Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

**IIC2233 Programación Avanzada (2021-1)**

# Tarea 1

## Entrega

- **Avance de tarea**
  - **Fecha y hora:** martes 20 de abril de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`
- **Tarea**
  - **Fecha y hora:** miércoles 28 de abril de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`
- `README.md`
  - **Fecha y hora:** jueves 29 de abril de 2020, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T1/`

## Objetivos

- Aplicar conceptos de programación orientada a objetos (POO) para modelar y resolver un problema.
- Utilizar *properties*, clases abstractas y polimorfismo como herramientas de modelación.
- Comunicar diseños orientados a objetos a través de documentación externa.
- Procesar *input* del usuario de forma robusta, manejando potenciales errores de formato.
- Investigar y aplicar documentación de librerías externas de python.

# Índice

# 1. *DCCanal*

El uso de barcos para transportar mercadería es vital para el mundo globalizado en el que vivimos hoy en día. Si alguna vez has pedido algo desde el extranjero, es posible que este haya viajado en barco hasta que llegara a nuestro país.

Dados tus amplios conocimientos en Programación Orientada a Objetos, has sido puesto a cargo del *DCCanal*, un canal muy importante dentro del mundo del transporte marítimo. Tu misión será administrar el canal para que los barcos que pasen por ahí puedan navegar sin problemas y pasen por el canal con éxito, a pesar de diversos riesgos como accidentes o encallamientos. Tu objetivo será mantener el canal despejado y expedito para poder seguir operando y evitar quedarte sin fondos.


Figura 1: Logo de *DCCanal*

# 2. Flujo del programa

*DCCanal* corresponde a una **simulación por turnos de un canal de navegación**, donde se simularán los eventos que ocurren dentro del canal hora a hora. Deberás modelar todas las entidades involucradas y posteriormente manejar el paso de embarcaciones, realizar acciones de emergencia sobre el canal y calcular el flujo de dinero que entra y sale de este, entre otras cosas.

Al iniciar el programa, deberás primero seleccionar un canal a simular de los disponibles en el archivo `canales.csv`. Cada canal tendrá características distintas que afectarán el cómo progresa la posterior simulación. Partirás con un saldo inicial en dólares (`USD`), y a medida que los barcos crucen por el canal en cada hora simulada, recibirás una remuneración para poder seguir operándolo a pesar de los costos. Además, cada barco tiene un costo de mantención, por lo que, con cada hora que pase, perderás cierta cantidad de dinero por barco cruzando el canal. Este costo de mantención está dado en la moneda del país de origen del barco, por lo que deberás asegurarte de realizar la conversión correcta antes de realizar el cobro al canal.

Cada vez que pase un barco por el canal, existe una probabilidad de que ocurra un evento especial. Algunos de estos eventos detendrán el flujo de barcos por el canal, por lo que tu misión será revertir esta situación lo antes posible. Para esto, tienes disponible la opción de liberar el canal, con cierta probabilidad de éxito. Mientras más te demores en solucionar el problema, y más se demoren en avanzar los barcos, los costos de mantención del canal podrían acabar con tus fondos.

Tu programa deberá poder simular una cantidad indefinida de horas, y mostrar un breve resumen de los eventos ocurridos tras cada hora. Finalmente, el programa acabará automáticamente cuando tus fondos lleguen a 0, significando la bancarrota del canal y el fracaso de la simulación.

# 3. Entidades

En esta sección se detallan las entidades que necesitarás para simular el *DCCanal*. Para modelarlas correctamente, deberás utilizar conceptos clave de **programación orientada a objetos:** herencia, clases abstractas, polimorfismo, *properties* y relaciones, ya sean de agregación o composición. Cada uno de estos elementos debe ser incluido en el programa al menos una vez, y deberás descubrir dónde implementarlos según lo propuesto por el enunciado.

Las entidades principales de *DCCanal* son **Barcos**, **Mercancía**, **Tripulación** y **Canal**. Debes incluir como mínimo las características nombradas a continuación, pero siéntete libre de añadir atributos y métodos adicionales.

