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

IIC2233 Programación Avanzada (II/2015)

Tarea 1

1. Objetivos

- Aplicar los conceptos de la programación orientada a objetos para modelar un sistema.
- Desarrollar algoritmos para la resolución de problemas complejos.
- Trabajar con archivos de texto para obtener/procesar datos.

2. Introducción

Los estudiantes de una Universidad ~~no~~ muy lejana han iniciado un movimiento revolucionario que busca eliminar permanentemente el sistema vigente de toma de ramos debido a problemas en su implementación. El caos gobierna la Universidad y su directiva tiene como prioridad hacer un pacto de paz y acabar con la guerra. Entre las medidas que han sido tomadas para apaciguar la ira del pueblo se destacan: extender las vacaciones, pedir disculpas por los incovenientes, decorar los errores del portal y bajar el monto de las multas en la biblioteca. En un acto de desesperación, la institución ha amenazado a los alumnos con su castigo más temido: los sumarios. Es por esto que el grupo revolucionario no puede aguantar más y tiene como objetivo principal imponer su nuevo sistema de ramos "Bummer UC", el que terminará por abolir las malas prácticas de los líderes actuales.

3. Problema

En esta tarea deberá modelar e implementar diversos sistemas de tomas de ramos, utilizando programación orientada a objetos, y creando múltiples algoritmos para su correcto funcionamiento.

4. Bummer UC

Esta parte de la tarea consiste en modelar el sistema de Bummer UC, además de crear un *main.py* que permita la interacción entre el usuario y el sistema, con el fin de probar las funcionalidades que implemente.

4.1. Estructura

El sistema está estructurado de la siguiente forma:

- Curso: Los cursos están compuestos de sigla, profesor, lista de alumnos, horario, sección, campus, evaluaciones, requisitos y capacidad máxima.

- Alumno: Cada alumno tiene un nombre, apellido, contraseña, horario de inscripción de cursos, cursos aprobados y cursos por tomar.
- Horario: El horario de un curso está subdividido por el tipo de actividad (cátedra, ayudantía o laboratorio). Estos informan el día de la semana, la hora y la sala en que se realizan cada una de las actividades.
- Evaluación: Posee nombre, información acerca de la hora y el día en el que se realizar y el curso al cual se encuentra relacionada

4.2. Funcionalidades

El sistema debe tener las siguientes funcionalidades[1]:

- Sistema de Log-In: Un alumno o profesor podrá loguearse en el programa utilizando su usuario y contraseña. De no estar logueado solo se podrán consultar datos sobre los cursos.

Al conectarse como alumno, obtendrá las siguientes funcionalidades .

- Inscribir ramo: Al inscribir un ramo ocurre lo siguiente:
  - Disminuye una vacante del curso.
  - Se agrega el alumno a la lista de alumnos del curso.
  - Se agrega el curso como curso por tomar.
  - Se toman en cuenta las siguientes restricciones:
    - De vacantes.
    - De requisitos.
    - De topes de horario.
    - De créditos mínimos y máximo permitidos: La cantidad mínima de créditos que puede tomar un alumno es de 30. Por otro lado, la cantidad de máxima de créditos permitidos está dada por la siguiente fórmula: $55 + (6 - \text{grupo bummer }) \cdot 2$.
    - De evaluaciones: si alguna de las evaluaciones topa en fecha y hora, no se debe poder inscribir el curso.
    - De campus: No puede ocurrir que dos cursos de un alumno se dicten de forma consecutiva (horario) y en campus diferentes.
- Botar ramo: Al botar un ramo ocurre lo siguiente:
  - Aumenta una vacante del curso.
  - Se elimina el alumno de la lista de alumnos del curso.
  - Se elimina el curso como curso por tomar.
- Generar Horario: El programa debe permitir que un alumno imprima su carga horaria semanal. Se debe generar el horario de dos formas:
  1. Imprimir horario en consola (en una tabla de forma amigable para el usuario; indicando días, horarios, siglas y secciones correspondientes).
  2. Crear un archivo de texto (en cualquier path, pero que sea informado al usuario) que guarde el horario, tal como se ve al imprimirlo en cosola (tipo Portal UC).
- Generar calendario de evaluaciones: El programa debe permitir crear un archivo de texto con el calendario de evaluaciones del alumno (formato de libre elección).

NOTA: Solo se podrán inscribir y botar cursos durante el horario asignado al grupo del alumno.

[1]Si desea agregar más detalle en las funcionalidades (para hacerlo más realista aún) puede hacerlo siempre y cuando lo especifique en el README.

