


Evaluación Ordinaria : Trimestres 1 y 2 (Tiempo: 2h30' para Trimestre 2 y 2h para trimestre 1	25 de Mayo de 2020 - 16:00h
Nombre y Apellidos	DNI/NIE: Firma:
1º Desarrollo de Aplicaciones Web (Vespertino) Módulo: Programación	 IES Alonso de Avellaneda (Alcalá de Henares)

Se realizará la Parte del Trimestre 2 en primer lugar. Al finalizar se hará un descanso de 10 min para luego continuar con la Parte del Trimestre 1.

Es necesario obtener una calificación igual o mayor que 5 en ambos trimestres para poder superar el módulo. En caso contrario, se iría a la prueba final extraordinaria

Los alumnos que tengan pendiente sólo un trimestre deberán obtener una nota igual o mayor que 5 para poder superar el módulo, en caso contrario, deberán presentarse a la prueba final extraordinaria.

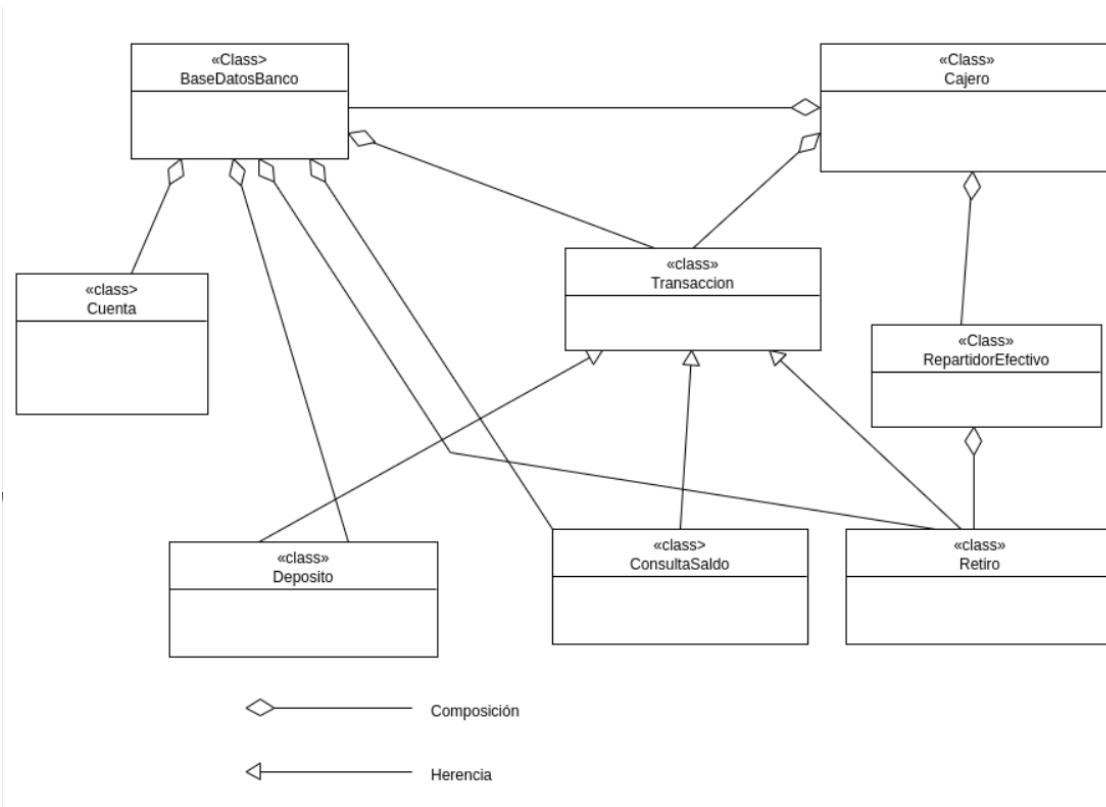
No se permite la copia de recursos de Internet o de otras pruebas, resultando no válido el ejercicio.

