Disposición CD EPeiT - 2 / 2026

JOSÉ C. PAZ, 06 de marzo de 2026

VISTO,

El Estatuto de la UNIVERSIDAD NACIONAL DE JOSÉ CLEMENTE PAZ aprobado por Resolución MINISTERIO DE EDUCACIÓN Nº 584 del 17 de marzo de 2015, el REGLAMENTO DE FUNCIONAMIENTO DEL CONSEJO DEPARTAMENTAL DE ECONOMIA, PRODUCCION E INNOVACION TECNOLOGICA, aprobado por Disposicion Nº 01 del citado CONSEJO Nº 01 del 26 de junio de 2020, el Acta de la sesion N° 55 y el Expediente Digital N° 109/2026 del Registro de ésta UNIVERSIDAD NACIONAL DE JOSÉ CLEMENTE PAZ, y

CONSIDERANDO:

Que por el Expediente mencionado en el VISTO tramitan las propuestas de los programas de la LICENCIATURA EN PRODUCCION Y DESARROLLO DE VIDEOJUEGOS correspondiente a la siguiente asignatura: (21.11) FUNDAMENTOS DE LA PROGRAMACIÓN II.

Que es competencia de este CONSEJO DEPARTAMENTAL aprobar y supervisar los programas curriculares de las carreras a su cargo, garantizando que aquellos se ajusten a los contenidos mínimos definidos en los correspondientes Planes de Estudios.

Que habiendo sido puestos a consideración del CONSEJO DEPARTAMENTAL en la Sesión N° 55, de carácter ordinaria, registrada en el Acta Nº 55 del 26 de febrero de 2026, dicho Cuerpo Colegiado compartió los términos y contenidos del referido instrumento, por lo que resulta necesario aprobar los respectivos programas de las asignaturas detalladas.

Que la presente medida se adopta en ejercicio de las atribuciones conferidas por los artículos 77 inciso f), del Estatuto de la UNIVERSIDAD, y 1º inciso d) del Reglamento de Funcionamiento de este consejo departamental.

Por ello,

EL CONSEJO DEPARTAMENTAL DE ECONOMÍA, PRODUCCIÓN E INNOVACIÓN TECNOLÓGICA

DE LA UNIVERSIDAD NACIONAL DE JOSÉ CLEMENTE PAZ

DISPONE:

ARTÍCULO 1º.- Apruebanse los programas de la Carrera LICENCIATURA EN PRODUCCION Y DESARROLLO DE VIDEOJUEGOS que se adjunta como Anexo a la presente, correspondientes a la siguiente asignatura: (21.11) - FUNDAMENTOS DE LA PROGRAMACIÓN II.

ARTÍCULO 2º.- Establécese que el programa aprobado precedentemente, tendrá DOS (2) años de vigencia, contados a partir del semestre siguiente al de su aprobación.

ARTÍCULO 3º.- Regístrese, comuníquese, publiquese en el Boletín Oficial de la UNIVERSIDAD NACIONAL DE JOSÉ CLEMENTE PAZ y cumplido, archivese.

Lic. Santiago Mónaco

Director

Dirección de Órganos de Gobierno

Mag. Pablo López

Director

Departamento de Economía, Producción

e Innovación Tecnológica

Archivos adjuntados

Nombre del archivo
PROGRAMA_(21.11).pdf

PROGRAMA UNIDAD CURRICULAR

Unidad Académica	**DEPARTAMENTO DE ECONOMÍA, PRODUCCIÓN E INNOVACIÓN TECNOLÓGICA**
Carrera/s	**LICENCIATURA EN PRODUCCIÓN Y DESARROLLO DE VIDEOJUEGOS**
Plan de Estudios	**Res. (CS) Nº 63/2021**

1. Datos sobre la unidad curricular

Nombre	FUNDAMENTOS DE LA PROGRAMACIÓN II	**Código**	21.11
Modalidad	Presencial	**Régimen**	Cuatrimestral
Equipo responsable	**Gustavo Del Dago**		
Año y mes de presentación del programa	**Febrero 2026**		

2. Carga horaria

Horas de clase semanales	4		
Horas de clase totales	64	Horas totales teóricas	
		Horas totales prácticas	
		Otras horas totales (laboratorio, trabajo de campo, etc.).	

