Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2019-1

02 de mayo de 2019
**Actividad Formativa**

# Actividad 05

### Excepciones

## Introducción

El Departamento de Ciencia de la Computación de una universidad cercana ha tenido un crecimiento exponencial los últimos años por la gran cantidad de estudiantes que se inscriben a sus *majors* y *minors*. Por este fenómeno, sus servidores colapsan en la época de inscripción de cursos y no logran procesar la información de los alumnos de manera correcta.

El profesor Ristian Cruz comenzó el año pasado a resolver este problema, pero lamentablemente tuvo que dejar el proyecto a medias para emprender nuevos caminos. Su solución fue desarrollar **DCCBanner**, un sistema de inscripción de cursos para el departamento. En él, los alumnos indican qué curso quieren tomar y algunos de sus datos personales. Sin embargo, el módulo de **DCCBanner** quedó incompleto, y no tiene ningún mecanismo que chequee la información que ingresan los alumnos antes de inscribirlos en el curso solicitado.

## DCCBanner

Para esta actividad deberás validar los *input* ingresados por los alumnos en la inscripción de cursos, primero **identificando** los errores que tengan y luego **capturándolos** para poder manejarlos adecuadamente. Para esto, les entregamos el módulo `dccbanner.py`, el que deberán completar con el fin de inscribir a los estudiantes en sus respectivos cursos.

La información ingresada por los alumnos está en el archivo `lista_alumnos.txt`, mientras que la oferta actual de cursos está en `cursos.json` (sección 1). En el módulo principal encontrarás la clase `DCCBanner`, que posee como atributos una lista `alumnos` y un diccionario `cursos`. Este último será el que tendrá el resultado final de la toma de ramos.

## 1. Archivos

A continuación te explicamos cada uno de los archivos y módulos que incluye esta actividad.

- `dccbanner.py`: Es el módulo principal que debes ejecutar para esta actividad. Contiene a la clase `DCCBanner` la que posee los siguientes atributos:
  - `cursos`: Diccionario de los cursos ofrecidos.
  - `alumnos`: Lista con los alumnos que han solicitado inscribir ramos.

Verás que ambos se inicializan vacíos, para poblarlo debes seguir las instrucciónes de la sección 3.1.

- `alumnos.py`: Contiene a la clase `Alumno`. No debes hacer nada con este archivo. Solo es importado desde `DCCBanner` para crear a los alumnos con el método `cargar_alumnos`. Una instancia de `Alumno` posee los siguientes atributos:
  - `nombre`: `String` con el nombre del alumno.
  - `rut`: `String` con el rut del alumno.
  - `num_alumno`: `String` con el número de alumno.
  - `ano_ingreso`: `String` con el año de ingreso del alumno.
  - `carrera`: `String` con la carrera del alumno. Puede ser `"Ingeniería"` o `"College"`.
  - `curso`: `String` con la sigla del curso pedido por el alumno.
  - `seccion`: `String` con la sección pedida del curso solicitado por el alumno.
  - `comentario`: `String` con un comentario ingresado por el alumno. *Just for fun.*
- `excepcion_propia.py`: Contiene la excepción `ErrorDatos`, la que debe ser lanzada bajo las condiciones de la sección 3.1.
- `alumnos.csv`: Es el archivo que contiene todas las solicitudes de inscripción de ramos de los alumnos. Cada línea tiene la información necesaria para crear una instancia de `Alumno`. Puedes cambiarlo por `big_alumnos.csv` si quieres probar un archivo MUCHO más grande.
- `cursos.json`: Es el archivo que contiene la información de todos los cursos de la oferta actual. No necesitas abrirlo, ya que el método `cargar_cursos` se encarga de leerlo y almacenar su información en el atributo `cursos` del DCCBanner, un diccionario que tendrá la siguiente estructura:

```
cursos = {
    "IIC2233": {
        "1": [],
        "2": [],
        "3": [],
        "4": []
    },
    "IIC2143": {
        "1": [],
        "2": [],
    },
    ...
}
```

## 2. Parte I: Identificar los errores

"Nunca confíes en el *input* del usuario" (*Never trust user input*). Este es uno de los más grandes lemas del profesor Ristian, por lo que los encomendó a revisar algunos de los campos ingresados por los alumnos,

específicamente aquellos que son importantes para este proceso: su RUT, su número de alumno y el curso que ingresaron. Para esto, deberás completar los siguientes métodos de la clase **DCCBanner** encargados de levantar las excepciones correspondientes cuando el *input* no corresponda a lo esperado:

- `def chequear_RUT(alumno)`: Recibe una instancia de la clase `Alumno` y chequea que el formato de escritura del RUT del alumno sea válido. En este caso, solo nos interesa que no tenga puntos pero sí guión antes del último dígito, es decir, de la forma: `"19644911-6"`. Los siguientes RUT levantarían excepciones: `"19.644.911-6"`, `"196449116"`. Puedes asumir que sólo se ingresarán RUT válidos, es decir, que el dígito verificador concuerda con el RUT. Todos serán superiores a los 10 millones.

