	UNIVERSIDAD NACIONAL DE ITAPUA – U.N.I. ***Creada por Ley Nº:1.009/96 del 03/12/96*** **Facultad de Ingeniería**	
	PROGRAMA DE ESTUDIOS	

Materia:	Introducción a la Programación II		Semestre:	Segundo
Ciclo:	Ingeniería Informática			
Código:	303			
Horas Semanales:	Teóricas:	4		
	Prácticas:	-		
	Laboratorio:	4		
Horas Semestrales:	Teóricas:	68		
	Prácticas:	-		
	Laboratorio:	68		
Pre-Requisitos:	Introducción a la Programación I			

I - OBJETIVOS GENERALES:

El Ingeniero en Informática debe entender profundamente las nociones de la programación de computadoras. Este curso es una continuación de Introducción a la programación I en la que el alumno aprende a modelar problemas del mundo real utilizando un lenguaje de programación.

Los objetivos de esta materia son desarrollar en el alumno las capacidades de:

1. Aplicar conocimientos matemáticos, científicos y de ingeniería.
2. Identificar, formular y resolver problemas de ingeniería.
3. Usar técnicas, capacidades, y herramientas modernas de ingeniería.

II - OBJETIVOS ESPECIFICOS:

Este semestre se enfoca en crear programas de una complejidad mayor a lo anteriormente aprendido (Introducción a la programación I). En particular se enfoca en diseñar programas que tienen varios módulos. Para el desarrollo de esta materia se recomienda un lenguaje orientado a objetos moderno como Java o Python, pero la elección de dicho lenguaje está sujeto a cualquier ajuste según el Dirección de la Carrera en concordancia con el resto del plan de estudios. Esto permite que con el tiempo se actualice el lenguaje según las necesidades académicas y consideraciones del mercado. El enfoque será de aprender a modelar e implementar programas utilizando un lenguaje orientado a objetos.

Al término del curso el estudiante será capaz de:

1. Modelar un sistema en términos de objetos
2. Utilizar las nociones de abstracción de datos, esconder información, y modularización para desarrollar programas altamente mantenibles
3. Entender los principios básicos de orientación a objetos: Encapsulación, Composición, Herencia, Polimorfismo
4. Resolver problemas algorítmicos de dificultad media

Algunas capacidades secundarias que podrá adquirir adicionalmente son:

5. Saber utilizar estructura de datos simples (Listas, Tablas Hash, etc..) y entender cómo funcionan
6. Saber comprobar y depurar sus programas
7. Utilizar librerías (APIs) de la plataforma
8. Escribir programas basados en eventos
9. Utilizar la documentación de referencia de un lenguaje de programación para lograr utilizar interfaces de programación de aplicaciones (APIs)
10. Utilizar depuradores para analizar la ejecución de su programa y poder solucionar problemas en el código utilizando estas herramientas

III. CONTENIDOS PROGRAMÁTICOS

La materia es fundamentalmente práctica y requerirá el uso de una computadora para realizar varios trabajos de programación relativamente largas.

Generalmente cada unidad tiene por lo menos un proyecto sustancial asociado.

Unidad I

Introducción: programas, programación, lenguajes

1. Necesidad de lenguajes de programación
2. Funciones de los lenguajes de programación: programar la computadora, ayudar al programador a pensar
3. Tipos de lenguajes de programación (panorama general)
4. El valor del código fuente y la importancia de programar de manera profesional
5. Características de programas bien escritos: conciso/claro/correcto/completo (4c's), mantenerlo simple y obvio (KISS), no ser repetitivo (DRY)
6. Tipos de programas: aplicaciones de escritorio, web, móvil, applets
7. Proceso de compilación de un programa
8. Repaso breve de nociones básicas de la programación (bucles, condicionales, variables, comentarios)

Unidad II

Creando Objetos y Definiendo clases simples

1. Orientación a objetos como simulación del mundo real
2. Nociones de modelación de un sistema utilizando objetos
3. Diferencia entre una clase y un objeto
4. Cómo crear objetos y llamar métodos
5. Objetos predefinidos por librerías de gran utilidad (ej: Scanner, String, StringBuffer, ArrayList, FileReader, etc.)
6. Definición de clases
7. Clases definen comportamiento del objeto: el comportamiento es más importante que la representación interna
8. Variables de instancia vs. variables locales vs. parámetros de métodos
9. Cómo funciona un método
10. Abstracción procedual y abstracción de datos
11. Escribiendo programas con clases y objetos simples
12. Tipos primitivos vs. objetos en Java
13. Estructuras de datos simples
14. Métodos estáticos para construcción controlada de objetos
15. Arreglos unidimensionales y bidimensionales

Unidad III

Encapsulación, Composición, Herencia y Polimorfismo

1. Escondiendo información con encapsulación
2. Controlando acceso a datos de un objeto con private, package protected, public
3. Organizando clases con paquetes
4. Modelando con encapsulación y composición
5. Jerarquías y modelado con jerarquías de clases
6. Diagramas de clases con UML
7. Ejemplos de herencia en la práctica
8. Aprovechando herencia para lograr polimorfismo

9. Soluciones polimorficas vs. soluciones con if-else
10. Ejemplos aplicados

Unidad IV

Otras abstracciones, Colecciones, Lambda

1. Otras abstracciones importantes para lograr polimorfismo (ej: interface en Java o equivalente en otro lenguaje, o mixins en lenguajes que lo soportan)
2. Nociones básicas de programación funcional (en Java): lambda, referencias a funciones, funciones / métodos de alto orden
3. Colecciones básicas: Listas dinámicas (ej: ArrayList), tablas hash (ej:HashMap, HashSet), colecciones ordenadas (ej: TreeMap) soportadas por el lenguaje