Al conectarse como profesor obtendra las siguientes funcionalidades:

- Darle permiso a un alumno para tomar un ramo que usted dicte, pese a no cumplir con los requisitos. Esto implica que el alumno, dentro de su horario predefinido, podra inscribir el ramo exitosamente ignorando las restricciones y requisitos (no así la capacidad del curso). Podrá quitarle el permiso posteriormente, incluso si el alumno ya inscribió aquel ramo. En este caso se lleva a cabo el mismo proceso que ocurre cuando un alumno bota un ramo.

4.3. Division de grupos Bummer UC

Los alumnos se dividirán en 10 grupos. Cada grupo tendrá una capacidad máxima de 435 alumnos. Para determinar la asignación de alumnos a grupos, se tendrá en cuenta la "bacanosidad"de cada alumno. Para esto, durante el semestre se realizó una encuesta al alumnado sobre el resto de sus compañeros, donde debían indicar a quienes encontraban "bacanes". Junto a este enunciado encontrará el resultado de esta encuesta, con el cual deberá crear los grupos de Bummer UC según los siguientes criterios:

- Los alumnos con una mayor influencia irán en los primeros grupos.
- La "bacanosidad"de un alumno viene determinada en gran parte por la cantidad de compañeros que lo encuentran bacán.
- No obstante, la cantidad de estudiantes que siguen a un alumno no es lo único que determina su influencia. Si el alumno "Gonzalo Gonzalez"(un alumno muy bacán a opinión de muchas personas) es el único que encuentra bacán a "Fernando Fernandez", entonces este último tendrá de igual forma una gran "bacanosidad"(pues lo sigue alguien muy bacanoso).
- Los dos puntos anteriores implican que la bacanosidad está definida de forma recursiva. Tome su tiempo para pensar cómo modelar este problema.

Los horarios de cada grupo corresponden a los siguientes:

- Grupo 1: 08:30 - 10:30
- Grupo 2: 09:30 - 11:30
- Grupo 3: 10:30 - 12:30
- Grupo 4: 11:30 - 13:30
- Grupo 5: 12:30 - 14:30
- Grupo 6: 13:30 - 15:30
- Grupo 7: 14:30 - 16:30
- Grupo 8: 15:30 - 17:30
- Grupo 9: 16:30 - 18:30
- Grupo 10: 17:30 - 19:30

5. Pacmático

Esta parte de la tarea consiste en modelar e implementar las funcionalidades del sistema Pacmántico. El grupo revolucionario ha pensado en todo, por lo que en caso de fallar Bummer UC, tienen un sistema de toma de ramos de emergencia: Pacmántico. Este sistema le otorga a cada alumno una cantidad de puntos determinada por su "bacanosidad", además de su cantidad de créditos aprobados. Los alumnos tendrán distribuidos por defecto su cantidad total de puntos en cada curso. Cada alumno tendrá la facultad de redistribuir sus puntos según los siguientes criterios:

- Se podrán redistribuir 1000 puntos en total (la suma de la redistribución no debe superar este valor).
- Si a la distribución de un curso se le suman "x"puntos, entonces al resto de las distribuciones se le deben restar los x puntos en partes iguales, de manera que la suma de estos valores sea "x".
- La distribución de puntos de un curso nunca puede resultar negativa.
- Solo se puede redistribuir el puntaje de a lo más 45 créditos.
- A diferencia de Bummer UC, el Pacmático no controla tope horario.

Deberá crear un programa en consola que permita (de manera clara) seleccionar cursos y redistribuir puntos entre ellos, además de otorgar a cada alumno su cantidad de puntos correspondiente.

Esto último se hará de igual forma que la repartición de los grupos de Bummer UC. La cantidad de puntaje viene definida por la siguiente función:

$$\text{Puntaje} = (1 + b_{relativa}/4 + Creds/4000) \cdot 800$$

Donde $b_{relativa}$ es la bacanosidad que tiene con respecto al más bacán de la Universidad y $Creds$ es la cantidad de créditos aprobados que tiene el alumno.

Por último, luego de que todos los alumnos hayan seleccionado cursos y distribuido puntos, su programa debe ser capaz de dar los cursos a los alumnos de una forma óptima, maximizando la suma de puntos efectivos apostados por los alumnos. Los puntos efectivos son aquellos que fueron apostados por un alumno a un ramo en el que *sí* quedó inscrito (Un alumno al que no le otorgan ningún ramo tendría cero puntos efectivos).

6. Carga de Datos

Junto al enunciado encontrará cuatro archivos de texto:

- `personas.txt`
  Posee la información de las personas que se encuentran en el sistema. Para cada persona se detalla:
  - nombre [str]: Nombre y apellido de la persona
  - usuario [str]: Usuario del sistema
  - clave [str]: Clave del sistema
  - alumno [SI / NO]: Diferencia a los alumnos de los profesores
  - idolos [list]: Lista que contiene los nombres de las personas a la que este alumno encuentra bacán. Los profesores tienen esta lista vacía.
  - ramos [list]: Lista que contiene las siglas de los ramos que ya han sido aprobados por el alumno. Los profesores tienen la lista vacía.