## 3.1. Barcos

Corresponden a las embarcaciones que desean cruzar el *DCCanal*. Son manejadas por los **tripulantes** y cargan **mercancía** y, posiblemente, pasajeros. Estos deben pagar por hacer uso de un canal, se mueven a ciertas velocidades que les permiten salir en una cantidad determinada de turnos de este. Además, tienen la probabilidad de encallar durante el transcurso de su viaje, lo que impedirá tanto el movimiento propio como el de todos los barcos que vengan tras este. Los barcos poseen los siguientes atributos:

- **Nombre**: corresponde a un `str` que representa el nombre del barco simulado.
- **Costo de mantención**: un `float` que corresponde al valor de mantención que el canal debe pagar por cada hora que el barco se encuentre viajando a través de él. Cabe notar que esté valor se encuentra en la divisa o moneda que utiliza dicho barco, por lo que será necesario hacer la conversión a dólares (`USD`) antes de realizar el cobro.
- **Velocidad Base**: un `int` que corresponde a la velocidad base a la que el barco recorre el canal asignado.
- **Pasajeros**: todos los barcos, además de llevar mercancía, pueden llevar pasajeros. Es un `int` que corresponde a la cantidad de pasajeros **no pertenecientes a la tripulación** que lleva el barco.
- **Carga Máxima**: es la carga máxima que puede llevar un barco, específica para cada tipo de barco. Está representada por un `int`.
- **Moneda de origen**: corresponde a una variable tipo `str` que contiene el nombre abreviado de la divisa que utiliza dicho barco.
- **Probabilidad de encallar**: corresponde a la probabilidad base de que un barco sufra un accidente. Depende del tipo de barco.
- **Tripulación**: almacena a los miembros de la Tripulación que maneja el barco.
- **Mercancía**: almacena toda la mercancía que lleva el barco a bordo.

Además, los barcos pueden realizar las siguientes acciones:

- **Desplazarse**: Dependiendo de la velocidad base de cada barco, la carga total que este posea y los pasajeros que lleva, el barco se desplazará en una hora o iteración de la simulación esta cantidad de

kilómetros según la siguiente fórmula:

$$max(0.1, min(1, \frac{carga\_maxima - mercancia - 0.3 * pasajeros}{carga\_maxima})) * vel\_base \quad (1)$$

Donde:

- *carga_maxima* corresponde a la carga máxima del barco.
- *mercancía* corresponde al peso total de toda la mercancía que lleve el barco.
- *pasajeros* corresponde a la cantidad de pasajeros que lleve el barco.
- *vel_base* corresponde a la velocidad base del barco.

Esto significa que a lo menos, el desplazamiento del barco será $\frac{1}{10}$ de la velocidad del barco sin peso.

- **Encallar:** En cada hora en el que el barco se encuentre navegando dentro del canal, existe el riesgo de que sufra un accidente y encalle, bloqueando el paso a sí mismo y para todos los barcos que vengan tras este. La probabilidad de encallar en cierto momento es según la siguiente fórmula:

$$min(1, \frac{vel\_base + mercancia - exp\_tripulacion}{120}) * tend\_encallar * dif\_canal \quad (2)$$

Donde:

- *vel_base*: corresponde a la velocidad base del barco.
- *mercancia*: corresponde al peso total de la mercancia del barco.
- *exp_tripulacion*: corresponde a la suma de los años de experiencia de todos los miembros de la tripulación a bordo del barco.
- *tend_encallar*: corresponde a la tendencia que tiene el barco a encallar, y su valor dependerá del tipo de barco que sea.
- *dif_canal*: corresponde a un ponderador que tomará el valor `PONDERADOR_PRINCIPANTE` o `PONDERADOR_AVANZADO`, según la dificultad del canal que se está simulando.

- **Ejecutar evento especial**: Cada barco es susceptible a un evento especial en el transcurso de su viaje. Este evento especial tiene una probabilidad de `PROBABILIDAD_EVENTO_ESPECIAL` de ocurrir en cada turno y puede ocurrir solo **una vez por barco**. Los efectos del evento especial dependen del tipo de barco.

### 3.1.1. Tipos de barcos

Cada barco puede ser de uno de estos tres tipos, los cuales tienen valores diferentes para sus atributos y poseen un evento especial único:

- **Barco de pasajeros**: el barco de pasajeros puede llevar poca carga, ya que la mayoría de su capacidad la usan sus pasajeros. Su tendencia de encallamiento es de `TENDENCIA_ENCALLAR_PASAJEROS`. El evento especial de este tipo de barco corresponde a la intoxicación masiva de los pasajeros a bordo, lo que provoca que el barco pague una cantidad `DINERO_INTOXICACION` de dinero al canal a cambio de medicamentos para sus pasajeros.
- **Barco Carguero**: el barco carguero tiene una capacidad mediana de carga y se dedica exclusivamente a llevar mercancía por el canal. Su tendencia de encallamiento es de `TENDENCIA_ENCALLAR_CARGUERO`.

