Programación I

1. Características generales

Nombre:	Programación I	**Grupo**: 01, I-2017
Sigla:	CI-0112	**Horario**: L9-12, J9-11 205-IF
Créditos:	4	**Profesor**: Jeisson Hidalgo Céspedes
Horas:	5 horas de teoría	**Correo**: jeissonh@gmail.com
Requisitos:	Introducción a la Computación	**Oficina**: 201-IF
Correquisitos:	-	**Consulta**: J11-13 204-IF
Clasificación:	Curso propio	**Asistente**: Jose David Vargas Artavia
Ciclo:	II ciclo, 1er año	**Correo**: josed1608@gmail.com

2. Descripción del curso

Este curso fomenta en el estudiante habilidades generales para la resolución de problemas de programación, con énfasis en la etapa de implementación de soluciones. Se usa el paradigma de programación orientado a objetos porque permite solucionar problemas de forma más natural.

3. Objetivos

Objetivo general

El objetivo general del curso es que el estudiante sea capaz de resolver problemas mediante la programación de computadoras utilizando el paradigma de programación orientado a objetos.

Objetivos específicos

Durante este curso cada estudiante desarrollará habilidades para:

1. Explicar el modelo de ejecución de los programas.
2. Analizar problemas y diseñar soluciones básicas orientadas a objetos.
3. Implementar soluciones utilizando el paradigma de programación orientado a objetos.
4. Validar informalmente la ejecución de los programas mediante la definición de un conjunto básico de casos de prueba.
5. Aplicar buenas prácticas de programación.
6. Implementar y manipular estructuras de datos.
7. Implementar algoritmos básicos.
8. Utilizar un ambiente de desarrollo integrado.
9. Corregir errores en programas utilizando un depurador.

4. Contenidos

Objetivo	Eje temático	Desglose
8, 9	Herramientas de desarrollo	Herramientas de desarrollo: compilador, depurador, ambiente de desarrollo integrado (IDE).
1	Modelo de ejecución de programas	Rastreo de memoria de programa.

2	Resolución de problemas	Proceso de resolución de problemas, análisis, diseño, implementación, prueba, patrones, técnicas de resolución de problemas: divide y vencerás.
2, 3	Diseño e implementación de soluciones	Conceptualización y definición de clases, atributos de clases, instancias de clases, tipos y variables, parte pública, parte privada.
2, 3	Diseño e implementación de soluciones	Conceptualización e implementación de métodos (sin parámetros, con parámetros de valor, con parámetros de referencia), constructores y destructores.
6	Estructuras de datos	Estructuras de datos estáticas: arreglos de una y más dimensiones.
6	Estructuras de datos	Estructuras de datos dinámicas: lista enlazada y árbol binario.
3, 5, 7	Entrada y salida de datos	Flujos de entrada y salida de datos. Validación de datos.
7	Algoritmos	Estructuras básicas de control: secuenciación, bifurcación, iteración.
7	Algoritmos	Algoritmos básicos de búsqueda como: secuencial y binaria, y algoritmos de ordenamiento como: burbuja, selección e inserción.
7	Algoritmos	Funciones recursivas tales como: factorial, Fibbonacci, multiplicación de enteros, potencia de dos números, máximo común divisor de dos números, Torres de Hanoi; además, recorrido, inserción y borrado sobre árboles binarios.
4, 5, 8, 9	Calidad de programas	Casos de prueba básicos: casos extremos, intermedios, inválidos. Buenas prácticas de programación: nombres significativos, convenciones de escritura de nombres (clases, variables, métodos, constantes), disposición del texto (tabulación, delimitadores de bloques, espacios y líneas en blanco), documentación interna, orden de las declaraciones (campos, constructores, métodos), modificadores de acceso (privado y público), uso de constructores.

