Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2021-1

8 de Abril de 2021
Actividad Sumativa

Actividad Sumativa 1

Programación Orientada a Objetos I y II

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AS1/`
- **Hora del *push*:** 16:30

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

Estamos en el año 3021 y muchas cosas han cambiado, ahora los computadores programan a los humanos, el ramo de Programación Avanzada se enseña en las salas cuna y los celulares son literalmente parte de nosotros (todos tienen su celular directamente conectado con su cerebro). Sin embargo, hay algo que nunca cambia y hay un lugar donde todo el mundo siempre quiere ir: el mall. Es por esto que el DCC se ha hecho con la suya y ha decidido crear su propio centro comercial: el **DCCostaneraCenter**.

En este contexto, como estudiante de Programación Avanzada, te han contratado para ayudar. Tu misión es crear un programa que permita organizar el funcionamiento del mall, administrando los locales, realizando las ventas, calculando las utilidades, entre otros. Además, es necesario seguir ciertas medidas de seguridad, como respetar los aforos y revisar el estado de los clientes ya que hay un extraño virus computacional, el Progravirus, en los aparatos eléctricos que no permite que las personas puedan acceder a sus repositorios de Git favoritos. Es tu deber preocuparte de que se cumplan todas estas normas para mantener todo en orden.

Flujo del programa

Importante: Dado que hay varías clases en el programa, **te recomendamos que leas todo el enunciado antes de comenzar a programar.** Además, **se incluye un diagrama de clases al final del enunciado**, que puede ayudarte a entender las relaciones entre ellas.

El programa corresponde a una simulación de un mall que ocurre dentro de la clase principal `Mall`, la cual contiene a sus respectivos locales y clientes. Este crea las instancias de `Mall`, `Local`, `Trabajador` y `Cliente` a partir de una base de datos entregada. Una vez creadas las instancias, se comienza la simulación, donde cada cliente intenta ingresar a su local favorito dentro del mall. Un cliente puede ser rechazado de un local si está contagiado con Pogravirus o si el local está lleno. Una vez dentro del local, la simulación realiza la interacción entre el local y cliente, la cual consiste en la venta de un producto aleatorio del local. Finalmente, para cada local muestra un resumen del día en base a las interacciones con los clientes.

Archivos

Para esta actividad se te hará entrega de los siguientes archivos:

Código

- `main.py`: Este es el archivo principal del programa. Puedes ejecutarlo para probar el funcionamiento de tu programa completo. **Ya viene implementado** y **NO debes modificarlo.**
- `mall.py`: Contiene la clase `Mall`, la cual **debes completar** según lo pedido más adelante.
- `locales.py`: Contiene las clases `Local`, `Entretenimiento`, `Comida`, `Tienda`, `Casino` y `Supermercado`, las cuales **debes completar** según lo pedido más adelante.
- `personas.py`: Contiene las clases `Persona`, `Cliente` y `Trabajador`, las cuales **debes completar** según lo pedido más adelante.
- `cargar_datos.py`: Contiene funciones que extraen la información de los locales, clientes y trabajadores (archivos de datos). Luego, instancia a cada entidad con su respectiva clase. **Ya viene implementado** y **NO debes modificarlo.**
- `parametros.py`: Contiene las rutas de los archivos `.cvs`. **Ya viene implementado** y **NO debes modificarlo.**

Datos

La información contenida en estos archivos se carga en `cargar_datos.py`, módulo que **NO debes modificar**. **NO debes modificar estos archivos.**

- `locales.csv`: En este archivo, encontrarás la información de los locales del mall. La primera línea del archivo (*header*) contiene el nombre de las columnas y el resto de las líneas las características de los locales, separadas por coma, de la forma:

  ```
  categoria,nombre,aforo,productos
  ```

  Los productos tienen el siguiente formato: `producto1:precio1-producto2:precio2`, es decir, el nombre de un producto se une a su precio mediante `:` y además se separan varios productos con `-`. Por ejemplo, una línea sería de la forma:

  ```
  supermercado,Jumbo,30,tallarines:600-toalla_nova:990-agua:2190-CocaCola:1290
  ```