3. **Unidades correlativas** precedentes en el Plan de Estudios

Denominación	Código
FUNDAMENTOS DE LA PROGRAMACIÓN I	21.09

4. **Contenidos mínimos** según Plan de Estudios

Introducción a la Programación Orientada a Objetos: Clases, Herencias, Encapsulamiento, Polimorfismo. Complejidad computacional. Notación O (). Estructuras de datos: estructuras, escritura/lectura, persistencia. Relevamiento de motores de videojuegos (Game Engine). Selección y géneros específicos de motores de juegos. Recursos comerciales y código abierto. Entornos de desarrollo. Tecnología de diseño de niveles. Tecnología de integración de recursos (Assets). Motores gráficos y de animación. Periféricos de jugabilidad. Innovación tecnológica. Nuevos periféricos y su inclusión en el diseño de videojuegos.

5. **Fundamentación**

De acuerdo con el plan de estudios de la *Licenciatura en Producción y Desarrollo de Videojuegos*, *Fundamentos de la programación II* constituye el segundo espacio disciplinar perteneciente al campo de la programación. Por lo anterior, la propuesta se orienta a estudiantes que poseen conocimientos básicos sobre los conceptos, técnicas y herramientas fundamentales de la programación en el paradigma estructurado, nociones de algoritmia elemental, capacidad para elaborar estrategias de solución a problemas computacionales de complejidad moderada empleando la técnicas de división en subproblemas y un mínimo de fluidez en la lectura y escritura de código de programación a fines de comprender programas elaborados por otras personas y escribir programas funcionales que comuniquen con claridad las estrategias de solución empleadas.

En esta unidad curricular se abordan los fundamentos del diseño y programación bajo el paradigma orientado a objetos. La noción de algoritmo o estrategia de solución de problemas computacionales constituye uno de los ejes en torno al que se estructura la propuesta. Se presenta el lenguaje modelado unificado (UML) en tanto herramienta para expresar y comunicar modelos orientados a la solución de los problemas y desafíos propuestos. Durante el trayecto se recuperarán los conocimientos, técnicas y herramientas del lenguaje formal construidos y desarrollados en el espacio de *Fundamentos de la programación I* mediante puestas en práctica orientadas a evidenciar algunos límites del paradigma estructurado cuando se intenta abordar desafíos de mayor complejidad. Se trata de una problematización deliberada, una estrategia didáctica con intención de favorecer la búsqueda de soluciones y respuestas en el marco del paradigma orientado a objetos (objeto de estudio central de esta unidad curricular). Se estudian los pilares fundamentales del paradigma orientado a objetos. En *Fundamentos de la programación I*, tanto la capacidad de comunicar de manera clara y legible empleando código de programa textual (lenguaje formal) como la estructuración modularizada de las soluciones, constituían los criterios más importantes a la hora de valorar las producciones. En *Fundamentos de la Programación II* se agrega una serie de criterios que complementan a los anteriores. Se trata de modelar soluciones empleando las ideas de abstracción, delegación, encapsulamiento y polimorfismo. Con lo anterior se pretende favorecer la concepción de soluciones

dotadas de las características propias del paradigma orientado a objetos. Soluciones que tomarán formas concretas mediante diagramas y código de programa. Para los primeros se emplea el Lenguaje de Modelado Unificado (UML) mientras que para los segundos se emplea el lenguaje de programación C#. La elección de las mencionadas herramientas obedece a dos cuestiones centrales. En primer lugar, se trata de herramientas con amplia difusión tanto en el campo computacional académico como en el productivo. En segundo lugar, se trata de herramientas de uso extendido en el campo del diseño y programación de videojuegos. Los denominados "motores de videojuegos" más populares en la actualidad constituyen ejemplos notables respecto de la adopción por parte del sector productivo tanto de los conceptos del paradigma orientado a objetos como de los mencionados lenguajes formales (UML y C#).