5. Metodología y evaluación

Para alcanzar los objetivos del curso se seguirá una metodología híbrida constructivista-tradicional. El estudiante deberá dedicar 5 horas semanales a lecciones presenciales. De esas horas, una parte será empleada por el profesor para explicar los conceptos mediante clases magistrales. La otra parte, a convenir dinámicamente durante el avance del semestre, será dedicada para que los estudiantes resuelvan problemas en un laboratorio de computadoras con ayuda del profesor. En las 7 horas extraclase, los estudiantes deberán resolver problemas de programación y otras tareas asignadas durante el curso. Se dará crédito en la nota final a la evidencia generada por el estudiante de acuerdo al siguiente desglose, el cual se explica en las siguientes secciones.

Ejercicios	25%	23%	Ejercicios resueltos
		2%	Ejercicios inventados
Proyectos	15%	7%	Proyecto 1
		8%	Proyecto 2
Exámenes	60%	20%	Examen 1
		20%	Examen 2
		20%	Examen 3

Ejercicios de programación

Los estudiantes resolverán ejercicios de programación planteados por el profesor y estudiantes previos del curso (ejercicios resueltos) y además propondrán ejercicios propios (ejercicios inventados). Los ejercicios tendrán el formato usado en los concursos de programación de la ACM. Se utilizará como juez automático la plataforma HackerRank [https://www.hackerrank.com/]. Los estudiantes proveerán su correo electrónico al profesor y recibirán invitaciones para acceder a los ejercicios.

Los ejercicios están organizados en secciones temáticas. Aproximadamente cada semana del curso se habilitarán los ejercicios de una sección y se enviarán invitaciones a los estudiantes. Durante las lecciones de la semana correspondiente, el profesor explicará, al menos parcialmente, los conceptos de programación requeridos por los ejercicios. Algunos ejercicios requerirán más detalles o conceptos de los vistos en clase, con el fin de que el estudiante investigue otras fuentes de información para poder resolver el problema y desarrollar habilidades autodidácticas requeridas en el ejercicio de la profesión.

Los estudiantes resolverán los ejercicios en horas extraclase y durante las lecciones presenciales en un laboratorio de la Escuela. Durante las lecciones de laboratorio, y las horas de consulta, los estudiantes podrán recibir apoyo del profesor para avanzar en los ejercicios de programación. Cada vez que el estudiante resuelve un ejercicio de programación obtendrá puntos que irá acumulando en la plataforma HackerRank. El estudiante recibirá crédito por sus puntos de HackerRank hasta un máximo de 23% de la nota del curso.

Los estudiantes también pueden aportar ejercicios de programación y recibirán crédito por ello. Un **ejercicio inventado** se debe ubicar en una de las secciones temáticas. Cada sección tiene su propio peso en la nota. Un ejercicio inventado tendrá crédito por el peso de la sección donde se ubique. Los estudiantes crearán ejercicios por al menos un 2% de la nota. Más de este porcentaje, serán puntos extra que el estudiante podrá acumular en la nota del curso. Cada ejercicio inventado debe tener el mismo formato usado en los ejercicios resueltos. El profesor proveerá recursos para facilitar la elaboración de los mismos.

La creación de ejercicios correlaciona con una mayor comprensión de los conceptos de programación, de acuerdo a la literatura científica. Para crearlos, el estudiante debe prestar atención a los conceptos de programación involucrados en el tema, y puede tomar como ejemplo los ejercicios propuestos por el profesor y asistentes en la plataforma HackerRank. Los ejercicios inventados deben ser únicos entre estudiantes. Es decir, si dos o más estudiantes proponen el mismo ejercicio, se dará crédito sólo al primero en someterlo. El profesor servirá como el ente centralizador de los ejercicios inventados por estudiantes, y podrá ayudarles en el proceso de creación de los mismos durante las horas de consulta.

Proyectos

Los estudiantes resolverán a lo largo del curso dos problemas de programación de mayor complejidad que los ejercicios de programación, a los cuales se les llamará proyectos. Ambos proyectos deben resolverse en equipos de dos estudiantes. Los integrantes de los equipos pueden variar entre el Proyecto 1 y el Proyecto 2. En ambos proyectos realizarán las fases de un proceso de desarrollo básico: análisis, diseño, implementación y pruebas.

