

UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:PROGRAMACIÓN ORIENTADA A OBJETOS
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 04, LABORATORIO: 02)

SUMILLA

El curso es de naturaleza teórico-práctica y enseña a diseñar programas como conjuntos de objetos que colaboran: cada objeto guarda su estado, protege sus invariantes y ofrece su comportamiento a través de una interfaz. Se organiza en cuatro partes: **objetos y clases** (clases, referencias, encapsulamiento e invariantes); **herencia, polimorfismo e interfaces** (jerarquías de tipos, dynamic dispatch, clases abstractas, interfaces, composición y tipos `sealed`); **programar con tipos** (excepciones, genéricos, el Collections Framework, lambdas y streams); y **diseño** (principios SOLID, patrones de diseño, refactoring, persistencia en archivos y programación orientada a eventos). Se usa Java 25 y los unit tests con JUnit acompañan el curso desde la tercera semana. El laboratorio y un proyecto que crece durante todo el semestre (el sistema de inventario y ventas de una tienda) se desarrollan en un contenedor común para todos los estudiantes.

COMPETENCIAS

- Modela un problema con clases y objetos: identifica responsabilidades, estado e invariantes, y los protege con encapsulamiento (*encapsulation*).
- Explica cómo se representan los objetos en memoria (referencias, heap, garbage collector) y la diferencia entre identidad e igualdad.
- Usa herencia, clases abstractas e interfaces, explica el dynamic dispatch y elige entre herencia y composición con criterio (principio de sustitución de Liskov).
- Escribe código robusto y reutilizable con excepciones, genéricos (*generics*), el Collections Framework, lambdas y streams.
- Aplica principios de diseño (SOLID, bajo acoplamiento, alta cohesión) y patrones de diseño (*design patterns*) clásicos, y reconoce cuándo no hacen falta.
- Escribe unit tests con JUnit y refactoriza con ellos como red de seguridad, en un proyecto que crece durante todo el semestre.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas	Eck
I	Objetos y clases	1–4	Cap. 1.5, 2, 4, 5.1–5.4, 7
II	Herencia, polimorfismo e interfaces	5–8	Cap. 5.5–5.7
III	Programar con tipos	9–12	Cap. 3.7, 4.5, 5.8, 8, 10
IV	Diseño	13–16	Cap. 6, 9.4, 11, 13

Los temas que el libro no cubre (lo nuevo de Java 25, tipos `sealed` y pattern matching, principios SOLID, patrones de diseño y refactoring) se estudian con los apuntes del curso y la documentación oficial de Java.

PROGRAMACIÓN SEMANAL

Cada semana comprende dos sesiones de teoría de 2 horas (A y B) y una sesión de laboratorio de 2 horas. Las secciones indicadas del libro de Eck son lectura previa obligatoria; las marcadas como *apuntes* se leen en los apuntes del curso. Las sesiones marcadas con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Sesión A	Sesión B	Laboratorio
1	Presentación del curso. De C a Java (sec. 1.3, 2.1–2.3, 2.6, 3.8)	Objetos como estado y comportamiento (sec. 1.5, 5.1–5.2)	Entorno de trabajo: Docker, `javac`, `java`, `jshell`, `make`. *P0 publicado*
2	Clases, campos, métodos y constructores (sec. 5.1–5.2, 5.6.1)	Referencias, heap y `null`. Identidad e igualdad (sec. 5.1.2, 5.2.3)	Primeras clases y depuración
3	Encapsulamiento e invariantes (sec. 5.1.3, 8.2)	`static`, paquetes e inmutabilidad. Records (sec. 4.2, 4.6, 7.4)	Unit tests con JUnit. *Entrega P0, publicado P1*
4	Arreglos y `ArrayList` (sec. 3.8, 7.1–7.3)	Diseñar con clases: responsabilidades, diagramas UML de clases (sec. 5.3.4, 5.4)	Modelar un dominio con pruebas
5	Herencia: `extends`, `super`, override (sec. 5.5.1–5.5.3, 5.6)	La clase `Object`: `toString`, `equals`, `hashCode` (sec. 5.3.2, 10.1.6, 10.3.3)	Jerarquías y el contrato de `equals`
6	Polimorfismo y dynamic dispatch (sec. 5.5.3–5.5.4)	Clases y métodos abstractos (sec. 5.5.5–5.5.6)	Polimorfismo en el proyecto. *Entrega P1, publicado P2*
7	Interfaces (sec. 5.7)	Composición frente a herencia. Principio de sustitución de Liskov (apuntes)	Interfaces y test doubles
8	Tipos `sealed` y pattern matching (sec. 3.6.5; apuntes). Repaso	**Examen parcial**	Repaso con ejercicios
9	Excepciones: lanzar, capturar, propagar (sec. 3.7, 8.3)	Robustez: excepciones checked y unchecked, try-with-resources, contratos (sec. 8.1–8.4)	Manejo de errores y pruebas de excepciones. *Entrega P2, publicado P3*
10	Genéricos: clases y métodos (sec. 10.1, 10.5.1–10.5.2)	Wildcards y type erasure (sec. 10.5.3–10.5.4)	Una colección genérica propia