Aún, cuando esta propuesta formativa, atendiendo al perfil de los/as egresados/as, no está orientada a formar especialistas en el campo de la programación, el plan de trabajo se estructura con el propósito de convertir los temas centrales de la disciplina, en contenidos adecuados a los lineamientos y estructura del plan de estudios de la carrera. Contenidos que dotarán a los/as estudiantes con bases sólidas para el abordaje de futuras materias y disciplinas del plan de estudios (Taller de prototipado digital constituye un ejemplo paradigmático). La importancia relativa dentro del conjunto de fundamentos disciplinares, las aplicaciones prácticas en el campo del desarrollo de videojuegos y el nivel de accesibilidad para estudiantes de una unidad curricular de carácter introductorio conforman el conjunto de criterios que operaron en la selección y adecuación de los temas, la elaboración de los contenidos y el diseño de las actividades y desafíos propuestos.

En cuanto al enfoque didáctico, se sostiene un abordaje mediante lenguajes textuales a fines de avanzar rápidamente en las cuestiones relacionadas con el manejo simbólico propio de los lenguajes formales. El trayecto presenta de manera alternada momentos teóricos y momentos prácticos. Se propone un ambiente de trabajo donde las competencias técnicas (propiciadas a partir de la resolución de ejercicios o desafíos de programación) se interrelacionan con las conceptualizaciones (suscitadas por la reflexión conjunta sobre el proceso de elaboración de soluciones y su posterior análisis).

En *Fundamentos de programación I* se promueven intercambios mediados por "código fuente" entendido en tanto codificación de determinada estrategia de solución empleando lenguaje formal. En *Fundamentos de la programación II*, que debe ser considerado como un trayecto complementario del anterior, se agrega un lenguaje gráfico (UML) a fines de enriquecer el uso de múltiples sistemas de representación semiótica sobre los mismos objetos de estudio y enriquecer la comunicación e intercambio de ideas. Un proceso que, además de favorecer la claridad de las producciones, favorece el razonamiento en un nivel de abstracción más alto, algo imprescindible cuando se afronta el desafío de elaborar soluciones mediante una comprensión más acabada tanto del problema enfrentado como de las posibles soluciones y, eventualmente, la capacidad de establecer criterios valorativos que permitan comparar soluciones alternativas.

6. Objetivos

Que la/el estudiante:

- Comprenda los fundamentos del **diseño** y la **programación** de soluciones bajo el **paradigma orientado a objetos**.
- Elabore **modelos computacionales** que permitan resolver problemas de complejidad media relacionados con **programas interactivos en el espacio bidimensional**, a partir de la formulación de **abstracciones adecuadas**, **el tratamiento de múltiples objetos agrupados en colecciones** y participando en **relaciones de composición, asociación y/o herencia.**
- Construya **implementaciones concretas** de los modelos computacionales elaborados empleando el lenguaje **programación C#.**
- Elabore **casos y escenarios de prueba** elementales que permitan evaluar el funcionamiento de las soluciones.
- Desarrolle habilidades relacionadas con la escritura de código atendiendo a las mejores prácticas **(código limpio)**, favoreciendo la legibilidad de las soluciones y con el uso de **lenguaje de modelado unificado (UML).**
- Conozca y valore las ventajas de emplear **patrones de diseño**, especialmente aquellos aplicables al diseño y programación de videojuegos.

7. Contenidos (organizados por unidades).

Unidad 1: Diseño y programación orientada a objetos.

Introducción al paradigma orientado a objetos. Lenguaje unificado de modelado (UML). Principios fundamentales de modelado (Abstracción, delegación, encapsulamiento, polimorfismo). Lenguajes de programación orientada a objetos. Ambientes de desarrollo y ejecución. Objetos (Atributos, métodos y comportamientos). Variables y funciones miembro. Observador, ambiente, referencias y mensajes. Declaración y definición de clases. Relaciones de agregación, composición y herencia. Clases abstractas. Clases y métodos virtuales.

Unidad 2: Colecciones.

Tipos de datos genéricos. Colecciones de longitud fija (Arreglos n-dimensionales). Colecciones de longitud variable (Listas). Diccionarios. Estructuras de control e iteradores. Noción de orden superior. Funciones Lambda. Tipos enumerados y extensiones.

Unidad 3: Mecanismos de abstracción. Buenas prácticas de programación y testeo automatizado.

Espacios de nombres. Tipos y métodos genéricos. Interfaces. Sobrecarga de operadores. Código limpio

(Legibilidad, facilidad de mantenimiento, escalabilidad y claridad). Guías de estilo. Comentarios. Excepciones y manejo de errores. Introducción al testeo unitario automatizado.

