Escuela de Arquitectura y Tecnología Grado en Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS II

Examen parcial 1

Curso:	Segundo	**Curso académico:**	2022/2023
Fecha examen:	16/03/23	**Duración examen:**	100 minutos
Convocatoria:	Primera	**Material permitido:**	TODO menos RTC.
Instrucciones:	Crear un proyecto llamado nombreApellidoParcial1. Crea un paquete con vuestro nombre y un subpaquete separado para cada pregunta. Para entregarlo exportar el proyecto desde el IDE como un .zip y subirlo a la PDU. Se deben seguir los principios de POO en todo momento (encapsulación, herencia, polimorfismo) y minimizar la duplicidad de código.		

1. **(5 puntos)** Las cuentas bancarias se identifican mediante un código numérico de 20 cifras. Los cuatro primeros dígitos corresponden a la entidad bancaria, los siguientes cuatro dígitos corresponden a la oficina, seguidos de dos dígitos de control y por último 10 dígitos del número de cuenta.
   1.1. **(1 punto)** Quiero que crees la clase CuentaBancaria, con el/los atributos y métodos que tu consideres necesarios para albergar el número de cuenta y el saldo disponible. Quiero poder acceder a las partes que conforman vuestro número de cuenta (código de entidad, código de oficina, dígitos de control y numero de cuenta) de forma homogénea (con getters que devuelvan Strings), sin depender de los detalles de vuestra implementación interna (que no debe ser pública). Además, quiero que al imprimir un objeto de tipo cuenta, se impriman los dígitos de la cuenta como en la imagen.
   1.2. **(1 punto)** Quiero un constructor que reciba como parámetro de entrada dos Strings, uno representa a la entidad bancaria y otro representa la oficina. Para obtener los dígitos correspondientes a la entidad bancaria debes sumar el valor ASCII de cada letra, si el número resultante es mayor a 9999 haz el módulo. Para obtener los dígitos correspondientes a la oficina debes realizar la misma operación con el String de oficina, pero esta vez las letras mayúsculas valen dobles. Una vez trasladados los Strings a cifras, no volveremos a necesitar el String. EL número de cuenta se generará de forma automática al crear las cuentas, asignando un valor consecutivo para cada cuenta de cada banco. La primera cuenta creada de cada banco recibirá el 1, la segunda el 2, etc. Habrá una cuenta con el número 1 por cada banco.
   1.3. **(1 punto)** Quiero un método estático llamado calcularDigitosControl que recibe como parámetro de entrada los dígitos del banco, la oficina y la cuenta (de la manera que tu consideres) y devuelve el valor de los 2 dígitos de control (como tú prefieras). Para calcular los dígitos de control, se debe multiplicar el código de banco por el código de entidad, al resultado se le suma el número de cuenta y por último se realiza el módulo 99 de ese resultado. Llama a este método como parte de tu constructor.
   1.4. **(1 punto)** Quiero una función estática cerrarOficina, que reciba como parámetro un array de cuentas bancarias, un String que representa el banco cuya oficina va a cerrar, un String que representa la oficina que va a cerrar y un String que representa la nueva oficina de destino. Migra todas las cuentas de la oficina que cierra a la oficina de destino, cambiando el código de oficina y recalculando el código de control.
   1.5. **(1 punto)** Crea una serie de tests que comprueben el buen funcionamiento de tu clase CuentaBancaria. Testea que calculas adecuadamente los códigos de banco y oficina, los códigos de control y la migración de cuentas. Puedes utilizar los datos de mi imagen. También quiero que documentes mediante javadoc las funciones principales de esta clase, para que pueda usarlas cualquiera.