- `trabajadores.csv`: En este archivo, encontrarás la información de los trabajadores de los locales que pertenecen al mall. La primera línea del archivo (*header*) contiene el nombre de las columnas y el resto de las líneas las características de los trabajadores: edad, contagiado (*boolean* representado por 0 o 1), su sueldo y el nombre del local donde trabaja. Todas estas características están separadas por una coma de la forma:

  `nombre,edad,contagiado,sueldo,local`

- `clientes.csv`: En este archivo, encontrarás la información de los clientes del mall. La primera línea del archivo (*header*) contiene el nombre de las columnas y el resto de las líneas las características de los clientes, separadas por coma, de la forma:

  `nombre,edad,contagiado,local,dinero`

Parte 1: Modelación de Personas

En esta parte, deberás completar las clases `Persona`, `Cliente` y `Trabajador`, las cuales se encuentran definidas en el archivo `personas.py`. Deberás definirlas correctamente en cuanto a sus atributos, métodos y relaciones. Recuerda que al final del enunciado hay un diagrama de clases que te puede ayudar a visualizar las relaciones.

Clase Persona

- `class Persona`: Esta es la clase que define a todas las personas que se encuentran dentro del **DCCostaneraCenter**. Representa a todo `Cliente`, y `Trabajador`, que deben heredar de esta clase. Debes definirla como **abstracta**.
  - `def __init__(self, nombre: str, edad: int, contagiado: bool)`: **Este método ya viene implementado.** Recibe el `nombre` de una `Persona`, su `edad` y un booleano que indica si está `contagiado` o no. Son asignados a los atributos `self.nombre`, `self.edad` y `self.contagiado` respectivamente. Estos son los únicos atributos de la clase.
  - `def saludar(self)`: Deberás definir esto como un **método abstracto**. Puedes dejar un `pass` al crear el método, lo importante es que este definido correctamente como un **método abstracto**.

Clase Cliente

- `class Cliente`: Esta es la clase que va a definir a los clientes del **DCCostaneraCenter**. Deberás completarla correctamente tal que herede de `Persona`.
  - `def __init__(self, nombre: str, edad: int, contagiado: bool, nombre_local_favorito: str, dinero: int)`: Deberás completar este método pasando los argumentos que correspondan a la **superclase** y declarando los atributos restantes. También, debes llamar al método `saludar`. Además, deberás agregar los siguientes atributos:
    - `self.nombre_local_favorito`: Un atributo de tipo `str` que representa el nombre del local favorito del cliente.
    - `self.dinero`: Atributo de tipo `int` que representa el dinero que tiene el cliente para realizar sus compras.
  - `def saludar(self)`: Este método ya viene implementado. Imprime el nombre del cliente y su gusto. Por ejemplo:

```
"Hola mi nombre es Antonio Ossa y mi local favorito es McGoofy"
```

Clase Trabajador

- `class Trabajador`: Esta es la clase que va a definir a los trabajadores (vendedores de tienda) del **DCCostaneraCenter**. Deberás completarla correctamente tal que herede de `Persona`.
  - `def __init__(self, nombre: str, edad: int, contagiado: bool, sueldo: int, nombre_local: str)`: Deberás completar este método pasando los argumentos que correspondan a la **superclase** y declarando los atributos restantes. También, debes llamar al método `saludar`. Además, deberás implementar los siguientes atributos:
    - `self.sueldo`: Un atributo de tipo `int` que representa el sueldo del trabajador del local.
    - `self.nombre_local`: Un atributo de tipo `str` que corresponde al nombre del local donde trabaja el `Trabajador`.
  - `def saludar(self)`: Este método ya viene implementado. Imprime el nombre del trabajador, su sueldo y el nombre del `Local` en el que trabaja. Por ejemplo:

```
"Soy Cristian Ruz, trabajo en Sarita y mi sueldo es 500"
```

  - `def generar_posible_contagio(self): -> bool`: Este método ya viene implementado. Crea un número aleatorio entre 0 y 1, utilizando el método `uniform` de la librería `random`, que representa la probabilidad de que el `Trabajador` se contagie al atender a un `Cliente` con progravirus. Si la probabilidad de contagio es menor a 0.1 se actualiza el atributo `contagiado` del `Trabajador` a `True`.