Para el primer proyecto el profesor será el cliente. Para el segundo proyecto los estudiantes definirán el cliente, que puede ser ellos mismos u otra persona. Los estudiantes deben indagar y comprender el problema del cliente (análisis), derivar requerimientos, acordar con el cliente (y profesor) requerimientos para cada entregable, implementar los requerimientos y probarlos. Las revisiones de los proyectos serán a través de entregables, aproximadamente tres o cuatro por proyecto, distanciados por dos semanas.

Los entregables se realizarán en un repositorio de control de versiones a convenir entre los estudiantes y el profesor. Algunas revisiones podrán realizarse mediante reuniones, en las cuales, el profesor pedirá a los dos integrantes explicar al menos una funcionalidad (implementación de un requerimiento). Una tercera funcionalidad será revisada por el profesor directamente a partir del código fuente. La calificación de esta tercera parte dependerá entonces de la comprensión que el profesor pueda tener a partir del código fuente, y por tanto, de la calidad de la documentación y buenas prácticas de programación empleadas por los miembros del equipo.

Exámenes

Se realizarán tres **exámenes** parciales cuyas fechas serán acordadas entre los estudiantes y el profesor tras cubrir los temas correspondientes. El material cubierto en cada examen es acumulativo de los anteriores, por la naturaleza de los contenidos. Los exámenes serán realizados en computadora o en papel, un día sábado, y durante un período de tres horas.

En las horas de consulta el profesor podrá apoyar varias de las actividades descritas anteriormente:

1. Ayudar a los estudiantes en la resolución de los ejercicios de programación planteados en HackerRank.
2. Ayudar a los estudiantes a plantear sus propios ejercicios de programación.
3. Ayudar a los estudiantes en la solución de sus proyectos de programación.
4. Evaluar el avance de los proyectos de programación.
5. Aclarar dudas o consultas.

Lineamientos

1. En cualquiera de los tres tipos de evaluaciones (ejercicios, proyectos y exámenes), el profesor podría proponer actividades opcionales por crédito extra en la nota del curso.
2. Se utilizarán los equipos y herramientas computacionales que provee la ECCI para realizar las actividades del curso. Es obligación del estudiante tener acceso a esta plataforma. Alternativamente el estudiante podrá utilizar su propio equipo bajo su responsabilidad.
3. Durante los exámenes el estudiante podrá consultar material teórico (libros, apuntes) o código que sea producto de su trabajo en el curso o que haya sido proporcionado por el profesor. Los medios para consultar estos materiales durante el examen serán indicados previamente por el profesor.
4. Es ilegal presentar como propio, código parcial o total escrito por otras personas u obtenido de fuentes de información, como por ejemplo de libros o de Internet, sin la autorización expresa del docente. Los exámenes son estrictamente individuales, y en caso de ser por computadora, es ilegal violar esta regla a través de aplicaciones de correo electrónico o mensajería. En cualquier asignación en que se detecte plagio, se asignará un cero como nota en la evaluación y se aplicará lo estipulado en el Reglamento de Orden y Disciplina de los Estudiantes de la Universidad de Costa Rica (http://www.cu.ucr.ac.cr/uploads/tx_ucruniversitycouncildatabases/normative/orden_y_disciplina.pdf).
5. Todo código entregado por el estudiante debe compilar sin errores. De lo contrario será calificado con 0.

6. Bibliografía

- Barnes, David y Kolling, Michael. *Programación orientada a objetos con Java usando BlueJ.* Quinta edición. Pearson. 2013.
- Deitel y Deitel. *Java How To Program: Early Objects*. Décima edición. Pearson. 2014.
- Wu, C. Thomas. *An Introduction to Object-Oriented Programming with Java*. McGraw-Hill. Quinta edición. 2009.
- Ceballos, Francisco Javier. *Java2: Curso de Programación*. Cuarta edición. Editorial Alfaomega. 2011.