Unidad 4: Patrones de diseño.

Introducción a los patrones de diseño. Clasificación de patrones de diseño: patrones de creación, patrones estructurales y patrones de comportamiento. Principios de diseño (SOLID): Responsabilidad única, Abierto-Cerrado, Sustitución de Liskov, Segregación de interfaces e Inversión de dependencias. Relaciones entre clases abstractas e interfaces. Estudio e implementación de los patrones: Abstract Factory (Fábrica abstracta), Builder (Constructor), Prototype (Prototipo), Singleton (Único), Composite (Compuesto), Facade (Fachada). Command (Orden), Iterator (Iterador), Observer (Observador) y State (Estado).

8. Bibliografía obligatoria y complementaria (organizada por unidades).

Bibliografía obligatoria:

Unidad 1.

- Ceballos Sierra, F. (2011). Capítulo 2 - Introducción a C#. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.19-32), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 3 - Introducción a la POO. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.33-60), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 4 - Elementos del lenguaje. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.61-82), Madrid, España: RA-MA Editorial.
- Joyanes Aguilar, L. (2020). Capítulo 15 - Tipos abstractos de datos, objetos y modelado con UML 2.5.1. En *Fundamentos de programación Algoritmos, estructura de datos y objetos (5ª ed.)* (pp.565-592), Ciudad de México, México: McGraw-Hill Interamericana Editores.
- Joyanes Aguilar, L. (2020). Capítulo 16 - Diseño de clases y objetos: representaciones gráficas en UML. En *Fundamentos de programación Algoritmos, estructura de datos y objetos (5ª ed.)* (pp.593-632), Ciudad de México, México: McGraw-Hill Interamericana Editores.
- Joyanes Aguilar, L. (2020). Capítulo 17 - Relaciones entre clases: delegaciones, asociaciones, agregaciones, herencia. En *Fundamentos de programación Algoritmos, estructura de datos y objetos (5ª ed.)* (pp.633-670), Ciudad de México, México: McGraw-Hill Interamericana Editores.
- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 1 - Perspectiva general de UML. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.3-14), Madrid, España: Pearson Educación.
- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 2 - La naturaleza y propósitos de los modelos. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.15-22), Madrid, España: Pearson Educación.
- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 3 - Un paseo por UML. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.25-42), Madrid, España: Pearson Educación.

- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 4 - La vista estática. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.43-62), Madrid, España: Pearson Educación.
- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 13 - El entorno de UML. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.107-112), Madrid, España: Pearson Educación.

Unidad 2.

- Ceballos Sierra, F. (2011). Capítulo 8 - Matrices. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.177-224), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 9 - Más sobre métodos y colecciones. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.451-464), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 13 - Tipos y métodos genéricos. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.451-464), Madrid, España: RA-MA Editorial.
- Deitel, H., Deitel, P. (2007). Capítulo 8 - Arreglos. En Cómo programar en C# *(4ª ed.)* (pp.215-262), Ciudad de México, México: Pearson Educación.
- Deitel, H., Deitel, P. (2007). Capítulo 8 - Arreglos. En Cómo programar en C# *(4ª ed.)* (pp.215-262), Ciudad de México, México: Pearson Educación.
- Deitel, H., Deitel, P. (2007). Capítulo 25 - Genéricos. En Cómo programar en C# *(4ª ed.)* (pp.999-1016), Ciudad de México, México: Pearson Educación.
- Deitel, H., Deitel, P. (2007). Capítulo 26 - Colecciones. En Cómo programar en C# *(4ª ed.)* (pp.263-1040), Ciudad de México, México: Pearson Educación.

Unidad 3.