El evento especial de este barco corresponde a ser atacado por piratas y perder toda la mercancía a bordo, haciendo que no pueda pagar la tarifa de salida del canal.

- **Buque**: el buque es el tipo de barco más grande, que puede cargar con tanto pasajeros como con mercancía. Su tendencia de encallamiento es de `TENDENCIA_ENCALLAR_BUQUE`. El evento especial del buque corresponde a una avería interna, que no le permite avanzar por el canal por una cantidad `TIEMPO_AVERIA_BUQUE` de horas. Este evento solo afectará al buque afectado, por lo que el resto de los barcos podrá seguir avanzando sin problemas.

### 3.2. Caja de Mercancía

Transportada por barcos, las cajas de mercancía que estos carguen afectan en cuánto tarda un barco en moverse debido a su peso. Cada barco puede cargar una o más cajas de mercancía, y no es necesario que todas sean del mismo tipo. Los atributos que tiene cada caja de mercancía son:

- **Número de lote:** corresponde a un `int` único que permite identificar a la caja de mercancía del resto.
- **Tipo**: corresponde a un `str` que representa el tipo de mercancía que se lleva. Este puede ser **petróleo**, **ropa** o **alimentos**.
- **Tiempo expiración**: un `int` que corresponde a la cantidad de horas máxima en la que estaba presupuestado que la mercancía cruzara el canal.
- **Peso**: corresponde a un `int` que representa el peso en kilos de la caja de mercancía.

Además, la mercancía puede realizar la siguiente acción:

- **Expirar:** Cuando el tiempo de viaje del barco excede el tiempo de expiración de la caja de mercancía, esta expirará y se deberá cobrar al canal un monto de `MULTA_PETROLEO`, `MULTA_ROPA` o `MULTA_ALIMENTOS`, dependiendo del tipo de mercancía que se lleve a bordo.

### 3.3. Tripulación

Son personas que trabajan a bordo de los barcos, procurando que tengan un viaje seguro. Afectan la probabilidad de encallamiento y poseen diferentes efectos especiales que afectan al barco que viaja por el canal. Cada barco tiene entre 1 y 3 tripulantes, y puede haber como máximo un tripulante de cada tipo. Sus atributos son:

- **Nombre:** un `str` que contiene el nombre del tripulante.
- **Años de experiencia**: corresponde a los años de experiencia en el rubro que tiene el tripulante.

Además, cada tripulante tiene un efecto especial sobre el barco en que trabaja, siendo este efecto dependiente del tipo de tripulante que sea. Los efectos especiales de cada tripulante a bordo del barco se aplican automáticamente, sin necesidad de activarlos.

#### 3.3.1. Tipos de tripulantes:

- **DCCapitán:** el capitán es quien maneja el barco a través de canal. Su efecto especial le permite automáticamente desencallar al barco la **primera vez** que este encalle en el canal.
- **DCCocinero:** el cocinero se preocupa de mantener alimentada a la tripulación, su habilidad es cocinar. Su efecto especial le permite duplicar el **tiempo de expiración** de la mercancía que sea de tipo **Alimento**.

- **DCCarguero:** el carguero se preocupa de la mercancía, que no se pierda y se mantenga bien. Su efecto especial le permite aumentar la **carga máxima** del barco en `CARGA_EXTRA_CARGUERO`.

### 3.4. Canales

Son la entidad principal que maneja el programa y representan al canal que se desea simular. Tiene como atributos:

- **Nombre**: es una variable tipo `str` que corresponde al nombre del canal.
- **Dinero**: corresponde a un `float` que almacena el dinero actual del canal en dólares (`USD`). Cada canal comienza con una cantidad `DINERO_INICIAL` de dinero.
- **Cobro de uso**: corresponde al cobro en dólares (`USD`) que se le hace a un barco por usar el canal una vez este lo termina de cruzar. Es una variable tipo `int` y su valor es un depende de la dificultad del canal, pudiendo tomar el valor `COBRO_USO_PRINCIPIANTE` o `COBRO_USO_AVANZADO`.
- **Largo**: corresponde a un `int` que define la longitud del canal en kilómetros.
- **Dificultad**: corresponde a un `str` que define la dificultad del canal. Este puede ser `'Principiante'` o `'Avanzado'`.
- **Barcos**: almacena a todos los barcos que actualmente se encuentran dentro del canal.

