PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 — Programación Avanzada

Programa de curso

Actualización: 9 de noviembre de 2022

Profesores:	Hernán Valdivieso, Nicolás Elliott, Lesly Reyes, Joaquín Tagle
Clases:	Jueves, módulos 4 y 5
Ayudantías:	Martes, módulo 4
Formato:	Presencial
Requisitos:	IIC1103 — Introducción a la Programación
Sitio web:	iic2233.ing.puc.cl

Introducción

El desarrollo de la computación ha hecho que estemos rodeados de dispositivos que ejecutan programas computacionales todo el tiempo. Un aspecto importante para ser partícipes del desarrollo de estas herramientas es comprender su funcionamiento y cómo construir desde programas simples a software más complejo. Este curso profundiza aspectos de la programación que se introdujeron en cursos anteriores, en particular en las estructuras y técnicas de diseño que permiten construir herramientas más complejas y prácticas.

Objetivo General y Competencias

A lo largo de este curso, el alumno desarrollará técnicas para diseñar, implementar, ejecutar y evaluar herramientas de software que resuelvan problemas algorítmicos a partir de especificaciones detalladas. El alumno será capaz de desarrollar construcciones avanzadas de programación orientada a objetos y estructuras de datos fundamentales, construir código robusto, construir interfaces gráficas, y utilizar conceptos como *threading*, serialización y paso de mensajes.

Al finalizar el curso, el estudiante será capaz de:

1. **Comprender técnicas básicas de mantención de código** incluyendo uso de guía de estilo, modularización y sistemas de manejo de versiones.
2. **Inferir un modelo de objetos** para resolver problemas realistas e **implementar esta solución** usando técnicas de programación orientada a objetos.
3. **Usar estructuras de datos básicas** para resolver problemas de programación.
4. **Utilizar objetos iterables** para resolver problemas de programación.
5. **Aplicar el concepto de *threading*** para la modelación de problemas de colas.
6. **Construir interfaces gráficas** funcionales utilizando bibliotecas apropiadas.
7. **Aplicar las formas de manejo de excepciones** en un programa, para construir código robusto.
8. **Implementar estructuras de datos basadas en nodos**, como listas ligadas, árboles, y grafos.
9. **Utilizar el concepto de serialización** para construir codificadores y decodificadores.
10. **Utilizar el concepto de paso de mensajes** para construir una aplicación distribuida básica.

Contenidos

Fundamentos de programación

- **Estructuras de datos básicas:** tuplas, *named tuples*, *stacks*, colas, diccionarios, *sets*.
- **Programación orientada a objetos:** objetos, herencia, herencia múltiple, polimorfismo, clases abstractas.
- **Iterables:** objetos iterables; generadores; y funciones de mapeo, filtro y reducción.
- **Manejo de excepciones**: tipos de excepciones, control de excepciones.
- **Estructuras de datos basadas en nodos:** listas ligadas, árboles, grafos.

Herramientas de programación

- **Técnicas básicas de mantención de código:** concepto y uso herramientas de sistemas de manejo de versiones, uso de guías de estilo y modularización.
- ***Threading:*** concepto de *pseudo-paralelismo*, creación y sincronización de *threads*, concurrencia.
- **Interfaces gráficas:** introducción a las interfaces gráficas usando la librería PyQt5.
- **I/O:** manejo de *bytes*, serialización binaria, serialización en formato JSON.
- ***Networking:*** *sockets*, modelo cliente-servidor y paso de mensajes.

Metodología

El curso seguirá una metodología *blended* en la cual se utilizará el modelo de clase invertida (*flipped classroom*) con actividades de programación en clase, tareas individuales de programación, sesiones de apoyo en ayudantía y sesiones expositivas en clases.

La modalidad de la clase de cada semana será *flipped classroom*. En estas, los alumnos deben estudiar los contenidos de manera previa a la clase, para luego aplicarlos en ella mediante actividades prácticas de programación. Para este fin, el material de estudio se encontrará disponible en el sitio del curso (al menos) desde el viernes anterior a la clase, y consiste en apuntes donde se describen detalladamente los tópicos.

En el horario de cátedra, el profesor hará un breve repaso, de 20 minutos, sobre el material de estudio a modo de introducción para la clase y para resolver dudas con respecto a los contenidos estudiados. Luego, los alumnos resolverán una actividad donde se apliquen los nuevos conocimientos. Tanto el profesor como ayudantes del curso estarán presentes durante toda la duración de esta actividad para resolver dudas y guiar la actividad. Tras recolectar las actividades realizadas, los últimos 10 minutos de la clase el profesor realizará un cierre, donde se discutirán posibles soluciones y se resumirán los aprendizajes de la semana.