Parte 2: Modelación de Locales

En esta parte, deberás completar las clases `Local`, `Entretenimiento`, `Comida`, `Tienda`, `Casino` y `Supermercado`, las cuales se encuentran definidas en el archivo `locales.py`. Deberás definirlas correctamente en cuanto a sus atributos, métodos y relaciones. Recuerda que al final del enunciado hay un diagrama de clases que te puede ayudar a visualizar las relaciones.

Clase Local

- `class Local`: Esta es la clase que define a todos los locales de **DCCostaneraCenter**. Deberás completar esta clase, definiéndola como **abstracta**. Tiene los siguientes métodos:
  - `def __init__(self, productos: dict, nombre: str, aforo: int, trabajador: Trabajador)`: Recibe el nombre del local, un diccionario con productos, un `int` con el aforo permitido y una instancia de la clase `Trabajador`. Sus atributos son los siguientes:
    - `self.clientes`: Es una lista de instancias de clase `Cliente`.
    - `self.productos`: Es un diccionario en el cual su llave es el nombre del producto y el valor es el precio el producto.
    - `self.nombre`: Es un `str` con el nombre del local.
    - `self.aforo`: Es un `int` que corresponde al numero del aforo permitido en el local.

- `self.trabajador`: Es una instancia del objeto `Trabajador`, representa al trabajador del local.
- `self.abierto`: Es un `bool` que indica si el local esta abierto o no. Es inicializado en `True` si es que el trabajador no está contagiado. `False` en caso contrario.
- `self.__utilidades`: Es un atributo privado de tipo `int`, que representa las utilidades o ganancias del local. Estas serán modificadas a través de *properties*. El valor está inicializado en `0`.
- `self.clientes_rechazados`: Un `int` que indica el número de clientes que han sido rechazados de la tienda por tener Progravirus o por que el local esté lleno. Es inicializado en `0`.
- `self.categoria`: Un `str` que indica la categoría del local. Es inicializado en `None`.

- `def utilidades(self)`: Corresponde a la *property* de las utilidades del local. Esta debe encargarse de obtener y modificar el valor de `self.__utilidades`. Además, debe verificar que las utilidades sean mayores o iguales a 0. En caso contrario, el local cierra y se le asigna a `self.__utilidades` el valor de 0.
- `def cliente_ingresa(self, cliente: Cliente): -> bool`: Recibe una instancia de la clase `Cliente` como parámetro. Verifica si se cumplen las condiciones para que el cliente ingrese al local. En caso de que tiene Progravirus, existe la posibilidad de que contagie al trabajador y se cierre el local. Si el cliente cumple las condiciones y entra al local, retorna `True`, en caso contrario retorna `False`. **No debes modificar este método**.
- `def obtener_producto_a_vender(self): -> str`: Selecciona al azar el nombre de uno de los productos que vende el local y lo retorna. **No debes modificar este método**.
- `def entregar_resumen(self)`: Imprime el resumen del estado del local, mostrando el nombre, las utilidades, su estado (abierto o cerrado), el número de clientes dentro y el número de clientes rechazados. **No debes modificar este método**.

Clase Entretenimiento

- `class Entretenimiento`: Esta clase hereda de `Local` y representa aquellos locales de categoría Entretenimiento. Deberás completar la clase tal que herede correctamente de `Local`.
  - `def __init__(self, productos: dict, nombre: str, aforo: int, trabajador: Trabajador)`: Para completar este método deberás enviar los atributos a la superclase. Además deberás definir el siguiente atributo:
    - `self.categoria`: Un `str` con la categoría del local. En este caso: `"Entretenimiento"`.
  - `def cliente_ingresa(self, cliente: Cliente): -> bool`: Para completarlo debes mandar el parámetro cliente al método `cliente_ingresa()` de la superclase. En caso de que el cliente entre al local, deberás llamar el método `self.sanitizar_juegos()` y retornar `True`. En caso contrario, deberás retornar `False`
  - `def sanitizar_juegos(self)`: Se encarga de sanitizar los juegos del local. Una vez listo, imprime la confirmación de que se sanitizaron. **No debes modificarlo**.

Clase Comida

- `class Comida`: Esta clase hereda de `Local` y representa aquellos locales de categoría Comida. Deberás completar la clase tal que herede correctamente de `Local`.