El canal puede realizar las siguientes acciones:

- **Ingresar barco al canal**: esta acción permite seleccionar e ingresar uno de los barcos del archivo `barcos.csv` que se encuentre esperando fuera del canal a este, para que comience su viaje. Siempre se deberá ingresar como máximo un barco por hora, y este debe ser un barco que no esté actualmente dentro del canal. Cabe notar que un barco que ya haya cruzado el canal exitosamente puede volver a ser ingresado.
- **Avanzar barcos:** esta acción permite simular una hora en el canal, efectuando todas las acciones descritas en Simular nueva hora.
- **Desencallar barco:** en caso de que un barco encalle en el canal, bloqueando el paso a otras embarcaciones, el canal puede intentar desatascarlo con un costo de `COSTO_DESENCALLAR` y una probabilidad de éxito dada por:

$$\texttt{PROB\_BASE\_DESENCALLAR} * ponderador\_dificultad \tag{3}$$

  donde $ponderador\_dificultad$ toma el valor `PONDERADOR_PRINCIPIANTE` o `PONDERADOR_AVANZADO` dependiendo de la dificultad del canal.

## 4. Menús

Para esta tarea, la simulación del *DCCanal* será realizada a través de una serie de Menús en la consola. Estos menús deben ser a prueba de **todo tipo de errores** de usuario, y adicionalmente, cada uno debe tener la opción de **volver atrás** y **salir**. A continuación se explicarán los menús mínimos a incluir. Si lo consideras necesario, puedes añadir más menús que los explicitados.

### 4.1. Menú de Inicio

Al ejecutar el programa se deberá desplegar un menú de bienvenida en donde permita al usuario seleccionar la opción de **Comenzar una nueva simulación** o **Salir** del programa. Este menú es el único donde no es necesaria la opción de volver.

Si se selecciona **Comenzar nueva simulación**, el usuario deberá indicar qué canal desea simular de los almacenados en el archivo `canales.csv`.

Una vez cumplido lo anterior, se dirigirá al usuario al **menú de acciones**.

## 4.2. Menú Acciones

Este corresponde al menú principal de la simulación, en donde el usuario podrá tomar diferentes decisiones. Este menú contiene la opción de **Mostrar riesgo de encallamiento**, **Desencallar Barco**, **Simular nueva hora** y **Mostrar Estado** de la simulación.

### 4.2.1. Mostrar riesgo de encallamiento

Al seleccionar esta opción, se deberá visualizar la probabilidad de que una embarcación que cruce por el canal se quede atascada en este. Esto ocurre imprimiendo en la consola del usuario un listado de **todos los barcos que se encuentren dentro del canal** y su respectiva probabilidad de encallar.

### 4.2.2. Desencallar Barco

Si se elige esto, se desplegaran en la consola un listado de todos los barcos que actualmente se encuentren encallados y se dará la opción de seleccionar uno. Se deberá primero verificar que se tengan suficientes fondos y en caso de no poseerlos, se deberá impedir la acción.

En caso de sí poseer el dinero necesario, el programa deberá intentar desencallar el barco para que el canal vuelva a fluir, y mostrar en consola si la acción fue exitosa o no.

### 4.2.3. Simular nueva hora

Al seleccionar esta opción el programa deberá simular el paso de una hora en el canal.

Primero se deberá indicar si hay barcos atascados o no en el canal. En caso de que no haya ningún barco atascado, a partir del archivo `barcos.csv` se deberá mostrar una lista de todos los barcos que no se encuentren dentro del canal actualmente. El usuario deberá seleccionar a lo más uno de los barcos disponibles para que entre al canal y empiece a avanzar a través de este. En caso de que el canal se encuentre bloqueado, no se deberá dejar entrar ningún barco nuevo al canal.

A continuación, el programa deberá calcular si el barco encalló o no en base a su probabilidad de encallar. En caso de no encallar, deberá calcularse a continuación si el barco ejecutó su propio evento especial. Luego, se debe simular el movimiento de los barcos dentro del canal, haciendo que cada barco avance según su propia velocidad.