- Ceballos Sierra, F. (2011). Capítulo 10 - Clases, espacios de nombres y estructuras. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.265-348), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 11 - Operadores sobrecargados. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.349-380), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 12 - Clases derivadas e interfaces. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.381-450), Madrid, España: RA-MA Editorial.
- Ceballos Sierra, F. (2011). Capítulo 14 - Excepciones. En *Microsoft C#: Curso de programación (2ª ed.)* (pp.465-464), Madrid, España: RA-MA Editorial.
- Deitel, H., Deitel, P. (2007). Capítulo 11 - Polimorfismo, interfaces y sobrecarga de operadores. En Cómo programar en C# *(4ª ed.)* (pp.341-384), Ciudad de México, México: Pearson Educación.
- Joyanes Aguilar, L. (2020). Capítulo 18 - Ingeniería de software y metodología de la programación. En *Fundamentos de programación Algoritmos, estructura de datos y objetos (5ª ed.)* (pp.673-708), Ciudad de México, México: McGraw-Hill Interamericana Editores.
- Martin, R. (2012). Capítulo 1 - Código limpio. En Código Limpio. Manual de estilo para el desarrollo ágil de software (pp.28-43), Madrid, España: Ediciones Anaya Multimedia.
- Martin, R. (2012). Capítulo 2 - Nombres con sentido. En Código Limpio. Manual de estilo para el desarrollo ágil de software (pp.44-59), Madrid, España: Ediciones Anaya Multimedia.

- Martin, R. (2012). Capítulo 6 - Objetos y estructuras de datos. En Código Limpio. Manual de estilo para el desarrollo ágil de software (pp.124-133), Madrid, España: Ediciones Anaya Multimedia.
- Martin, R. (2012). Capítulo 7 - Procesar errores. En Código Limpio. Manual de estilo para el desarrollo ágil de software (pp.134-145), Madrid, España: Ediciones Anaya Multimedia.
- Martin, R. (2012). Capítulo 9 - Pruebas de unidad. En Código Limpio. Manual de estilo para el desarrollo ágil de software (pp.156-169), Madrid, España: Ediciones Anaya Multimedia.
- Martin, R. (2012). Capítulo 10 - Clases. En Código Limpio. Manual de estilo para el desarrollo ágil de software (pp.160-187), Madrid, España: Ediciones Anaya Multimedia.

Unidad 4.

- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (2003). Capítulo 3 - Patrones de Creación. En *Patrones de Diseño. Elementos de software orientado a objetos reutilizable* (pp.79-127), Madrid, España: Pearson Educación.
- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (2003). Capítulo 4 - Patrones Estructurales. En *Patrones de Diseño. Elementos de software orientado a objetos reutilizable* (pp.131-201), Madrid, España: Pearson Educación.
- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (2003). Capítulo 5 - Patrones de Comportamiento. En *Patrones de Diseño. Elementos de software orientado a objetos reutilizable* (pp.205-317), Madrid, España: Pearson Educación.
- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 5 - La vista de diseño. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.63-68), Madrid, España: Pearson Educación.
- Rumbaugh, J., Jacobson, I., Booch, G. (2007). Capítulo 7 - La vista de la máquina de estados. En *El Lenguaje Unificado de Modelado. Manual de Referencia (2ª ed.)* (pp.73-84), Madrid, España: Pearson Educación.

Bibliografía complementaria:

Aplicable a todas las unidades del programa

- Albahari, J. (2022). C# 10 in a Nutshell - The Definitive Reference. California, United States of America: O'Reilly Media, Inc.
- Albahari, J., Albahari, B. (2022). C# 10 Pocket Reference Instant Help for C# 10 Programmers. California, United States of America: O'Reilly Media, Inc.

Unidad 4

- Doran, J., Casanova, M. (2017). Game Development Patterns and Best Practices: Better games, less hassle. Birmingham, UK: Packt Publishing Ltd.

Van Horn II, B. (2022). Real-World Implementation of C# Design Patterns. Overcome daily programming challenges using elements of reusable object-oriented software. Birmingham, UK: Packt Publishing Ltd.

9. Metodología de trabajo

Las clases estarán divididas en momentos teóricos y momentos prácticos. Durante los primeros, se presentarán los conceptos correspondientes, se explicarán las técnicas de aplicación y se demostrarán los modos de uso de las herramientas computacionales específicas (ambientes integrados de programación). De este modo, se pretende desacoplar los conceptos abordados (objeto de estudio) de las diversas implementaciones (lenguajes formales), aplicaciones (técnicas) y tecnologías concretas (herramientas de programación). Los espacios de práctica se dispondrán de manera que habiliten las primeras instancias de puesta en acción de los temas bajo estudio. Si bien las instancias de práctica están concebidas para favorecer mayores niveles de apropiación de los conceptos, tienen el potencial de promover habilidades relacionadas con las técnicas y herramientas del campo de la programación.