- `cursos.txt`
  Posee casi toda la información de cada curso que se dicta este semestre. Para cada curso, se detalla:
  - curso [str]: Nombre del ramo.
  - sigla [str]: Sigla del ramo.
  - NRC [int]: Número identificador del curso.
  - retiro [SI/NO]: Establece si es retirable.
  - eng [SI/NO]: Establece si es dictado en inglés o no.
  - sec [int]: Sección del ramo.
  - apr [Si/NO]: Establece si requiere o no aprobación especial.
  - profesor [str / list]: .ªpellido nombre"del profesor que dicta el curso. En caso de ser más de uno, aparece una lista con todos los profesores.
  - campus [str]: Campus en que se dicta el curso.
  - cred [int]: Cantidad de créditos que otorga.
  - ofr [int]: Cupos totales del curso.
  - ocu [int]: Cupos ya tomados.
  - disp [int]: Cupos disponibles.

  Los siguientes atributos pueden estar o no, dependiendo del curso:
  - hora_cat [str]: Establece el horario de cátedras con el mismo formato que el Buscacursos (dia-dia:módulo).
  - hora_ayud [str]: Idem, pero para las ayudantías.
  - hora_lab [str]: Idem, pero para los laboratorios.
  - sala_cat [str]: Establece la sala en la que se dicta la cátedra.
  - sala_ayud [str]: Idem, pero para las ayudantías.
  - sala_lab [str]: Idem, pero para los laboratorios.

- `requisitos.txt`
  Posee todos los datos necesarios para conocer los prerrequisitos o correquisitos de un curso:
  - sigla [str]: Sigla para identificar el ramo.
  - equiv [str]: Otras siglas equivalentes, por temas de compatibilidad con alumnos antiguos.
  - prerreq [str]: Muestra los requisitos del curos de la forma:
    `"(FIS1503 y MAT1203) o (MAT1202 y MAT1620(c))"`
    donde se ocupan los operadores lógicos *'y'* y *'o'* junto con paréntesis para declarar los requisitos. El detalle *(c)* indica que este ramo es córrequisito y se puede cursar en paralelo con el ramo en cuestión.

- `evaluaciones.txt`
  Posee la información de todas las evaluaciónes. Para toda evaluación se detalla:
  - tipo [str]: Indica el tipo de evaluación.
  - sigla [str]: Indica a qué ramo corresponde esta evaluación.
  - sec [int]: Indica a qué sección del ramo corresponde esta evaluación.
  - fecha [str]: Detalla la fecha en la que será realizada ela valuación en el formato "dd/mm - hh:mm".

7. Notas

- La idea de que un alumno solo pueda tomar ramos en su horario específico es para efectos de la modelación. No hace falta que se tome en consideración la hora real del día. Basta con que al interactuar con su programa mediante consola, el usuario pueda establecer mediante algún input qué hora es, esto debiese bastar para que su programa se adecúe y se comporte tal como si esa fuera la hora real.
- Vea los archivos de texto y entiéndalos bien antes de empezar a hacer la lectura, ya que varios tipos de datos son variables en los archivos.
- Toda la información pertinente a los cursos es la misma que puede encontrar en el buscacursos versión 2015-2 para los ramos correspondientes a ingeniería. Observe la información que este le da acerca de los cursos para entender mejor los contenidos de los archivos.

8. Restricciones y alcances

- Esta tarea es estrictamente individual, y está regida por el Código de Honor de la Escuela: clickear aquí para leer.
- Tu programa debe ser desarrollado en Python 3.4
- Su código debe seguir la guía de estilos `PEP8`
- Si no se encuentra especificado en el enunciado, asuma que el uso de cualquier librería Python está prohibido. Pregunte por foro si se pueden usar librerías específicas.
- El ayudante puede castigar el puntaje[2] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- La revisión de la tarea será realizada con distintos archivos `.txt`.
- Debe adjuntar un archivo `README.md` donde comente sus alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*.
- Cree un módulo para cada conjunto de clases. Divídalas por las relaciones y los tipos que poseen en común.
- Cualquier aspecto no especificado queda a su criterio, siempre que no pase por encima de ningún otro.

9. Entrega

- **Fecha/hora:** 4 de Septiembre - 10:30 AM
- **Lugar:** GIT - Carpeta: Tareas/T01

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[2]Hasta −5 décimas.