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

IIC2233 - Programación Avanzada
1^{er} semestre 2017

Actividad 07

Decoradores

Instrucciones

La super-aplicación Chantander, una *app* para realizar depósitos, transferencias e inversiones, ha sido atacada por el *MavraVirus* creado por el malvado y explotador Dr. Mavrakis. Este virus hace que se eliminen todas las operaciones de seguridad y verificación de ciertas *apps*.

Los problemas que el virus genera son los siguientes:

- Al hacer una transferencia de una cuenta a otra no se verifica que:
  - Ambas cuentas existan
  - La cuenta de origen tenga el saldo suficiente para poder realizar la operación
  - La clave de la cuenta de origen sea la correcta
- Al ver el saldo total de una cuenta:
  - No verifica que la cuenta exista
  - El saldo se muestra multiplicado por 5.
- Al crear una cuenta no verifica que:
  - El número de cuenta no esté repetido
  - El RUT sea válido
  - La clave tenga 4 dígitos
- Al realizar una inversión no se verifica que
  - La cuenta exista
  - La cuenta tenga el saldo suficiente para realizar la operación
  - La clave de la cuenta sea la correcta
  - El monto total de las inversiones sea menor a 10.000.000.

Los desarrolladores han decidido que para evitar fraudes guardarán todas las operaciones que se realicen en archivos de texto. Todas las operaciones de un mismo tipo deben estar en el mismo archivo ordenadas por fecha y hora.

Como el sistema es muy complejo y no tienes tiempo de revisarlo entero, tomas la decisión de resolver el problema mediante el uso de **decoradores**.

Requerimientos

- (1.00 pts) Implementar correctamente el decorador `verificar_transferencia`. Usando los argumentos de entrada del método, este decorador realizar las verificaciones descritas en el enunciado: existencia de ambas cuentas, saldo suficiente para realizar la operación y la clave de la cuenta de origen. En la verificación levante la excepciones correspondientes indicando la causa del error.
- (1.00 pts) Implementar correctamente el decorador `verificar_inversion`. Usando los argumentos de entrada del método, este decorador realizar las verificaciones descritas en el enunciado para las inversiones: existencia de la cuenta, saldo suficiente para realizar la operación, la clave de la cuenta de origen y que el monto máximo de inversiones, después de realizar la operación, no sea mayor a 10.000.000. Levante las excepciones correspondientes indicando la causa del error.
- (1.50 pts.) Implementar correctamente el decorador `verificar_cuenta`. Este decorador debe verificar que cada vez que se cree una cuenta se utilice un número de cuenta no repetido, la clave tenga 4 números y que el RUT sea válido. Para esto último, basta con verificar que este no tenga más de un guión y que además del guión solo contenga números. **NO** es necesario calcular el dígito verificador. Si el número de cuenta ya existe, deberá crear uno que no exista de forma aleatoria. Levante las excepciones correspondientes indicando la causa del error.
- (1.00 pts.) Implementar correctamente el decorador `verificar_saldo`. Este debe decorar la función `saldo`. Debe verificar que la cuenta exista y que el valor de saldo que se retorna sea el correcto. Levante las excepciones correspondientes indicando la causa del error.
- (1.50 pts.) Implementar correctamente el decorador **log**. Este debe registrar cada operación realizada, como por ejemplo, creación de cuenta, transferencia e inversión, e ir guardándolas en un archivo de texto. El archivo puede ser diferente para cada función, pero el formato de las entradas siempre será `FechaHora - Nombre Acción : args1, args2, ... , kw1_name-kw1_value, kw2_name-kw2_value, ...| output1, output2 ...`. Debe ser posible indicar al decorador el *path* del archivo que se utilizará para guardar las operaciones.

Notas

- Solo puedes modificar el código original para agregar tus decoradores. Cualquier otro cambio en la estructura del sistema significará un 1.0 inmediato.
- Recuerda que puedes crear funciones auxiliares para verificación.
- Para levantar las excepciones correspondiente a la verificación puedes usar el tipo `AssertError()`.

Entrega

- **Lugar:** GIT - Carpeta: Actividades/AC07
- **Hora:** 16:55

Output

```
Cuenta creada correctamente
Cuenta creada correctamente
Cuenta creada correctamente
Rut invalido, tiene mas de un guion

Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 245000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 13000

Clave valida, revisando saldo...
Saldo suficiente, efectuando transaccion...
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 240000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000

Cuenta destino no existe
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 240000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000

Clave valida, revisando saldo...
Saldo insuficiente
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 240000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000

Clave invalida
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 240000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000

Clave valida, revisando saldo...
Saldo insuficiente
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 240000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000

Clave invalida
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 240000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000

Clave valida, revisando saldo...
Saldo suficiente, efectuando operacion...
Nombre: Mavrakis - Numero de cuenta: 1 - Saldo: 0
Nombre: Ignacio - Numero de cuenta: 2 - Saldo: 220000
Nombre: Diego - Numero de cuenta: 3 - Saldo: 18000
```