En cada uno de los encuentros se presentarán problemas y desafíos con el propósito de ofrecer instancias de puesta en acción de los conceptos, técnicas y herramientas. Se consignarán actividades optativas de resolución domiciliaria cuyas soluciones serán discutidas al inicio de la clase posterior a aquella en que hayan sido consignadas. Las actividades domiciliarias cuya demanda excede normalmente el tiempo disponible durante las clases, tendrán carácter integrador y, justamente por ello, constituirán una excelente forma de preparación para las instancias de exámen.

Atendiendo a los intereses y necesidades de las personas destinatarias de la propuesta, el dominio de los problemas y desafíos seleccionados pertenece al campo de los videojuegos. El dominio de las soluciones, por su parte, permitirá abordar la totalidad de los contenidos mínimos de acuerdo al plan de estudios.

El espacio digital (aula en el campus de la Universidad y/o plataformas de programación disponibles en línea) se constituyen como punto de encuentro propicio para el intercambio de ideas y reflexiones sobre los temas abordados. El mismo espacio se dispone como un canal adicional que favorezca el trabajo colectivo y colaborativo. Se fomentarán instancias de revisión entre pares y esquemas de estudio basados en la ayuda mutua.

10. Evaluación

Por su carácter teórico-práctico, el espacio de Fundamentos de la Programación II se desarrolla en el ámbito del laboratorio de computación y no admite examen libre. Por esa misma singularidad, se propone una evaluación constante de tipo formativo. La evaluación formativa tendrá lugar en los momentos destinados a las actividades prácticas y a la resolución de problemas y desafíos. Las actividades evaluables en los dos exámenes parciales se orientan a la puesta en práctica de los conceptos fundamentales durante la resolución de problemas computacionales de complejidad media (adecuados a los objetivos de este trayecto formativo). Las soluciones deberán evidenciar la aplicación de los conceptos, técnicas y herramientas conceptuales y de los lenguajes formales propias del paradigma orientado a objetos. La legibilidad del código fuente, el uso de nombre claros y significativos en los identificadores, la modularización y la definición de funciones y procedimientos, los modelos diseñados mediante abstracciones, encapsulamiento, relaciones de composición, agregación y herencia, el uso e implementación de patrones de diseño adecuados al desafío

planteado, constituyen el conjunto de criterios con los que se evaluarán las producciones en cada una de las instancias de evaluación. Ambas instancias de evaluación parcial tendrán carácter presencial y consistirán en la resolución práctica de problemas y desafíos con las características mencionadas anteriormente. En ambos casos se contará, siguiendo las pautas del Régimen General de Estudios, con la correspondiente instancia de recuperatorio.

Regularidad y aprobación en modalidad presencial, según el Régimen General de Estudios, Resolución 150/2018

Las asignaturas se aprueban mediante:

a) Promoción

b) Examen integrador

c) Examen Final

En cualquiera de los casos se requiere el 75% de asistencia a clase.

Para aprobar la asignatura por promoción se requiere obtener calificaciones no inferiores a 6 (seis) en al menos dos evaluaciones parciales o sus respectivos recuperatorios, y un promedio de 7 (siete) puntos o más.

Para aprobar a través de examen integrador se requiere obtener calificaciones no inferiores a 4 (cuatro) en al menos dos evaluaciones parciales. Esta instancia se desarrolla luego de finalizada la cursada, no requiere inscripción previa y es llevada adelante por el profesor de la comisión, quien indica a cada estudiante los contenidos a evaluar y su modalidad (escrito, oral, trabajo, etc.). El examen integrador se aprueba con 4 (cuatro) puntos.

Los estudiantes que no aprueben por promoción o por examen integrador tendrán la posibilidad de aprobar la asignatura mediante examen final. Para acceder a esta instancia se requiere obtener calificaciones no inferiores a 4 (cuatro) puntos en al menos dos instancias parciales o sus respectivos recuperatorios.

Para más información se sugiere leer los artículos 31 a 39 del Régimen General de Estudios de UNPAZ y la Resolución N°154/2022 y tomar en cuenta cualquier otra normativa que fuera publicada en la web de la UNPAZ.

