## UNIVERSIDAD TÉCNICA NACIONAL

**CARRERA: INGENIERÍA DEL SOFTWARE**

**CURSO: PROGRAMACIÓN I**

**CÓDIGO: ISW-211**

**NATURALEZA DEL CURSO: TEÓRICO-PRÁCTICO**

**NIVEL: II**

**CRÉDITOS: 6**

**HORAS PRESENCIALES / SEMANA: 8 (4 HORAS TEORÍA, 4 HORAS PRÁCTICA)**

**TIEMPO DE ESTUDIO INDEPENDIENTE POR SEMANA: 11, 18 HORAS**

**MODALIDAD: CUATRIMESTRAL**

**REQUISITOS: ISW-111 PRINCIPIOS DE PROGRAMACIÓN**

**CO-REQUISITOS: NINGUNO**

**I. DESCRIPCIÓN DEL CURSO**

Este curso está orientado a estudiantes del II nivel de Carrera. Este primer curso de programación, pretende que el estudiante desarrolle su habilidad para resolver problemas de forma metódica, aplicando los conceptos de la Programación Orientada a Objetos, a la vez que aprende a utilizar las estructuras dinámicas y estáticas de datos

Durante el curso, el estudiante utilizará las técnicas de desarrollo de programas, aprendiendo a identificar el problema, analizarlo, y proceder a la descomposición del mismo para crear un modelo y diseñar la especificación del algoritmo que luego codificará con el lenguaje apropiado en el momento en que se imparte, asegurándose del cumplimiento con estándares de calidad.

Este curso es del tipo teórico y práctico, en el que se combina la base teórica que fundamenta los principios de la programación orientada a objetos con el diseño de prácticas en el laboratorio, que hagan uso de dichos conceptos, paralelo a las tareas programadas que reforzarán las destrezas del estudiante en el área de la programación..

**II. OBJETIVO GENERAL**

* Crear soluciones para resolver problemas simples mediante el modelaje, diseño detallado y programación, aplicando técnicas actuales de desarrollo de software orientado a objetos y considerando criterios de calidad apropiados, mediante la utilización de la notación UML.

**III. OBJETIVOS ESPECÍFICOS**

1. Resolver problemas mediante el uso de modelos abstractos, para la formación de un esquema lógico conceptual.
2. Comprender los elementos y estructuras básicas presentes en un lenguaje de programación orientado a objetos, para la implementación de programas modulares, claros y simples.
3. Aplicar buenas prácticas de construcción de software, con el uso de estándares de documentación, codificación, verificación y validación para el aseguramiento de la calidad en el desarrollo del programa.
4. Analizar problemas mediante un proceso de descomposición y refinamiento en pasos sucesivos, para que brinde mayor comprensión al problema.
5. Sintetizar los resultados del proceso de análisis mediante el diseño de algoritmos, para la resolución de problemas.
6. Comprender la temática principal de los tipos de datos abstractos, mediante lecturas y prácticas apropiadas, para la obtención de mejores resultados en materia de manipulación de información.
7. Dominar el entorno de la programación con datos abstractos, mediante lecturas y prácticas apropiadas.

**IV. CONTENIDOS**

**UNIDAD I. Conceptos básicos en la programación orientada a objetos.**

* Introducción al ciclo de desarrollo de un programa: Identificación del problema, Análisis y descomposición del problema, Diseño, Implementación, Ejecución del programa, y Pruebas.
* Conceptos de: Ambiente de desarrollo, lenguaje, código fuente, código objeto, compiladores e intérpretes.
* Estructura básica de un programa orientado a objetos.
* Escritura, compilación y ejecución de un programa.

**UNIDAD II. Primitivas de codificación.**

* Tipos de datos, variables, constantes, valores (literales) y su uso para la representación de atributos de una clase.
* Declaración e inicialización de variables para el almacenamiento de valores.
* Uso de variables como atributos de un objeto
* Declaración e inicialización de referencias a instancias de objetos.
* Entrada y salida básica de datos.
* Operadores aritméticos y su prioridad de evaluación.
* Evaluación de expresiones para la solución de problemas.