- `def __init__(self, productos: dict, nombre: str, aforo: int, trabajador: Trabajador)`: Para completar este método deberás enviar los atributos a la superclase. Además deberás definir el siguiente atributo:
  - `self.categoría`: Un `str` con la categoría del local. En este caso: `"Comida"`.
- `def cliente_ingresa(self, cliente: Cliente) -> bool`: Para completarlo debes mandar el parámetro cliente al método `cliente_ingresa()` de la superclase. En caso de que el cliente entre al local, deberás llamar el método `self.entregar_menu()` y retornar `True`. En caso contrario, deberás retornar `False`.
- `def entregar_menu(self)`:Se encarga de entregar el menú del local. Una vez entregado, se imprime una confirmación. **No debes modificarlo**.

Clase Tienda

- `class Tienda`: Esta clase hereda de `Local` y representa aquellos locales de categoría Tienda. Deberás completar la clase tal que herede correctamente de `Local`.
  - `def __init__(self, productos: dict, nombre: str, aforo: int, trabajador: Trabajador)`: Para completar este método deberás enviar los atributos a la superclase. Además deberás definir el siguiente atributo:
    - `self.categoría`: Un `str` con la categoría del local. En este caso: `"Tienda"`.
  - `def cliente_ingresa(self, cliente: Cliente): -> bool`: Para completarlo debes mandar el parámetro cliente al método `cliente_ingresa()` de la superclase. En caso de que el cliente entre al local, deberás llamar el método `self.anunciar_oferta()` y retornar `True`. En caso contrario, deberás retornar `False`.
  - `def anunciar_oferta(self)`: Anuncia oferta de un producto aleatorio del local, imprimiendo un mensaje con el producto y el precio. **No debes modificarlo**.

Clase Casino

- `class Casino`: Esta clase hereda de `Entretenimiento` y `Comida` y representa aquellos locales de categoría Casino. Deberás completar la clase tal que herede correctamente de sus superclases.
  - `def __init__(self, productos: dict, nombre: str, aforo: int, trabajador: Trabajador)`: Para completar este método deberás enviar los atributos a la superclase. Además deberás definir el siguiente atributo:
    - `self.categoría`: Un `str` con la categoría del local. En este caso: `"Casino"`.
  - `def cliente_ingresa(self, cliente: Cliente): -> bool`: Realiza la operación de ingresar a un cliente y retorna `True` o `False` dependiendo de si fue aceptado el cliente o no. También sanitiza los juegos, entrega menús y verifica que el cliente sea mayor de edad. **No debes modificarlo.**

Clase Supermercado

- `class Supermercado`: Esta clase hereda de `Comida` y `Tienda` y representa aquellos locales de categoría Supermercado. Deberás completar la clase tal que herede correctamente de sus superclases.

- `def __init__(self, productos: dict, nombre: str, aforo: int, trabajador: Trabajador)`: Para completar este método deberás enviar los atributos a la superclase. Además deberás definir el siguiente atributo:
  - `self.categoría`: Un `str` con la categoría del local. En este caso: `"Supermercado"`.
- `def cliente_ingresa(self, cliente: Cliente): -> bool`: Realiza la operación de ingresar a un cliente y retorna `True` o `False` dependiendo de si fue aceptado el cliente o no. También anuncia oferta y entrega menús. **No debes modificarlo.**

Parte 3: Modelación de Mall

En esta parte, deberás completar la clase `Mall`, la cual se encuentran definida en el archivo `mall.py`. Deberás definirla correctamente en cuanto a sus atributos, *properties* y métodos. Recuerda que al final del enunciado hay un diagrama de clases que te puede ayudar a visualizar las relaciones.

Clase Mall

- `class Mall`: Esta es la clase en donde se realiza la simulación del **DCCostaneraCenter**. Además, el mall contiene a todos los clientes y locales, y es el encargado de dejar constancia de toda acción que se realiza en él. Debes completar los siguientes métodos.
  - `def __init__(self, clientes: list, locales: dict)`: Recibe una lista de los clientes y un diccionario de los locales del mall. A partir de ello, crea los siguientes atributos:
    - `self.clientes`: Una lista de instancias de `Cliente`. **No debes modificarlo**.
    - `self.locales`: Un diccionario cuyas llaves son los nombres de los locales, y valores son las instancias de `Local` (sus clases hijas) correspondientes. **No debes modificarlo**.

