|  |
| --- |
| **CÓDIGO DE ASIGNATURA**  1113 |

ASIGNATURA: Programación Avanzada

JEFE DE CÁTEDRA: Verónica Inés Aubin

AÑO: **2022**

CARGA HORARIA: **10**

MODALIDAD DE DICTADO: “Tipo B”

OBJETIVOS:

**Objetivos Generales:**

1. Capacidad de aplicar los conocimientos en la práctica.
2. Capacidad para organizar y planificar el trabajo.
3. Capacidad para analizar, plantear y resolver problemas.
4. Capacidad de trabajo en equipo.
5. Compromiso para realizar el trabajo con calidad.

**Objetivos Específicos:**

1. Proporcionar una comprensión sólida de los conceptos fundamentales del modelo de objetos.
2. Vincular los contenidos teóricos de la materia con situaciones de la vida real.
3. Identificar, diseñar y desarrollar un algoritmo eficiente para la resolución de una situación problemática dada.
4. Aplicar la técnica de Desarrollo Dirigido por Tests (Test Driven Development o TDD) para el desarrollo del software.
5. Implementar algunos principios de la programación XP (programación Extrema) como parte de la cultura del buen programador.
6. Emplear algunos principios de PSP (Personal Software Process) como parte de la cultura del buen programador.
7. Reconocer e incluir a la Verificación y Validación de Software como parte fundamental en la actividad del programador.
8. Lograr un dominio avanzado del lenguaje de programación Java.
9. Adquirir conocimientos básicos de la programación funcional y lógica.
10. Estimular la integración del estudiante en grupos de trabajo, de forma de potenciar su propio aprendizaje a través de la interacción y cooperación con sus pares.

CONTENIDOS MÍNIMOS:

Diseño Orientado a Objetos. El Lenguajes de Programación JAVA. Tipos de datos abstractos. Estructuras de control. Recursividad. Eventos. Excepciones.

Concurrencia. Estructuras de Datos en JAVA. Diseño y Análisis de Algoritmos.

Concurrencia y Paralelismo. Algoritmos concurrentes, distribuidos y paralelos.

Complejidad computacional. Paradigmas de programación: imperativo, orientado a objetos, funcional y lógico. Incluye TCP.

PROGRAMA ANALÍTICO:

**Unidad I Una Metodología del desarrollo de software (3 clases)**

Metodologías iterativas, incremental, espiralada. Procesos Ágiles: Programación extrema (XP) y Scrum. Métricas del software. Principios de la programación XP; Conceptos de PSP (Personal Software Process); Desarrollo Dirigido por Tests (Test Driven Development o TDD). Una metodología propuesta por la cátedra para el desarrollo de software.

La preparación del lote de prueba. Documentación del Lote de Pruebas. El programa probador. El programa verificador del Input. Ventajas de preparar la prueba antes de comenzar la programación. Pruebas del software: de caja negra, de caja blanca, inspecciones. Ventajas de la inspección del código fuente.

**Unidad II Paradigmas y metodologías de programación (1 clase)**

Paradigmas de programación: Imperativo, Declarativo, Orientado a objetos, Funcional y Lógico. Estilos de programación en la programación Imperativa: Estructurada por procedimientos, modular, con objetos y orientada a objetos. Apoyo de un lenguaje para cada estilo de programación. Diseño Orientado a Objetos. Programación orientada a objetos y lenguajes orientados a objetos.

**Unidad III El Lenguajes de Programación JAVA (7 clases)**

Tipos de datos abstractos. Estructuras de control. Recursividad. Eventos. Excepciones. Concurrencia. Estructuras de Datos en JAVA. Datos y métodos. Métodos static. Constructores. Las clases complejo, vectorMath y matrizMath. Implantación de las clases correspondiente a los TDA básicos: Clase lista, pila y cola.

**Unidad IV Herencia Y Polimorfismo (2 clases)**

Jerarquías de clases: Generalización y Especialización (“es un“). Composición y Agregación. (“tiene un“ y “es parte de“). Miembros protegidos de la clase base. Clases derivadas. Herencia simple y múltiple. Accesibilidad. Clases abstractas. Métodos virtuales. Aplicación de métodos polimórficas. Diagrama de clases usando UML.

**Unidad V Complejidad Computacional (4 clases)**

Medición del tiempo de ejecución de un algoritmo. Orden de un algoritmo. Velocidad de crecimiento del tiempo de ejecución. Función O. Efectos al aumentar la velocidad del computador y/o al aumentar el tamaño del problema. Medición del tiempo de ejecución. Comparaciones de los distintos algoritmos desarrollados en los TTPP.