2. **(5 puntos)** Quiero cuentas bancarias de diferentes tipos en función del tipo de operaciones que realizan los clientes a los que van dirigidas. Para simular el comportamiento de los clientes, cada cuenta tendrá un método **nextMovement()** que devolverá el siguiente movimiento que realizaría el cliente. Además, cada cuenta tiene un número de movimientos que realiza al mes, que se fija al construirse. Todas las cuentas tienen además unos gastos fijos de (5-10€) al mes y una penalización por estar en números rojos del 1% del saldo debido.
   2.1. **(1 punto) Cuenta de ahorro**, el cliente realiza entre 10 y 15 movimientos al mes, generalmente (80% de las veces) hace ingresos de dinero pequeños (50-100€) y el resto de las veces saca cantidades de dinero moderadas (100-200€).
      - Las cuentas de ahorro otorgan un interés cada mes, cada cuenta tiene un valor de interés fijado al crear la cuenta (1-3%)

   2.2. **(1 punto) Cuenta de crédito**, el cliente realiza entre 30 y 45 movimientos al mes, generalmente (85% de las veces) realiza gastos grandes (500-1000€), algunas veces (14%) realiza gastos moderados (200-400€) y muy raramente (1%) hace un ingreso enorme (50.000-100.000€).
      - Las cuentas de crédito tienen una demora del pago de descubiertos, los dos primeros meses que están en números rojos no pagan ninguna penalización (a partir del tercer mes sí). Si vuelven a estar en positivo, volverán a disfrutar de esos dos meses de demora.

   2.3. **(1 punto) Cuenta de hipoteca**, el cliente tiene un patrón de operaciones muy determinado y siempre realiza las mismas operaciones (no dependen del azar). Primero hace un ingreso de la nómina (1000€-2000€), después paga la hipoteca (500€-1000€) y por último paga 3 facturas de consumos (20-100€).
      - Las cuentas de hipoteca reciben un interés cada mes, como las de ahorro (El valor de interés, sin embargo, es solo del 0,5%). Además, también tienen una demora del pago de descubiertos, como las cuentas de crédito, pero solo de 1 mes.

   2.4. **(1 punto)** Todas las cuentas deben tener un método **simulaUnMes()** que se encargue de simular los movimientos que el cliente hace en un mes, imprimiendo un mensaje por pantalla que corresponda a cada operación realizada. Al final de cada mes, se cobran los gastos fijos, se calculan los intereses a pagar a los clientes y por ultimo se calculan las penalizaciones por números rojos.
   2.5. **(1 punto)** Crea una clase **Simulación**, que contiene un array de 10 Cuentas, llénalo con 10 cuentas, garantizando que hay al menos una cuenta de cada tipo, y realiza una simulación de 100 meses. Al finalizar el mes, **ordena** las cuentas en función de su saldo (descendente, la que más dinero tiene la primera) e imprímelas por pantalla.

```
CuentaBancaria[] cuentas = new CuentaBancaria[6];

cuentas[0] = new CuentaBancaria("", "");
cuentas[1] = new CuentaBancaria("", "");
cuentas[2] = new CuentaBancaria("a", "a");
cuentas[3] = new CuentaBancaria("A", "A");
cuentas[4] = new CuentaBancaria("BBVA", "A");
cuentas[5] = new CuentaBancaria("BBVA","Centro");

for(CuentaBancaria cb : cuentas) {
    System.out.println(cb);
}
```

```
0000-0000-01-0000000001  - Saldo: 0.0
0000-0000-02-0000000002  - Saldo: 0.0
0097-0097-05-0000000001  - Saldo: 0.0
0065-0130-36-0000000001  - Saldo: 0.0
0283-0130-62-0000000001  - Saldo: 0.0
0283-0686-01-0000000002  - Saldo: 0.0
```

```
System.out.println("El BBVA cierra su oficina A y mueve sus cuentas a la oficina Centro");
CuentaBancaria.cerrarOficina(cuentas, "BBVA", "A", "Centro");
```

```
El BBVA cierra su oficina A y mueve sus cuentas a la oficina Centro
0000-0000-01-0000000001  - Saldo: 0.0
0000-0000-02-0000000002  - Saldo: 0.0
0097-0097-05-0000000001  - Saldo: 0.0
0065-0130-36-0000000001  - Saldo: 0.0
0283-0686-00-0000000001  - Saldo: 0.0
0283-0686-01-0000000002  - Saldo: 0.0
```