El curso contará con ayudantías semanales con el fin de reforzar los contenidos antes de la sesión de discusión en clases o, durante las semanas de tareas, apoyar a los alumnos con la elaboración de sus tareas mediante sesiones de ayuda. Es importante notar, que las sesiones de reforzamiento de contenidos también asumen revisión previa del material y no están orientadas a ser un reemplazo de estudio previo, sino que son una instancia de resolución de dudas, previa a la clase.

El detalle de la modalidad de cada clase estará indicado desde el inicio del semestre en el calendario del curso.

Evaluación

La evaluación será efectuada mediante actividades de programación durante la clase y tareas individuales de programación de mayor extensión.

Evaluaciones en clases. Durante las clases de modalidad *flipped classroom* se realizará una evaluación de carácter formativo o sumativo, que consistirá en una actividad de programación. Se realizarán cuatro (4) actividades de tipo formativa, cuatro (4) de tipo sumativa, y una (1) actividad bonus en las fechas detalladas en el calendario del curso. Dado el carácter acumulativo del curso, cada evaluación incluye el contenido de las semanas anteriores a menos que se explicite lo contrario, pero el foco será el contenido de la semana.

Actividades sumativas. Su objetivo es evaluar el aprendizaje acumulado individual de cada estudiante mediante una actividad práctica durante la clase. Cada trabajo será evaluado y se asignará una nota de evaluación y retroalimentación personalizada a cada alumno. Se realizarán cuatro (4) durante el semestre, generando las siguientes notas: $\mathbf{AS_1}, \mathbf{AS_2}, \mathbf{AS_3}$ y $\mathbf{AS_4}$.

Actividades formativas. Su objetivo es conseguir que el estudiante practique el contenido de la semana, resuelva sus dudas de forma guiada por el profesor y ayudantes, y reflexione sobre su aprendizaje.

Actividades bonus. Su objetivo es evaluar el aprendizaje individual de cada estudiante en el último contenido esencial del curso. Esta actividad se rendirá al final del semestre y otorgará décimas adicionales al promedio de tareas o actividades. Según lo que más beneficie a cada alumno. La corrección de esta evaluación será de forma automática y se proveerá de un reporte a cada estudiante con la respuesta de cada caso testeado automáticamente.

Luego, la nota **AC** de actividades corresponderá al promedio aritmético de las cuatro notas considerando las notas $\mathbf{AS_1}, \mathbf{AS_2}, \mathbf{AS_3}, \mathbf{AS_4}$) del estudiante. Es decir, la nota **AC** se calcula de la siguiente manera:

$$\mathbf{AC} = \frac{\mathbf{AS_1} + \mathbf{AS_2} + \mathbf{AS_3} + \mathbf{AS_4}}{4}$$

El ausentarse a una actividad sumativa implica directamente nota mínima en la nota correspondiente, a menos de que se justifique correctamente ante la Dirección de Pregrado. En el último caso, debe notificarse dicha justificación con la mayor anticipación posible, para resolver el caso particularmente.

En caso de ausentarse a una actividad **sumativa** por motivo de contagio Covid, nos guiaremos por el protocolo dictado por la universidad. Deberá presentar un certificado emitido por enfermera del campus San Joaquín. Para obtener este certificado, deben reportar su situación en este **formulario**.

Tareas. Se publicarán cuatro tareas de programación las que deberán ser resueltas **individualmente** por cada alumno. Debido a los niveles de dificultad y extensión de cada tarea, la primera tendrá un plazo de realización de una semana mientras que las siguientes tendrán un plazo de al menos dos semanas. La nota de cada tarea se especifica como $\mathbf{T_0}$, $\mathbf{T_1}$, $\mathbf{T_2}$ y $\mathbf{T_3}$, y la nota ponderada total de tareas se calcula como:

$$\mathbf{T} = \frac{\mathbf{T_0} + 2 \times \mathbf{T_1} + 3 \times \mathbf{T_2} + 3 \times \mathbf{T_3}}{9}$$

Proceso de recorrección. Luego de publicadas las notas de una evaluación, se dará un periodo de una semana para recibir solicitudes de recorrección. Cada solicitud debe estar debidamente justificada, y debe ser enviada por los canales que el curso disponga para este propósito.