**Unidad VI Algoritmos (6 clases)**

Diseño y Análisis de Algoritmos. Algoritmos de ordenamientos y sus implantaciones. Comparaciones de los distintos algoritmos en función de sus operaciones básicas. Estabilidad y sensibilidad al input de un algoritmo de ordenamiento. Utilización de estructuras estáticas y dinámicas. Algoritmos de búsqueda. Algoritmos de Grafos: Dijkstra, Floyd, Warshall, Prim, Kruskal y coloreo de grafos. Resolución de problemas reales y ficticios. Concurrencia y Paralelismo. Algoritmos concurrentes, distribuidos y paralelos.

**Unidad VII Taller Programación Básica en Java con WebSphere/Rational (16 clases**)

Conceptos básicos de Java. Conceptos y características de J2ME, J2SE, J2EE. Arquitectura de los distintos frameworks. Concurrencia y Paralelismo. Algoritmos concurrentes, distribuidos y paralelos. Implementación de un cliente pesado en Java. Implementación de un JavaBean. Acceso a datos.

**Unidad VIII Programación Lógica (4 clases)**

Lógica de predicados argumentos, interpretación, reglas, preguntas, variables, ámbito de una variable, corte. Estructura de un programa Prolog. Listas y secuencias finitas. Aplicaciones en grafos. Resolución de caminos mínimos. Equivalencia entre el Prolog y los operadores del TDA conjunto. Equivalencia entre Prolog y consultas SQL.

**Unidad IX. Programación Funcional (1 clase)**

Introducción a la programación funcional. Haskell & Hugs. Tipos y clases. Funciones y operadores.

BIBLIOGRAFÍA:

BIBLIOGRAFÍA BÁSICA

| **Autor** | **Título** | **Editorial** | **Año** | **Edición** |
| --- | --- | --- | --- | --- |
| Horstmann  Cornell | Core Java Volumen I – Fundamentos | Pearson Prentice Hill | 2005 | 7ma edición |
| Deitel -Deitel | Java, cómo programar. 7ma Edición | Pearson Prentice Hill. Séptima edición | 2008 | 2da y 4ta edición |
| Naughton P | Manual de Java | McGraw –Hill | 2002 |  |
| Aho, Hopcroft,j Ullman. | Estructuras de datos y algoritmos. | Addison-Wesley Iberoamericana | 1983 |  |
| Sedgewick | Algoritmos en C++ | Addison-Wesley | 1995 |  |
| D. Arnow- G. Weiss. | Introducción a la Programación en Java | Addison Wesley | 2001 |  |
| FONTELA CARLOS | Orientación a Objetos con Java y UML | NUEVA LIBRERIA | 2003 |  |
| Henrik Kniberg | Scrum y XP desde las trincheras Como hacemos Scrum | http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf | 2007 |  |
| GiannesiniKa  nouiPaseroC  aneghem | PROLOG | Addison Wesley  Iberoamericana | 1989 |  |
| Bruce Eckel | Thinking in Java,  4th. edition | Prentice Hall  http://www.saeedsh.com/resources/Thinking%20in%20Java%204th%20Ed.pdf | 2006 |  |
| Mark Allen Weiss | Estructuras de datos y algoritmos | Addison Wesley  Iberoamericana | 1995 |  |

BIBLIOGRAFÍA COMPLEMENTARIA

| **Autor** | **Título** | **Editorial** | **Año** | **Edición** |
| --- | --- | --- | --- | --- |
| Gamma, Erich and Helm, Richard and Johnson, Ralph and Vlissides, John | Design Patterns Elements of reusable object-oriented sofware | Addison Wesley | 2009 |  |
| Eric Freeman, Kathy Sierra, Bert Bates, Elisabeth Robson | Head first design patterns | O'Reilly Media, Inc. | 2008 |  |
| Miran Lipovaca | Learn You a Haskell for Great Good!: A Beginner's Guide | No Starch Press | 2011 |  |

HABILIDADES QUE SE BUSCA DESARROLLAR EN LOS ESTUDIANTES:

* Mejorar la calidad de los algoritmos creados gracias al mayor conocimiento adquirido.
* Utilización correcta de las estructuras datos para obtener programas más óptimos.
* Resolución de problemas reales aplicando el mejor algoritmo teniendo en cuenta su complejidad computacional y utilización de memoria.
* Resolución de problemas reales aplicando los principios de POO.
* Utilización de diversas herramientas de programación.

COMPETENCIAS:

* Identificar, formular y resolver problemas de ingeniería.
* Utilizar de manera efectiva las técnicas y herramientas de aplicación en la ingeniería informática.
* Adquirir hábitos de pensamiento.
* Desempeñarse de manera efectiva en equipos de trabajo.
* Comunicarse con efectividad.
* Aprender en forma continua y autónoma.

