-matters-most/201603/how-assess-your-strengths-5-tactics-self-growth

Práctica 12

Nombre de la práctica	Plantea tus objetivos como metas de aproximación y replantea tus metas de evitación.
Descripción de la práctica	Con base a lo que plantea Grenville, en la práctica podrás definir diferentes tipos de metas y encontrar la mejor manera de conseguirlas.
Palabras clave	Objetivos, metas, planes.
Instrucciones para el aprendedor	La autora Bridget Grenville-Cleave (2012) comenta que en el establecimiento de metas es importante distinguir los tipos de metas que hay y menciona dos: **1. Metas de aproximación (approach):** Son las metas con resultados positivos (deseables, placenteros, benéficos o que nos gustaría tener), y hacia las cuales trabajamos. **2. Metas de evitación (avoidance):** Son las metas con resultados negativos (indeseables, dolorosos, dañinos, o nos disgustan), y en las cuales trabajamos para evitarlas. Ejemplo: **Meta de aproximación:** Ser más eficiente. Ser amigable y extrovertido en reuniones. Asumir el rol de líder en el trabajo. **Meta de evitación:** Dejar de aplazar. Dejar de ser tan tímido en las reuniones. No pasar desapercibido en el trabajo. Las investigaciones que se han realizado respecto a estos tipos de metas muestran que perseguir metas de evitación resulta en un detrimento del bienestar. Estos descubrimientos sugieren que el establecer metas de aproximación o replantear las metas de evitación es benéfico.

	Reflexiona: ¿Qué tipo de metas te has planteado tú? ¿Hay algunas metas que puedas replantear en una forma más positiva? ¿Cuándo las tendrás listas?
Fuente	*Secretos para el establecimiento de metas, tomado de:* Grenville, B. (2012). *GOAL-SETTING SECRETS*. Recuperado de *http://positivepsychologynews.com/news/bridget-grenville-cleave/2012013120696*