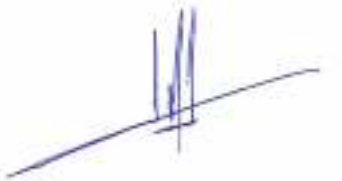
11. Instancias de práctica (si corresponde)

12. Cronograma de actividades	
Semana 1	Presentación de la materia: contenidos mínimos, programa de estudios y plan de trabajo. Unidad 1: Introducción al paradigma orientado a objetos. Lenguaje unificado de modelado (UML). Principios fundamentales del paradigma orientado a objetos (Abstracción y encapsulamiento). Lenguajes de programación orientada a objetos. Ambientes de desarrollo y ejecución. Modelos computacionales. Nociones básicas: objeto, atributo, método, comportamiento, clase. Mecanismos de comunicación: referencias y mensajes.
Semana 2	Unidad 1: Declaración y definición de clases. Variables y funciones miembro. Principios fundamentales del paradigma orientado a objetos (Delegación). Privacidad y alcance de variables. Variables y funciones públicas y privadas. Constructores. Introducción a las relaciones de composición.
Semana 3	Unidad 1: Relaciones de composición. Tipos de datos, clases y referencias. Métodos de acceso y propiedades. Introducción a las relaciones de agregación y herencia. Principios fundamentales del paradigma orientado a objetos (Polimorfismo).
Semana 4	Unidad 1: Relaciones jerárquicas (herencia). Clases abstractas. Clases y métodos virtuales. Clases bases y derivadas. Invocación de constructores. Variables y funciones protegidas.
Semana 5	Unidad 1: Código limpio (Legibilidad, facilidad de mantenimiento, escalabilidad y claridad). Guías de estilo. Comentarios. Tipos de datos genéricos. Colecciones de longitud fija (Arreglos n-dimensionales).
Semana 6	Unidad 2: Arreglos bidimensionales (aplicaciones en el campo de los videojuegos) algoritmos de recorrido. Relaciones de composición. Colecciones homogéneas y heterogéneas. Promoción de tipos de datos.
Semana 7	Unidad 2: Colecciones de longitud variable (Listas y diccionarios). Iteradores y propiedades.
Semana 8	Primer examen parcial.
Semana 9	Unidad 2: Noción de orden superior. Funciones Lambda. Tipos enumerados y extensiones.
Semana 10	Unidad 3: Espacios de nombres. Tipos y métodos genéricos. Interfaces. Sobrecarga de operadores. Relaciones entre clases abstractas e interfaces. Excepciones y manejo de errores. Introducción al testeo unitario automatizado.

Semana 11	Unidad 4: Introducción a los patrones de diseño. Clasificación de patrones de diseño: patrones de creación, patrones estructurales y patrones de comportamiento. Principios de diseño (SOLID): Responsabilidad única, Abierto-Cerrado, Sustitución de Liskov, Segregación de interfaces e Inversión de dependencias.
Semana 12	Recuperatorio del primer examen parcial.
Semana 13	Unidad 4: Estudio e implementación de los patrones: Abstract Factory (Fábrica abstracta), Builder (Constructor), Prototype (Prototipo), Singleton (Único).
Semana 14	Unidad 4: Composite (Compuesto), Facade (Fachada). Command (Orden), Iterator (Iterador), Observer (Observador) y State (Estado).
Semana 15	Unidad 4: Revisión de motores de videojuegos desde la perspectiva del paradigma orientado a objetos. Integración y repaso.
Semana 16	Segundo examen parcial.

A partir de aquí completar únicamente las unidades curriculares con régimen anual	
Semana 17	
Semana 18	
Semana 19	
Semana 20	
Semana 21	
Semana 22	
Semana 23	
Semana 24	
Semana 25	
Semana 26	

Firma del docente/s responsable/s:



Del Dago, Gustavo

Hoja de firmas

Sistema: sudocu
Fecha: 06/03/2026 10:42:21
Cargado por: ROMINA MAROLT

Sistema: sudocu
Fecha: 06/03/2026 10:50:03
Autorizado por: SANTIAGO MONACO

Sistema: sudocu
Fecha: 25/03/2026 20:25:12
Autorizado por: PABLO JULIO LOPEZ