Trimestre 2

Ejercicio 1: (6p) La red de Cajeros RedTar necesita un programa para que los usuarios puedan retirar el dinero y hacer depósitos según las siguientes especificaciones:

- Los usuarios deben hacer login (número cuenta) con usuario y password (pin) y el programa debe verificar tanto el número de cuenta como el pin. Si no fuera alguno de los anteriores válidos volvería a la pantalla de login. El programa debe crear un par de usuarios de prueba con un identificador numérico para el usuario y otro para el pin. Además de un balance disponible (saldo para retirar del cajero) y un balance total (balance disponible más los depósitos que se hayan realizado a la espera de que sean aceptados).
- Una vez que el usuario entra en la aplicación le sale un menú con las opciones: 1 *Ver mi balance*, 2 *Retirada efectivo*, 3 *Depositar efectivo*, 4 *Salir* . Cada una de estas opciones realiza lo siguiente:
- Opción 1: visualiza el balance disponible y el total. Es decir la cantidad disponible para sacar o retirar y la cantidad anterior más los depósitos que se hayan realizado.
- Opción 2: El cajero tiene billetes de 20, por ejemplo, si tiene 10 billetes de 20€ podrá retirar hasta 200€. Puede retirar las cantidades de 20, 40, 60, 100 y 200 € y debe verificar que tenga disponible efectivo el cajero para poder dárselos al cliente, en caso contrario deberá de cancelar la operación. También se puede cancelar la transacción si el usuario lo desea y volver al menú principal.
- Opción 3: Depositar efectivo de tal forma que se puede ingresar una cantidad entera de euros. Se supone que el ingreso lo hace mediante un sobre donde debe haber una cantidad entera. Después el programa pregunta si ha introducido el sobre con la cantidad. Si ha sido así, muestra un mensaje de que se comprobará el efectivo y sumará la cantidad al balance total. Si no ha sido así, se cancela la operación y vuelve al menú principal. Una vez depositado el dinero se suma al balance total. Se debe suponer que cuando el usuario ha

hecho un depósito se tiene que verificar que el dinero está correcto, esto último se supone y no es necesario implementarlo.



Explicación de las clases:

- **BasesDatosBanco:** (0.5p): es la base de datos bancaria que tiene las Cuentas de usuario.
 - Atributos: listado de cuentas.
 - El constructor crea un par de cuentas de ejemplo con los siguientes datos:
 - Número Cuenta, Pin, balance disponible, balance total: 11444, 1111, 1000.0, 1200.0
 - Número Cuenta, Pin, balance disponible, balance total: 22333, 2222, 200.0, 200.0
 - `autenticarUsuario()`: valida el usuario y el pin.
 - `credito()`: se suma la cantidad a la cuenta del usuario. Utiliza la cuenta y el usuario.
 - `debito()`: se resta la cantidad a la cuenta del usuario. Utiliza la cuenta y el usuario
 - `getBalanceDisponible()`: retorna el balance disponible.
 - `getBalanceTotal()`: retorna el balance total.
 - `getCuenta()`: retorna la cuenta según el número de cuenta recibido.
- **Cajero:** imprime el menú principal, autentica los usuarios y realiza las transacciones. (1p)
 - Atributos: `numCuentaCorriente`, `basesdatosBanco` (para la base de datos de bancos referencia a la clase)
 - `autenticarUsuario()`: recoge la información para verificar que el usuario y el pin están almacenados mediante la clase `BasesDatosBanco`.
 - `crearTransacción()`: crea la transacción si es una retirada, un depósito o una consulta.
 - `realizarTransacción()`: ejecuta la transacción actual.
 - `mostrarMenu()`: muestra menú principal.
- **ConsultaSaldo** (0.25p)
 - `ejecutar()`: ejecuta la transacción y muestra balance disponible y balance total con dos decimales.
- **Cuenta:** (1p)