Una vez que un barco haya terminado de desplazarse, se debe verificar si este salió del canal. Cuando una embarcación termina de cruzar el canal, le paga a este el costo de uso de utilizar el canal. Por el contrario, por cada hora que un barco se mantenga en el canal, este último deberá pagarle al barco su costo de mantención, debido a costos logísticos y operacionales. En este último caso, al usar los barcos su propia divisa, deberás convertir el **costo de mantención** de cada barco desde su propia moneda de origen a la moneda que utiliza el canal, que son los dólares (`USD`), antes de realizar el cobro al canal. Para lograr esto, deberás investigar y utilizar la librería externa de python **CurrencyConverter**.

Todos los eventos detallados anteriormente deberán mostrarse en consola cuando ocurran.

Al final de cada hora, debe ser posible visualizar un resumen que contenga los nombres de todos los barcos que están actualmente en el canal, junto con su posición dentro de este y un reporte de los movimientos de dinero, que contenga el dinero recaudado y los gastos incurridos durante la hora simulada.

Una vez finalizada la simulación de la hora, se debe volver al menú anterior.

#### 4.2.4. Mostrar Estado

Al seleccionar esta opción, se deberá desplegar en consola toda la información relevante del estado de la simulación. Esta deberá incluir como mínimo: el **nombre, largo y dificultad del canal**, la **cantidad de horas simuladas** en total, el **balance de dinero actual**, el **dinero recibido y dinero gastado** total, la **cantidad de barcos que han pasado**, **cantidad de barcos que han encallado** y **cantidad de eventos especiales ocurridos**. Puedes incluir más información que consideres relevante.

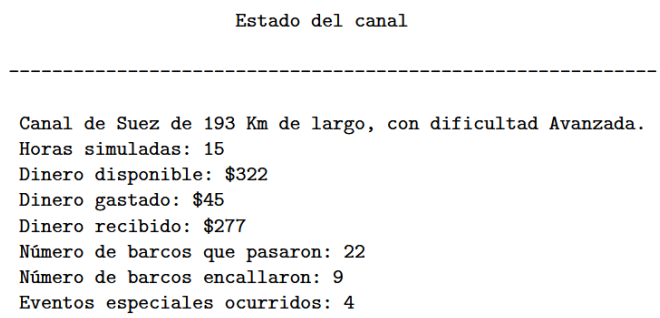


Figura 2: Ejemplo estado canal

## 5. Archivos

En esta tarea se te entregarán tres archivos de extensión `.csv` con la información correspondiente de los barcos, los canales y tripulantes. Además, deberás crear y rellenar un archivo `parametros.py`, el cual almacene todos los parámetros presentados a lo largo del enunciado.

### 5.1. `barcos.csv`

Este archivo contiene información acerca de los barcos que se trasladan por los canales del *DCCanal*. En la siguiente tabla puedes encontrar la información presente de cada barco en el archivo:

| Nombre | Tipo de Dato | Descripción |
|---|---|---|
| Nombre | `str` | Nombre del barco. |
| Tipo | `str` | Tipo de embarcación. Puede ser `'Pasajero'`, `'Carguero'` o `'Buque'`. |
| Costo de mantención | `int` | valor de mantención asociado al barco. |
| Velocidad base | `float` | Float que indica la velocidad base que tendrá el barco. |
| Pasajeros | `int` | Cantidad de pasajeros dentro del barco. |
| Carga máxima | `int` | Indica cuanto peso soporta como máximo un barco. |
| Moneda de Origen | `str` | Sigla que representa la divisa que utiliza el barco. |
| Tripulación | `list` | lista con los nombres de los tripulantes, separados por (`';'`). |
| Carga | `list` | lista con los números de lotes de las cajas que carga del barco, separados por (`';'`). |

Un ejemplo de cómo se verían los datos de `barcos.csv` es el siguiente:

```
El Aventura,Pasajero,130,48,757,500,JPY,Heather Sparks,108;6
Barbablanca,Buque,310,42,394,560,AUD,Thomas Vaughan;Kevin Hancock,13;126;54;169;147
```

```
Caribe,Carguero,200,41,318,630,DKK,Christopher Gray;David Saunders,103;167;95;87;41;53
Pachamama,Buque,310,39,368,560,BGN,Matthew Cooley,171;8;115;107;71;1
```