Unidad V

Programación de Interfaces de usuario

1. Interfaces de usuario como ejemplo de programación OO
2. Arquitectura básica de una interfaz de usuario
3. Consideraciones sobre el diseño de interfaces de usuario: facilidad de uso, uso de colores, seguimiento de estándares
4. Introducción a creación de ventanas, botones, listas, combobox
5. Uso de gestores de distribución (layout managers) para la distribución correcta de controles en la pantalla
6. Modelo-Vista-Controlador
7. Responder a eventos en la interfaz de usuario
8. Programación dirigido a eventos, conceptos básicos de programación reactiva
9. Cómo hacer una arquitectura con capas
10. Cómo analizar un problema
11. Aplicando OO a la solución del problema
12. Implementación del problema
13. Ejemplos con interfaz gráfica (ej: Swing u equivalentes en otros lenguajes/plataformas)

Unidad VI

Manejo de Errores y recursos

1. Manejo de errores con excepciones revisadas y no revisadas
2. Lanzar excepciones
3. Atrapar excepciones
4. Cuándo atrapar excepciones
5. El uso de finally para liberar recursos.
6. Distintos sintaxis para liberar recursos (ej: try {} finally vs try () en Java o equivalente en lenguaje usado)

Unidad VII

Flujos y Archivos

7. El uso de la abstracción flujo (Stream) en programas para lidiar con archivos, redes, etc. (aplicable a Java, si el lenguaje usado es más simple se puede optar por omitir esto)
8. Aplicación y composición de flujos para ejemplos prácticos: lectura escritura de archivos, compresión, comunicación por redes (aplicable a Java, si el lenguaje usado es más simple se puede optar por omitir esto)
9. Correcta limpieza de recursos con archivos
10. Uso de buffers y su importancia

Contenidos según guía de la ACM 2013: SDF/Fundamental Programming Concepts (Core-Tier1), SDF/Fundamental Data Structures (Core-Tier1), SDF/Development Methods (Core-Tier1), PL/Object-Oriented Programming (Core-Tier1, Core-Tier2
PL/Event-Driven and Reactive Programming (Core-Tier2)

IV - METODOLOGIA:

La Metodología principal para introducir la programación a los alumnos es “Objetos Primero” (Objects First). Esto solidifica los principios de Orientación a Objetos al principio de la carrera del alumno haciendo que otros paradigmas como el Procedual/Imperativo y Funcional sean más simples.

Esta materia debe requerir que los alumnos desarrollen varios trabajos prácticos para lograr sus objetivos. Los alumnos necesitarán practicar mucho para adquirir una habilidad significativa de

programación y modelación de soluciones. Por tanto los alumnos deberán realizar escribir varios programas empezando por algo simple e ir aumentando en complejidad durante el semestre.

Los alumnos deberán recurrir a manuales, y referencias técnicas (como Javadoc) para lograr cosas que no pudieron ser expandidas en clase. Esto enseñará al alumno a investigar y poder renovar su conocimiento constantemente.

También es crítico emplear metodologías que estimulan al alumno a resolver problemas de cierta dificultad y no acostumbrarse a solamente realizar problemas fáciles. Esta habilidad le permitirá tener éxito luego en su carrera universitaria.

La materia también tiene un componente de desarrollo con interfaces de usuario que es no solamente útil como ejemplo de programación OO sino como base para entender arquitecturas básicas de programación y poder juntar sus conocimientos de la materia en un programa que incluye una interfaz de usuario, lógica de la aplicación (en la capa de negocios), y escritura a un sistema simple de persistencia (utilizando archivos).

En esta materia los alumnos realizan varios trabajos extensos de programación fuera de clase, también resuelven ejercicios en casa. Las clases teoricas son expositivas mostrando cómo funcionan distintos aspectos del lenguaje de programación, demostrando técnicas, algoritmos y a veces resolviendo ejercicios en clase. Se utilizan materiales audio visuales y demostraciones para aclarar conceptos. En la clase de laboratorio el énfasis es resolver ejercicios de programación y aclarar temas dados en la clase teórica.

Para estimular el estudio y la lectura también se suelen usar pequeños examenes sobre la lectura para asegurar que los alumnos adquieran el hábito de lectura.

V- CRITERIOS DE EVALUACION

El componente central de evaluación son los proyectos de programación hechos por los alumnos. Para obtener derecho a examen los alumnos deben probar su aptitud de programación a través de los proyectos. Otros componentes que pueden ser deberes y examenes. En conjunto forman las calificaciones “parciales” del reglamento necesarias para acceder al examen final.

La evaluación precisa depende del reglamento de cátedra y el reglamento vigente de la Facultad.

VI. BIBLIOGRAFÍA

Dean, J.S. Dean, R.H. (2009). Introducción a la Programación con Java. McGraw-Hill. México.

Horstmann, C., Cornell G. (2006). Core Java 2. v5. Volumen I - Fundamentos. Pearson Prentice Hall. Madrid, España.

Arnow, D., Weiss, G. (2001). Introducción a la Programación con Java: Un enfoque Orientado a Objetos. Prearson. Madrid, España.

Schildt, H. (2005). La Biblia de Java 2 v5.0. McGraw-Hill. Madrid, España.

Cohoon, J., Davidson, J. (2006). Programación en Java 5.0. McGraw-Hill. Madrid, España.