- `def chequear_numero_alumno(alumno)`: Recibe una instancia de la clase `Alumno` y chequea que el número de alumno sea válido bajo las siguientes condiciones:

  - Debe empezar con los dos últimos dígitos del año de ingreso del alumno
  - Inmediatamente después, debe aparecer el código de carrera inscrito por el alumno. Para efectos de esta actividad, solo podrán ser los códigos de Ingeniería o de College: `"63"` o `"61"`, respectivamente.
  - Finalmente, si contiene una letra, solo puede ser la `"J"` y debe estar al final del número

  Ejemplo: Si el alumno es de la generación 2013 y de Ingeniería, obligatoriamente debe empezar con 13 y continuar con 63, por lo que `"14676520"` debería levantar una excepción, al igual que `"J1789023"`, `"13619022"` ó `"1389876K"`.

- `def chequear_curso(alumno)`: Recibe una instancia de la clase `Alumno` y hace dos verificaciones importantes: **existencia** y **formato**.

  - **Existencia**
    - Se debe verificar que exista el curso pedido por el alumno en la oferta actual, es decir, en el diccionario de `cursos` del DCCBanner. Si no existe, se arroja la excepción correspondiente. El manejo se ve en la sección 3.2, en la tabla.
    - Se verifica que exista la sección dentro del curso especificado por el alumno. El manejo se ve en la sección 3.2, en la tabla.
  - **Formato** Se debe lanzar una excepción en caso de los siguientes errores de formato:
    - Si existen espacios en el nombre del curso. Ejemplo: `IIC 2233`
    - Ingresan `"seccion <numero>"` o `"todas"` en vez de sólo el número. Pueden asumir que este error seguirá siempre este formato.

  **Importante**:

  - Si un alumno ingresa un curso IIC que no existe en la oferta actual, éste siempre habrá ingresado la sección 1 de dicho ramo.
  - Los métodos descrito en esta sección no retornan nada.
  - Los cursos de la oferta actual son solamente del DCC (i.e: Empiezan con IIC).
  - Pueden suponer que cualquier número de alumno entregado tendrá, a lo más, una letra.
  - No habrá ningún alumno con errores distintos a los especificados en este apartado.

# 3. Parte II: Capturar excepciones

## 3.1. Cargar los datos a DCCBanner

No tan solo no hubo tiempo para trabajar en la validación, sino que tampoco se pudo trabajar en poblar el módulo con la información de los cursos ofrecidos y las solicitudes enviadas por los alumnos en la inscripción de ramos. Deberás completar el siguiente método para lograrlo:

- `def cargar_datos(archivo_cursos, archivo_alumnos)`: Recibe las rutas de los archivos nombrados en la sección 1 e intenta cargarlos al diccionario de `cursos` y a la lista de `alumnos` del `DCCBanner` a través de los métodos especializados `cargar_cursos` y `cargar_alumnos` respectivamente. Estos últimos te los entregamos y no es necesario que les cambies nada. En caso de que `archivo_cursos` o `archivo_alumnos` no existan, se debe imprimir un mensaje indicando este error e indicando el nombre del archivo que está causando problemas.
  Ejemplo: `"Error: El archivo <nombre> no existe."`

Además de esto, verás que al final del módulo incluimos un poco de lógica para que puedas *debuggear* tu actividad (no debes hacer nada acá, es sólo para facilitar tu trabajo), y también un bloque de **Try/Except/Finally**. En este bloque se usa excepción de `ErrorDatos`, una excepción personalizada que creó uno de los ayudantes del profesor. Dicha excepción debe ser levantada cuando se intenta inscribir a los alumnos en sus cursos, pero el diccionario `cursos` y/o la lista `alumnos` están vacíos. Tu tarea será **levantar** esta excepción en el lugar apropiado - no es necesario manejarla ya que para eso está la sentencia *finally*, que se encarga de poblar el sistema con los archivos correctos.

## 3.2. Inscribir a los alumnos en los cursos pedidos

En esta última parte manejarán los errores que hayan identificado en las solicitudes de los alumnos para poder inscribirlos en los cursos que solicitaron. Para esto, completarán el siguiente método:

- `def inscribir_alumnos()`: Inscribe a cada uno de los alumnos de la lista `alumnos` del `DCCBanner` en el curso solicitado, ingresándolo a la lista de la sección pedida de dicho curso del diccionario `cursos`. Antes de hacerlo, chequea que sus datos estén correctamente ingresados, y en caso de no estarlo, los corrige según la información de la tabla a continuación.