En caso de que la respuesta a la solicitud de recorrección no sea satisfactoria, se deberá llenar un formulario — dentro de una semana de publicada la recorrección— para solicitar que los profesores revisen el caso. La decisión que se tome en esta instancia es inapelable.

Tanto actividades como tareas de programación se entregan **únicamente** a través de un repositorio privado y personal del alumno. Este se alojará en la plataforma **GitHub** y será provisto por el equipo docente a cada alumno. **Este es el medio de entrega oficial y único del curso. Cada evaluación especificará la carpeta en que debe entregarse, dentro de su repositorio.**

Nota final y aprobación. La nota del curso se calcula como $\mathbf{NC} = (2 \times \mathbf{T} + \mathbf{AC})/3$. La nota final del curso **NF** se calculará como:

$$\mathbf{NF} = \begin{cases} \mathbf{NC} & \text{si } \mathbf{T} \geq 3{,}95 \text{ y } \mathbf{AC} \geq 3{,}95 \\ \text{mín}(\mathbf{NC};\, 3{,}9) & \text{en otro caso.} \end{cases}$$

El alumno aprobará el curso si su nota final del curso **NF** es mayor a 3,95.

Todas las notas serán calculadas con **dos decimales**, salvo la nota final del curso que se calculará con **un decimal**.

Cualquier situación de **copia** detectada en alguna evaluación tendrá como **sanción un 1**,**1 final en el curso**. Esto sin perjuicio de sanciones posteriores que estén de acuerdo a la Política de Integridad Académica de la Escuela de Ingeniería y de la Universidad, que sean aplicables al caso. Rige para este curso tanto la política de integridad académica del Departamento de Ciencia de la Computación (ver anexo) como el **Código de honor de la Escuela de Ingeniería**.

Debido a la naturaleza de la disciplina en la que se enmarca el curso, está permitido el uso de código escrito por una tercera parte, pero solo bajo ciertas condiciones. Primero que todo, el uso de código ajeno **siempre debe estar correctamente referenciado**, indicando la fuente de donde se obtuvo. Y por otro lado, se permite el uso de código encontrado en internet u otra fuente de información similar, siempre y cuando su autor sea **externo al curso**, o en su defecto, sea parte del **equipo docente** del curso. Es decir, se puede hacer referencia a código ajeno al curso y código perteneciente al curso pero solo aquel escrito por el equipo docente, como material o ayudantías. Luego, compartir o usar código de una evaluación **actual o pasada** se considera una falta a la ética.

Evaluación de última instancia. Se dará la oportunidad de rendir una evaluación oral el **lunes 12 de diciembre** a aquellos alumnos que se encuentren en el borde de reprobar el curso. Un alumno podrá optar a realizar esta evaluación si su nota del curso **NC** es mayor o igual a 3.95, pero alguna de sus notas parciales (**T** o **AC**) se encuentra entre 3.50 y 3.94 (inclusivo). El aprobar esta instancia permitirá calcular la nota final del curso como $\mathbf{NF} = \mathbf{NC}$.

Bibliografía

- K. Pichara, C. Pieringer. *Advanced Programming in Python*, disponible en **Amazon**.

Anexo: Política de integridad académica del Departamento de Ciencia de la Computación

Se espera los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile mantengan altos estándares de honestidad académica, acorde al Código de Honor de la Universidad. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Pregrado de la Escuela de Ingeniería (Disponible en SIDING, en la sección Pregrado/Asuntos Estudiantiles/Reglamentos/Reglamentos en Ingeniería/Integridad Académica).

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente *política de integridad académica*. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho **individualmente** por el alumno, **sin apoyo en material de terceros**. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros.

En particular, si un alumno copia un trabajo, o si a un alumno se le prueba que compró o intentó comprar un trabajo, **obtendrá nota final 1.1 en el curso** y se solicitará a la Dirección de Pregrado de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral.

Por "copia" se entiende incluir en el trabajo presentado como propio, partes hechas por otra persona. En caso que corresponda a "copia" a otros alumnos, la sanción anterior se aplicará a todos los involucrados. En todos los casos, se informará a la Dirección de Pregrado de la Escuela de Ingeniería para que tome sanciones adicionales si lo estima conveniente.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, **siempre y cuando se incluya la referencia correspondiente**.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile (**http://admisionyregistros.uc.cl/alumnos/informacion-academica/reglamentos-estudiantiles**). Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.