Sem.	Sesión A	Sesión B	Laboratorio
11	Colecciones: `List`, `Set`, `Map` (sec. 10.2–10.3)	`Comparable`, `Comparator` e iteradores (sec. 10.1.5, 10.4)	Elegir la colección adecuada
12	Lambdas, interfaces funcionales y referencias a métodos (sec. 4.5)	Streams (sec. 10.6)	Reportes con streams. *Entrega P3, publicado P4*
13	Principios de diseño: cohesión, acoplamiento, SOLID (sec. 4.1; apuntes)	Refactoring guiado por pruebas (apuntes)	Refactorizar el proyecto
14	Patrones de diseño I: Strategy y Observer (apuntes; sec. 13.3.2)	Patrones de diseño II: Factory y Composite (apuntes; sec. 9.4.3)	Patrones en el proyecto
15	Archivos y E/S: `java.nio.file`, streams. Decorator (sec. 11.1–11.2)	Persistencia simple: CSV y serialización (sec. 11.3)	Guardar y cargar el estado. *Entrega P4, publicado P5*
16	Programación orientada a eventos. Interfaces gráficas (*) (sec. 1.2, 6.1, 6.4)	Cierre: diseño de una aplicación completa. Repaso	Presentación de proyectos. *Entrega P5*
17	**Examen final**		

METODOLOGÍA

- **Teoría:** clases expositivas con discusión de casos, ejercicios resueltos en clase y programación en vivo. El estudiante lee las secciones indicadas antes de cada sesión.
- **Laboratorio:** prácticas guiadas en un contenedor común para todos los estudiantes, con el JDK 25 y JUnit, que funciona igual en Linux, Windows y macOS. Cada práctica se entrega con sus pruebas.
- **Proyecto:** un solo proyecto que crece durante el semestre en seis entregas, cada una con pruebas automáticas que deben seguir pasando en las entregas siguientes.
- Los términos técnicos se usan en inglés cuando así los dicen los programadores y así aparecen en la bibliografía y en la documentación (override, garbage collector, unit test). Los que de verdad se usan en español (clase, herencia, excepción) se presentan en español, con su equivalente en inglés la primera vez.

Proyecto: Bodega, sistema de inventario y ventas

N.º	Entrega	Lo que agrega	Semanas
P0	Calentamiento	Programas cortos en Java; la primera clase del proyecto	1–3
P1	Inventario	Clases encapsuladas con invariantes, `ArrayList`, pruebas JUnit	3–6
P2	Catálogo	Jerarquía de productos, polimorfismo e interfaces (precios y descuentos)	6–9
P3	Ventas	Excepciones, genéricos y colecciones (`Map`, `Set`)	9–12
P4	Reportes	Lambdas y streams; refactoring con principios SOLID	12–15
P5	Aplicación	Patrones de diseño, persistencia en archivos e interfaz de línea de comandos	15–16

POLÍTICAS

- **Trabajo en parejas:** P0 es individual. P1–P5 se pueden hacer en parejas, y la pareja se mantiene durante el semestre.
- **Integridad académica:** se puede discutir ideas, pero el código entregado debe ser propio. Se usan herramientas de detección de similitud. Copiar soluciones publicadas en internet cuenta como plagio.
- **Explicar el propio trabajo:** en cualquier entrega, laboratorio o examen se le puede pedir al estudiante que explique de forma oral su código. Cualquier integrante de la pareja debe poder explicar cualquier parte del proyecto.

Uso de herramientas

El objetivo del curso es que el estudiante aprenda a diseñar y escribir código orientado a objetos, y eso solo pasa si lo escribe él mismo. Por eso las herramientas que escriben código por el estudiante no se permiten; las que ayudan a entender, buscar o revisar, sí.

Herramienta	En este curso
Mensajes del compilador (`javac -Xlint:all`), de la JVM y del depurador; `jshell`	Permitidos en el proyecto y los laboratorios. Leerlos es parte del curso.
Documentación oficial (API de Java, JLS, dev.java, JUnit) y búsqueda en la web	Permitidas para consultar. Copiar una solución completa encontrada en la web cuenta como plagio.
Autocompletado del editor que sugiere nombres de métodos, campos o `import`	Permitido. No se permite el que completa métodos o clases enteros.
Correctores de ortografía, traductores y herramientas de accesibilidad	Permitidos, también para leer el libro y la documentación en inglés.
Formateadores de código y linters	Permitidos.
IA generativa para pedir explicaciones de conceptos, mensajes de error o código del curso, sin que genere código para el estudiante	Permitida para estudiar. No se permite en los exámenes.
IA generativa que escribe código (asistentes en el editor, chats que devuelven código)	No se permite en el proyecto, los laboratorios ni los exámenes. Entregar código producido así cuenta como plagio.

Si el estudiante usó una herramienta permitida de forma que influyó en su entrega (una explicación de IA generativa, una página web que le mostró cómo resolver un problema), lo declara en el `README.md` de la entrega: qué herramienta, para qué y qué parte del trabajo tocó. Declararlo no es una falta; no declararlo sí cuenta como falta a la integridad académica.

BIBLIOGRAFÍA

Texto principal

1. Eck, D. J. (2022). *Introduction to Programming Using Java* (versión 9). Licencia CC BY-NC-SA 4.0. Disponible gratuitamente en `https://math.hws.edu/javanotes/`

Textos complementarios

2. Downey, A. B. y Mayfield, C. (2019). *Think Java* (2.ª ed.). O'Reilly. Disponible gratuitamente en `https://greenteapress.com/wp/think-java-2e/`
3. Bloch, J. (2018). *Effective Java* (3.ª ed.). Addison-Wesley.
4. Gamma, E., Helm, R., Johnson, R. y Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

5. Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code* (2.ª ed.). Addison-Wesley.

Recursos en línea

- Tutoriales oficiales de Java: `https://dev.java/learn/`
- Documentación de la API de Java 25: `https://docs.oracle.com/en/java/javase/25/docs/api/`
- Especificación del lenguaje Java, Java SE 25: `https://docs.oracle.com/javase/specs/jls/se25/html/`
- Guía de usuario de JUnit: `https://docs.junit.org/current/user-guide/`
- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.