METODOLOGÍA DE LA ENSEÑANZA

Se utilizan metodologías activas de enseñanza, especialmente aprendizaje basado en problemas (ABP). Se presenta cada núcleo temático, introduciendo las ideas fundamentales con la asistencia de ejemplos reales, lo que motiva a los alumnos y permite relacionar los contenidos de la materia con las herramientas de desarrollo que están acostumbrados a utilizar. Luego, se formalizan las definiciones correspondientes para complementar y organizar los contenidos en un marco teórico establecido.

Los contenidos de la asignatura se presentan de forma iterativa e incremental de modo que los estudiantes puedan integrar los nuevos conceptos a los anteriores de forma de lograr aprendizajes significativos.

Se motiva a los estudiantes en el uso de foros internos, para la resolución de dudas tanto de conceptos teóricos como prácticos, permitiendo desarrollar las capacidades de comunicación y afianzar el uso del lenguaje técnico.

Se utiliza la plataforma LOOM para desarrollar la actividad de revisión de a pares como metodología para que los estudiantes adquieran la práctica de inspección de código, la detección de errores y favorecer la adquisición de competencias transversales como crítica y autocrítica. La plataforma LOOM se utiliza además el seguimiento de la evolución de los trabajos de los estudiantes, tanto individual como grupal.

DESCRIPCIÓN DE LA ACTIVIDAD CURRICULAR TANTO PRESENCIAL COMO VIRTUAL

La asignatura tiene un fuerte vínculo entre la teoría y la práctica. La actividad curricular consta de clases teórico – prácticas dictadas en laboratorio (siempre que este esté disponible).

Las clases presenciales se potencian por el uso del uso del cañón junto con el Mimio y el soporte de la pizarra tradicional. Las clases virtuales se potencian gracias al uso de reuniones interactivas en la plataforma Microsoft Teams y los recursos que esta brinda como grupos, pizarras, canales, chat, disco virtual, etc. En ambas modalidades la plataforma MIeL da soporte al dictado de la materia. Permite la distribución de material didáctico desarrollado por los docentes y hacer un seguimiento del avance de los estudiantes en las actividades planteadas, a través del uso de las herramientas de la plataforma: foro, mensajería y tutoría.

Cada curso cuenta con un docente a cargo y uno o dos docentes auxiliares. Los docentes guían a los estudiantes en el abordaje de los contenidos teóricos, supervisan los trabajos de laboratorio y dan soporte a las consultas específicas para resolución de dudas tanto teóricas como prácticas.

Los conceptos teóricos se acompañan de ejemplos prácticos, que muchas veces son desarrollados en forma colaborativa con la participación de los estudiantes. Se muestran ejemplos de resolución de “problemas tipo” de aplicaciones reales.

Los estudiantes cuentan con guías de trabajos prácticos para la resolución de ejercicios que le facilitan la incorporación de los conceptos teóricos abordados.

Esta asignatura incluye un Trabajo de Calificación Profesional (TCP), que en la práctica se implementa en un Taller de Programación, con una carga horaria de 32 horas. En el mismo se desarrolla una aplicación que implementa los conceptos teórico-prácticos de la POO en Java

DESARROLLO DEL TCP

Las clases del TCP se llevarán a cabo los días sábado, en el horario de cursada correspondiente a cada comisión.

A lo largo de los encuentros se irá desarrollando un TP especial con entregas parciales.

Al estar afectada esta materia a la modalidad de cursada tipo “B” semi-presencial, el TCP se dictará en forma virtual con encuentros presenciales pautados entre estudiantes y docentes. Los mismos serán debidamente informados con anterioridad en la plataforma MIeL.

Para el desarrollo de las clases presenciales se utilizan los laboratorios de la Universidad.

RECURSOS DIDÁCTICOS

Apuntes desarrollados por la cátedra, presentaciones de diapositivas, guía de trabajos prácticos, problemas resueltos, literatura, bibliotecas y manuales de uso de las mismas.

Para la realización de los trabajos prácticos se ha elegido Eclipse implementando el lenguaje Java.

RÉGIMEN DE CURSADA

Se requiere una asistencia no inferior al 75%, en caso de no cumplirse, el alumno queda en condición de ausente.

Las evaluaciones parciales, serán presenciales, pudiéndose desarrollar en laboratorio, en forma escrita o una combinación de ambas.

Se tomarán 2 (dos) evaluaciones parciales, y en caso de no aprobar una de ellas se tomará un recuperatorio. La calificación del recuperatorio reemplazará la de la evaluación que se recupere.

Las fechas estimadas de los parciales aparecen en el cronograma adjunto. La fecha definitiva será consensuada en el aula junto a los docentes de cada comisión.

La aprobación del TCP es condición necesaria, pero no suficiente, para la aprobación de la materia.