    Además, tiene el siguiente atributo que deberás modificar.
    - `self.abierto`: Un booleano que indica si el Mall está abierto o cerrado. Debe inicializarse como `True`.
  - `def utilidades(self)`:: *Property* que debe retornar la suma de las utilidades de todos los locales en el mall.
  - `def pedir_resumen(self)`: Un método que pide el resumen de cada local. Deberás llamar al método `local.entregar_resumen()` de cada local en el diccionario `self.locales`.
  - `def vender(self, local: Local, cliente: Cliente)`: Este método se encarga de vender un producto de `local` a `cliente`. Selecciona un producto al azar del `local` (utilizando el método `obtener_producto_a_vender()` de la clase `Local`) y verifica que su precio sea **menor o igual** al dinero de `cliente`. En este caso (el cliente puede comprar) el `cliente` pierde el dinero gastado en el producto y se suma el mismo monto a las utilidades del `local`. También imprime un mensaje indicando que el `cliente` (indicando su nombre) ha comprado el producto seleccionado (el nombre del producto). Por ejemplo:

```
"María José Hidalgo ha comprado pollo"
```

    En caso contrario, imprime un mensaje que indica que el `cliente` (indicando su nombre) no puede comprar. Por ejemplo:

```
"Joaquín Tagle no puede comprar pollo"
```

- `def iniciar_simulacion(self)`: Este método lleva a cabo la simulación de `DCCostaneraCenter`. La simulación trata de ingresar a cada cliente a su local favorito, solo si se cumplen las condiciones. Una vez adentro del local, se efectúa la venta de productos al cliente. Finalmente, se muestra un resumen de la simulación, mostrando el resumen de cada local y la cantidad de clientes que no pudieron ingresar a su local favorito. **No debes modificar este método.**