### 5.2. `canales.csv`

Este archivo incluye información sobre los diferentes canales habilitados para el tránsito de tus embarcaciones. Puedes ver la descripción del atributo y el tipo de dato que debes trabajar en tu programa a continuación:

| Nombre | Tipo de Dato | Descripción |
|---|---|---|
| Nombre | `str` | Nombre del canal. Puede ser `'Canal de Panamá'` o `'Canal de Suez'`. |
| Tamaño | `int` | Tamaño del canal en kilómetros. Esto definirá cuanto tarden las embarcaciones en cruzarlo. |
| Dificultad | `str` | Un canal puede tener dificultad `'Principiante'` o `'Avanzado'` |

Un ejemplo de cómo se verían los datos sería el siguiente:

```
nombre,tamaño,dificultad
Canal de Panamá,80,principiante
Canal de Suez,193,avanzado
```

### 5.3. `tripulantes.csv`

Este archivo contiene información sobre los tipos de tripulantes posibles y sus atributos. En la siguiente tabla se muestra cada atributo, el tipo de dato que debes manejar en tu programa y su descripción correspondiente:

| Nombre | Tipo de Dato | Descripción |
|---|---|---|
| Nombre | `str` | Nombre del tripulante. |
| Tipo | `str` | Tipo de tripulante. Puede ser `'DCCapitán'`, `'DCCocinero'` o `'DCCarguero'`. |
| Años de experiencia | `int` | Variable de tipo `int` que indica los años de experiencia de la persona. |

Un ejemplo del archivo es el siguiente:

```
nombre,tipo,años de experiencia
Benjamin Parsons,DCCarguero,20
Jackson Thomas,DCCapitán,19
Glenn Campbell,DCCarguero,10
```

### 5.4. `mercancia.csv`

Este archivo contiene información sobre las cajas de mercancía disponibles y sus atributos. En la siguiente tabla se muestra cada atributo, el tipo de dato que debes manejar en tu programa y su descripción correspondiente:

| Nombre | Tipo de Dato | Descripción |
|---|---|---|
| Lote | `int` | Número de lote de la caja, permite identificarla. |
| Tipo | `str` | Tipo de caja de mercancía. Puede ser `'petróleo'`, `'ropa'` o `'alimentos'`. |
| Tiempo de expiración | `int` | Variable de tipo `int` que indica la cantidad de horas máximas en la que estaba presupuestado que la mercancía cruzara el canal. |
| Peso | `int` | Representa el peso en kilos de la caja. |

Un ejemplo del archivo es el siguiente:

```
105,alimentos,13,94
106,petróleo,3,92
107,ropa,3,91
108,ropa,12,83
```

### 5.5. `parametros.py`

Para esta tarea, deberás crear un archivo `parametros.py` y completarlo con todos los parámetros mencionados a lo largo del enunciados, los cuales encontrarás en `ESTE_FORMATO`. Además, debes agregar cualquier valor constante en tu tarea, junto con los *paths* que utilices.

Recuerda que los parámetros deben ser descriptivos y reconocibles:

```
CINCUENTA = 50   # mal parámetro
COBRO_USO_CANAL_PRINCIPIANTE = 100   # buen parámetro
COBRO_USO_CANAL_AVANZADO = 200   # buen parámetro
```

Si necesitas agregar algún parámetro que varíe de acuerdo a otros parámetros, una correcta parametrización sería la siguiente:

```
ALTURA = 13
AREA_CIRCUNFERENCIA = 6
VOLUMEN_CILINDRO = ALTURA * AREA_CIRCUNFERENCIA
```

Dentro del archivo `parametros.py` deberás hacer uso de todos los parámetros almacenados y deberás importarlos correctamente. Si se almacena cualquier información no correspondiente a parámetros, esto tendrá un descuento en tu nota. Por último, no está permito que un parámetro se encuentre ***harcodeado***[1], ya que es una mala práctica y su uso conlleva un descuento.

Para esta tarea, el archivo `parametros.py` no se debe ignorar y debes subirlo a tu repositorio. En caso contrario, tu tarea no se podrá corregir.

## 6. Avance de tarea

En conjunto con el programa, deberás realizar un **diagrama de clases** modelando las entidades necesarias del *DCCanal*. Este diagrama se entregará en dos ocasiones:

[1] *Hard-coding* es la mala práctica de incrustar datos directamente en el código fuente del programa, en vez de obtener los datos de una fuente externa.

Una versión preliminar que refleje cómo planeas modelar tu programa, que corresponderá al **avance** de esta tarea. A partir de los avances entregados, se te brindará un *feedback* general de lo entregado y además, te permitirá optar por **hasta 2 décimas** adicionales en la nota final de tu tarea.