| Descripción | Solución esperada | Ejemplo | Resultado |
|---|---|---|---|
| Los RUT vienen con puntos o sin guión. | Eliminar los puntos o agregar el guión. | 19.644.911-6, 196449116, 19.644.9116 | 19644911-6 |
| El número de alumno no concuerda con su año de entrada, su carrera o tiene letras inválidas. | Cambiar los primeros dos dígitos por el año de ingreso, los segundos dos por la carrera y/o la letra correcta en su lugar. | 14565606 (Ing 2012), J1463626, 1761602K | 12635606, 1463626J, 1761602J |
| El número de la sección no existe para un curso. | Dejarla como sección 1. | 888 | 1 |
| Se escribe texto, en vez de un número. | Dejarla como sección 1. | todas | 1 |
| Se ingresa "seccion N" con N el número de la sección, en vez de sólo el número | Eliminar la palabra, para sólo dejar el número. | sección 5 | 5 |
| El curso tiene un espacio entre la sigla del departamento y la sigla numérica. | Eliminar ese espacio. | IIC 1103 | IIC1103 |
| El curso no está presente en la oferta actual | Si el curso parte con IIC, se agrega al diccionario `cursos`. Si no, se imprime un mensaje que indique que el alumno no será inscrito y se pasa al siguiente. | ▪ ICH1105<br>▪ IIC2413 - no existe en oferta actual. | ▪ Se imprime mensaje<br>▪ Se agrega al diccionario `cursos` de DCCBanner. |

A continuación veremos un ejemplo del funcionamiento de este método:

`alumnos.csv`:

```
Ricardo Schilling;12.345.678-9;14622323;2016;Ingeniería;IIC 2333;1;Seré ayudante!
Enzo Tamburini;108657632;16613540;2017;Ingeniería;IIC2513;1;Por favor quiero egresar
Hernan Valdivieso;18867509-2;J1461616;2014;College;ICH1103;2;Quien quiere ser conmigo?
Christian Eilers;22343621-K;17636530;2017;Ingeniería;IIC2613;todas;Con precálculo pls
```

`cursos.json`:

```
{
    'IIC2233': {
        '1': [],
        '2': [],
        '3': [],
        '4': []
    },
    'IIC2613': {
        '1': [],
        '2': []
    },
    'IIC2333': {
        '1': []
    }
}
```

Luego del proceso de inscripción de cursos, el output de `dccbanner.py` debería verse así:

```
No se pudo inscribir al alumno Hernan Valdivieso, el curso solicitado no es IIC.

Curso: IIC2233
Seccion: 1
**********************************************
Curso: IIC2613
Seccion: 1

Nombre alumno: Christian Eilers
Rut: 22343621-K
Numero alumno: 17636530
Año de ingreso: 2017
Carrera: Ingeniería
Comentario: Con precálculo pls

Seccion: 2

**********************************************
Curso: IIC2333
Seccion: 1

Nombre alumno: Ricardo Schilling
```

```
Rut: 12345678-9
Numero alumno: 16632323
Año de ingreso: 2016
Carrera: Ingeniería
Comentario: Seré ayudante!

**********************************************
Curso: IIC2513
Seccion: 1

Nombre alumno: Enzo Tamburini
Rut: 10865763-2
Numero alumno: 17633540
Año de ingreso: 2017
Carrera: Ingeniería
Comentario: Por favor quiero egresar
```

## Notas

- Recuerden que `string.isalpha()` retorna `True` si todos sus caracteres son letras, `False` en caso de que 1 o más no lo sean. Análogo para chequear si todos los caracteres son números es `isdigit()`.
- Existe la posibilidad de que necesiten usar `ValueError`. Pueden revisar de qué se trata **aquí**.

## Requerimientos

- (3.00 pts) **Parte 1: Levantar excepciones**

  Se levantan las excepciones cuando:

  - (0.75 pts) Se ingresa un RUT con punto o sin guión.
  - (0.75 pts) Se ingresa un número de alumno inválido.
  - (0.75 pts) Se ingresa texto en la sección o la sección no existe en el curso.
  - (0.75 pts) Los cursos ingresados no tienen el formato correcto o no existen.

- (3.00 pts) **Parte 2: Capturar excepciones**
  - (0.5 pts) Se captura la excepción cuando no existan los archivos `archivo_curso` o `archivo_alumno` e se imprime en pantalla el mensaje correspondiente.
  - (0.5 pts) Se levanta la excepción personalizada cuando corresponde.
  - (0.5 pts) Se arregla *outputs* cuando se ingresa un RUT con punto o con guión.
  - (0.5 pts) Se arregla *outputs* cuando se ingresa un número de alumno inválido.
  - (0.5 pts) Se arregla *outputs* cuando la sección no existe o no tiene el formato correcto.
  - (0.5 pts) Se arregla *outputs* cuando los cursos ingresados no existen o no tienen el formato correcto.

## Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC05/`
- **Hora del *push*:** 16:30

## Auto-evaluación

Como esta corresponde a una actividad formativa, te extendemos la instancia de responder, después de terminada la actividad, una auto-evaluación de tu desempeño. Esta se habilitará a las **16:50 de jueves 02 de mayo** y tendrás plazo para responderla hasta las **23:59 del día siguiente, viernes 03 de mayo**. Puedes acceder al formulario mediante el siguiente enlace:

`https://forms.gle/mXYwhxVgBLfjsyzL8`

El asistir, realizar la actividad y responder la auto-evaluación otorgará como bonificación al alumno 2 décimas para sumar en su peor actividad sumativa del semestre.