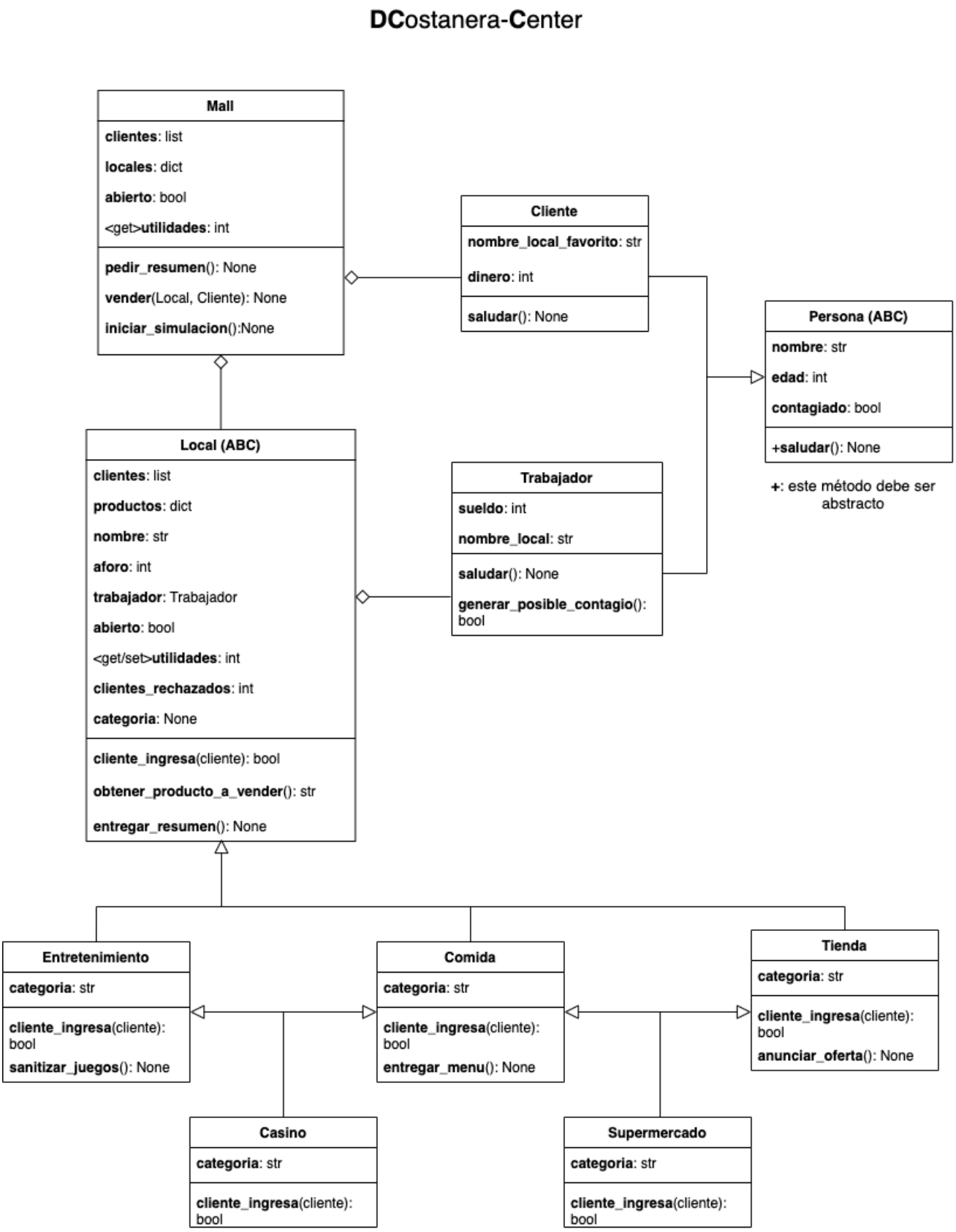
Notas

- La recolección de la actividad se hará en la rama principal (`main`) de tu repositorio.
- Para las *properties* pueden crear nuevos métodos o modificar los existentes si creen que es necesario.
- Si aparece un error inesperado, ¡léelo! Intenta interpretarlo.
- Siéntete libre de agregar nuevos `print` en cualquier lugar de tu código para encontrar errores. Es una herramienta muy útil.

Requerimientos

- (2.00 pts) Modelación de Personas
  - (0.40 pts) Modelar correctamente la Clase Persona
  - (0.80 pts) Modelar correctamente la Clase Cliente
  - (0.80 pts) Modelar correctamente la Clase Trabajador
- (2.00 pts) Modelacion de Locales
  - (0.50 pts) Modelar correctamente la Clase Local
  - (0.30 pts) Modelar correctamente la Clase Entretenimiento
  - (0.30 pts) Modelar correctamente la Clase Comida
  - (0.30 pts) Modelar correctamente la Clase Tienda
  - (0.30 pts) Modelar correctamente la Clase Casino
  - (0.30 pts) Modelar correctamente la Clase Supermercado
- (2.00 pts) Modelación del Mall
  - (2.00 pts) Modelar correctamente la Clase Mall

Anexo


DCostanera-Center
Mall
clientes: list
locales: dict
abierto: bool
<get>utilidades: int
pedir_resumen(): None
vender(Local, Cliente): None
iniciar_simulacion():None
Cliente
nombre_local_favorito: str
dinero: int
saludar(): None
Persona (ABC)
nombre: str
edad: int
contagiado: bool
+saludar(): None
+: este método debe ser abstracto
Local (ABC)
clientes: list
productos: dict
nombre: str
aforo: int
trabajador: Trabajador
abierto: bool
<get/set>utilidades: int
clientes_rechazados: int
categoria: None
cliente_ingresa(cliente): bool
obtener_producto_a_vender(): str
entregar_resumen(): None
Trabajador
sueldo: int
nombre_local: str
saludar(): None
generar_posible_contagio(): bool
Entretenimiento
categoria: str
cliente_ingresa(cliente): bool
sanitizar_juegos(): None
Comida
categoria: str
cliente_ingresa(cliente): bool
entregar_menu(): None
Tienda
categoria: str
cliente_ingresa(cliente): bool
anunciar_oferta(): None
Casino
categoria: str
cliente_ingresa(cliente): bool
Supermercado
categoria: str
cliente_ingresa(cliente): bool