Luego de esto, junto a la entrega final, deberás entregar una **versión final** de tu diagrama que **represente fielmente** la modelación del problema usada en tu programa.

En ambos casos, el diagrama deberá:

- Entregarse en **formato PDF o de imagen**[2].
- Contener todas las clases junto con sus atributos y métodos. También deberás identificar cuáles clases serán abstractas y cuáles no.
- Contener todas las relaciones existentes entre las clases (agregación, composición y herencia).
- No es necesario indicar la cardinalidad ni la visibilidad (público o privado) de los métodos o atributos.

Para realizar el diagrama de clases te recomendamos utilizar **draw.io**, **lucidchart** o aplicaciones similares.

Es conveniente adjuntar a tu diagrama un documento con una explicación general de tu modelación. Esto es con el fin de ayudar en la corrección del ayudante a comprender tu razonamiento.

Tanto el diagrama (en formato PDF o de imagen) como la explicación de su modelación (en formato **Markdown**) deben ubicarse en la misma carpeta de entrega de la tarea.

## 7. `.gitignore`

Para esta tarea **deberás utilizar un** `.gitignore` para ignorar los archivos indicados, este deberá estar dentro de tu carpeta `Tareas/T1/`. Puedes encontrar un ejemplo de `.gitignore` en el siguiente ***link***.

Para esta tarea, los archivos a ignorar corresponden a las bases de datos entregadas para la simulación. Es decir, deberás ignorar:

- `barcos.csv`
- `canales.csv`
- `tripulantes.csv`
- `mercancia.csv`

Se espera que no se suban archivos autogenerados por las interfaces de desarrollo o los entornos virtuales de Python, como por ejemplo: la carpeta `__pycache__`.

Para este punto es importante que hagan un correcto uso del archivo `.gitignore`, es decir, los archivos no **deben** subirse al repositorio debido al archivo `.gitignore` y no debido a otros medios.

## 8. Entregas atrasadas

Posterior a la fecha de entrega de la tarea se abrirá un formulario de Google Form, en caso de que desees que se corrija un *commit* posterior al recolectado, deberás señalar el nuevo *commit* en el *form*.

El plazo para rellenar el *form* será de 24 horas, en caso de que no lo contestes en dicho plazo, se procederá a corregir el *commit* recolectado.

[2] Cualquier otro formato no será considerado como una entrega válida y no tendrá décimas de avance o puntaje en la tarea.

## 9. Importante: Corrección de la tarea

Acá se deben indicar los aspectos importantes de la corrección, se debe actualizar según la pauta de cada tarea.

Para esta tarea, el carácter funcional del programa será el pilar de la corrección, es decir, **sólo se corrigen tareas que se puedan ejecutar**. Por lo tanto, se recomienda hacer periódicamente pruebas de ejecución de su tarea y *push* en sus repositorios.

Cuando se publique la distribución de puntajes, se señalará con color **amarillo** cada ítem que será evaluado a nivel de código, todo aquel que no esté pintado de amarillo significa que será evaluado si y sólo si se puede probar con la ejecución de su tarea.

En tu archivo `README.md` deberás señalar el archivo y la línea donde se encuentran definidas las funciones o clases relacionados a esos ítems.

Finalmente, si durante la realización de tu tarea se te presenta algún problema o situación que pueda afectar tu rendimiento, no dudes en contactar al ayudante jefe de Bienestar a su **correo**.

## 10. Restricciones y alcances

- Esta tarea es **estrictamente individual**, y está regida por el **Código de honor de Ingeniería**.
- Tu programa debe ser desarrollado en Python 3.7.
- Tu programa debe estar compuesto por uno o más archivos de extensión `.py`.
- Si no se encuentra especificado en el enunciado, supón que el uso de cualquier librería Python está prohibido. Pregunta en la *issue* especial del **foro** si es que es posible utilizar alguna librería en particular.
- Debes adjuntar un archivo `README.md` **conciso y claro**, donde describas los alcances de tu programa, cómo correrlo, las librerías usadas, los supuestos hechos, y las referencias a código externo. **Tendrás hasta 24 horas después del plazo de entrega** de la tarea para subir el `README` a tu repositorio.
- Tu tarea podría sufrir los descuentos descritos en la **guía de descuentos**.
- Entregas con atraso de más de 24 horas tendrán calificación mínima (1,0).
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Las tareas que no cumplan con las restricciones del enunciado obtendrán la calificación mínima (1,0).