**UNIDAD III. Estructuras de Control.**

* Operadores relacionales, operadores lógicos y prioridad entre los mismos.
* Tipos de estructuras de control del flujo de programas (secuenciales, condicionales y de iteración)
* Estructuras condicionales: condición lógica (if - else), selección de casos múltiples (switch - case).
* Estructuras de iteración: ciclos con contador, ciclos con condición de entrada y ciclos con condición de salida (for, while, do-while).
* Eje transversal**:** Importancia de los estándares de codificación.
* Eje transversal: Importancia del control de calidad, listas de revisión y casos de prueba.

**UNIDAD IV. Métodos.**

* Funciones matemáticas de biblioteca (potencias, raíz cuadrada, redondeo, truncamiento, generación de números aleatorios, entre otras).
* Métodos: Uso de los métodos para representar las acciones que lleva a cabo un objeto. Paso de parámetros por valor y por referencia.
* Acceso a los atributos de un objeto (asignar y retornar los valores de cada atributo de un objeto).
* Constructores y destructores.
* Principio de localidad de las variables.
* Acceso a los miembros de una clase
* Uso de métodos públicos para acceder atributos privados.
* Sobrecarga de métodos.

**UNIDAD V.Manipulación de Caracteres e Hileras de Texto.**

* Manipulación de caracteres.
* Hileras de caracteres (Ejemplo: String).
* Operaciones básicas con hileras de caracteres. Ejemplo: convertir a mayúscula, minúscula, eliminar blancos, buscar sub-hileras, comparar hileras.

**UNIDAD VI. Fundamentos de arreglos (unidimensionales y multidimensionales).**

* Arreglos unidimensionales y su uso para la representación y desarrollo de estructuras de datos.
* Arreglos multidimensionales (matrices) y su uso para la representación y desarrollo de estructuras de datos.
* Algoritmos clásicos de ordenamiento (burbuja y selección) y búsqueda en arreglos (búsqueda binaria y secuencial).

**UNIDAD VII.Fundamentos de estructuras de datos abstractos.**

* Clasificación y formas de almacenamiento de las estructuras de datos.
* Objetivos de cada estructura de datos.
* Pilas : Operaciones sobre pilas
* Implementación sobre arreglos
* Implementación sobre listas

**UNIDAD VIII.Método de Ordenamiento y Búsqueda**

* Ordenamiento y búsqueda
* Burbuja, Quick Sort, Heap Sort, Merge Sort, etc.

## UNIDAD IX. Fundamentos de Graficación

* Contextos gráficos. La pantalla gráfica vista como una matriz de píxeles.
* Manejo básico de tipos de letra y color.
* Formas simples: líneas, círculos, rectángulos y polígonos.

**V. METODOLOGÍA**

El curso se lleva a cabo en forma teórico-práctica con una amplia participación por parte del estudiante y trabajo extraclase.

**VI. EVALUACIÓN**

1º Parcial 20%

2º Parcial 20%

1º Proyectos 20%

2º Proyectos 20%

Trabajos, pruebas cortas y tareas 20%

**Total 100%**

**VII. BIBLIOGRAFÍA**

*Arnow, D. y Weiss, G. (2001)* ***Introducción a la programación con Java****. Addison Wesley.*

*Deitel y Deitel.(1998). Cómo programar en Java. Prentice Hall.*

*Humphrey, Watts S.(1997). Introduction to the Personal Software Process.Addison Wesley.*

*Larman, Craig. (1999). UML y Patrones: Introducción al análisis y diseño orientado a objetos. Prentice Hall.*

*Lemay, Laura; Cadenhead, Rogers. (1999) Aprendiendo Java 2 en 21 días. Prentice Hall.*

*Meyer, Bertrand.(1997). Object-Oriented Software Construction.Prentice Hall.*

*Page-Jones, Meilir.(2000). Fundamentals of Object-Oriented Design in UML. Addison Wesley.*

*Savitch, Walter J. (1999). Java: an introduction to computer science and programming. Prentice Hall.*