La situación final de los estudiantes que tengan aprobado el TCP y aprobados ambos parciales con 7 (siete) o más será 'Aprobó' (por promoción). Los que en ambos parciales tengan 4 (cuatro) o más puntos y menos de 7 (siete) en uno o ambos, quedarán en situación de 'Cursó'. Los que tengan un aplazo en uno o ambos parciales quedarán en situación de 'Reprobó'. Finalmente, los que no se presentan a la 2da evaluación parcial (ni al recuperatorio), quedarán en situación de 'Ausente'.

MECANISMOS DE SEGUIMIENTO, SUPERVISIÓN Y EVALUACIÓN DE LAS ACTIVIDADES:

Se llevará a cabo un seguimiento y supervisión contínua de las actividades individuales y grupales desarrolladas por los estudiantes tanto en las clases presenciales como virtuales en las distintas plataformas utilizadas.

Los parciales corregidos serán entregados en mano a los estudiantes, teniendo la posibilidad de realizar preguntas sobre las correcciones efectuadas. Previamente a la entrega de los parciales corregidos, se explicará brevemente la solución a los problemas planteados haciendo hincapié en los puntos donde se observaron los errores más comunes.

CRONOGRAMA ORIENTATIVO DE ACTIVIDADES:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fecha TM** | **Fecha TN** | **Nro** | **Dia** | **Contenido** |
| 28-mar | 29-mar | Clase 1 | MA | Metodología de desarrollo de software. Instalación de herramientas Entorno de desarrollo Eclipse. Introducción al lenguaje de programación Java. Ejercicios simples de programación. |
| 30-mar | 31-mar | Clase 2 | JU | Pruebas del software. Lote de prueba. Programa probador. Construir la prueba para enunciados de ejercicios seleccionados de la OIA. |
| 4-abr | 5-abr | Clase 3 | MA | Lectura y escritura de archivos en Java. Ejercicios con lectura, procesamiento y salida de archivos |
| 6-abr | 7-abr | clase 4 | JU | Conceptos de POO. Clase Punto |
| 11-abr | 12-abr | clase 5 | MA | Continuación de POO-Herencia y Polimorfismo |
| 13-abr | 14-abr | clase 6 | JU | ejercicios |
| 18-abr | 19-abr | clase 7 | LU\_MA | Diseño Orientado a Objetos. Diagrama de clases UML. Diseño de la solución orientada a objetos del ejercicio Luchadores Japoneses de la OIA. |
| 20-abr | 21-abr | clase 8 | MI\_JU | Colecciones - Excepciones |
| 25-abr | 26-abr | clase 9 | LU\_MA | Métodos de ordenamiento elementales |
| 27-abr | 28-abr | clase 10 | MI\_JU | Métodos de ordenamiento no elementales |
| 2-may | 3-may | clase 11 | LU\_MA | Complejidad Computacional |
| 4-may | 5-may | clase 12 | MI\_JU | TDA Montículos - Cola de Prioridad |
| 9-may | 10-may | clase 13 | LU\_MA | Complejidad Computacional + Programación Dinámica |
| 11-may | 12-may | clase 14 | MI\_JU | Programación Dinámica |
| 16-may | **17-may** | clase 15 | LU\_MA | Patrones de Diseño |
| 18-may | 19-may | clase 16 | MI\_JU | Repaso |
| 23-may | 24-may | clase 17 | LU\_MA | 1 Parcial |
| 25-may | 26-may | clase 18 | MI\_JU | Devolución de la corrección |
| 30-may | 31-may | clase 19 | LU\_MA | Prim / Kruskal |
| 1-jun | 2-jun | clase 20 | MI\_JU | Dijkstra / Floyd / Warshall |
| 6-jun | 7-jun | clase 21 | LU\_MA | Coloreo de grafos. Sec, Welsh-Powell y Matula. |
| 8-jun | 9-jun | clase 22 | MI\_JU | Ejercicios de Grafos |
| 13-jun | 14-jun | clase 23 | LU\_MA | Resolución ejercicios OIA nivel 3 |
| 15-jun | 16-jun | clase 24 | MI\_JU | Prolog |
| 20-jun | 21-jun | clase 25 | LU\_MA | Prolog |
| 22-jun | 23-jun | clase 26 | MI\_JU | Prolog |
| 27-jun | 28-jun | clase 27 | LU\_MA | Haskell |
| 29-jun | **30-jun** | clase 28 | MI\_JU | Prolog, Haskell, Dudas |
| 4-jul | 5-jul | clase 29 | LU\_MA | 2 Parcial Presencial |
| 6-jul | **7-jul** | clase 30 | MI\_JU | Devolución de la corrección |
| 11-jul | 12-jul | clase 31 | LU\_MA | Recuperatorio Presencial |
| 13-jul | 14-jul | clase 32 | MI\_JU | Llenado de actas |