- Atributos: numCuenta, pin, balanceDisponible, balanceTotal.
 - credito(): suma al balance total la cantidad que se deposite:
 - debito(): resta del balance total y disponible la cantidad
 - getBalanceDisponible(): retorna el balance disponible.
 - getBalanceTotal(): retorna el balance total.
- getNumCuenta()
 - validarPin(): valida el pin
- **Deposito:** (0.75)
 - atributos: cantidad, depositoRecibido.
- ejecutar(): si el usuario ha introducido la cantidad debe preguntar si lo ha hecho o no y el usuario debe contestar con Si ó No. En caso afirmativo debe generar un mensaje de que se ha realizado el depósito y que se verificará la cantidad y ahora el balance total se incrementará en lo depositado. En caso negativo, se cancela la transacción y vuelve al menú principal
 - imprimirParaCantidadDeposito(): mensaje que imprime el mensaje de cantidad a depositar y recibe la cantidad.
- **RepartidorEfectivo:** (1p)
 - Atributos: cuentaInicial (número billetes de 20 que suponemos por ejemplo que tenga 10).
 - darEfectivo(): suponer que trabaja con billetes de 20€ y que tiene una cantidad inicial dada en esos billetes. Al dar el efectivo se dará de 20 en 20.
 - hayEfectivo(): verifica si dada una cantidad se dispone de suficiente efectivo. Esto sería si, por ejemplo, se tuviera una cantidad inicial por ejemplo de 100€ en billetes de 20 y se quiere sacar 200€, no habrían suficiente efectivo y el sistema le indicaría que eligiera otra cantidad menor.
- **Retiro** (1p)
 - Atributos: cantidad, repartidorEfectivo (referencia a objeto de la clase RepartidorEfectivo)
 - ejecutar(): este método deberá controlar que hay suficiente balance disponible y efectivo en el cajero para retirar la cantidad. Se debe de dar la opción de cancelar el retiro. Si no hay efectivo y/o no hay saldo disponible, se debe de imprimir un mensaje de que no hay suficiente efectivo y/o fondos respectivamente.
 - mostrarMenuCantidades(): se deben mostrar las cantidades que se pueden retirar: 20, 40, 60, 100, 200 y también se podrá cancelar dicha opción.
- **Transaccion:** clase abstracta (0.5p)
 - Atributos: bddBanco (referencia a la BaseDatosBanco), numCuenta
 - getNumCuenta(): retorna el número de cuenta
 - getBddBanco(): retorna la lista de los bancos.
 - ejecutar(): método abstracto que será implementado donde corresponda

Aparte de los métodos de arriba, se pueden implementar otros que crea necesario. También se exponen algunos atributos importantes y que puede complementar con otros que necesite. Los constructores que se vayan a utilizar deben realizar las acciones de inicio de los atributos además de otras funciones que considere necesarias. No se solicita que se hagan todos los getters y setters. **Se debe seguir la implementación dada por el diagrama.**

Muestra de uso:

_____RedTar de Cajeros_____

|
|

```
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
|_____Login_____|
|   Introduce la información de usuario   |
```

Introduce tu número de cuenta:
1144

Introduce el PIN:
11

Número de cuenta o PIN inválida. Por favor, inténtalo otra vez
_____RedTar de Cajeros_____

```
|_____|
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
|_____Login_____|
|   Introduce la información de usuario   |
```

Introduce tu número de cuenta:
11444

Introduce el PIN:
1111

Menú principal:
1 - Ver mi balance
2 - Retirada efectivo
3 - Depositar efectivo
4 - Salir
Introduzca una opción:
1

Información de balance:
- Balance disponible
\$1.000,00
- Total balance:
\$1.200,00

Menú principal:
1 - Ver mi balance
2 - Retirada efectivo
3 - Depositar efectivo
4 - Salir
Introduzca una opción:
2

Menú de retiro:
1 - 20€
2 - 40€
3 - 60€
4 - 100€
5 - 200€
6 - Cancelar transacción
2

Su efectivo ha sido dispensado. Puede recoger su efectivo.
Menú principal:
1 - Ver mi balance
2 - Retirada efectivo
3 - Depositar efectivo

4 - Salir

Introduzca una opción:

1

Información de balance:

- Balance disponible

\$960,00

- Total balance:

\$1.160,00

Menú principal:

1 - Ver mi balance

2 - Retirada efectivo

3 - Depositar efectivo

4 - Salir

Introduzca una opción:

3

Por favor, introduzca una cantidad a depositar en € (o pulse 0 para cancelar):

100

Por favor introduzca la cantidad a depositar en un sobre: 100,00.Introducido sobre? S / N:

N

No insertó el sobre por lo que la operación ha sido cancelada por el cajero.

Menú principal:

1 - Ver mi balance

2 - Retirada efectivo

3 - Depositar efectivo

4 - Salir

Introduzca una opción:

3

Por favor, introduzca una cantidad a depositar en € (o pulse 0 para cancelar):

100

Por favor introduzca la cantidad a depositar en un sobre: 100,00.Introducido sobre? S / N:

S

Su sobre ha sido recibido.

NOTA: El dinero sólo ha sido depositado estará disponible cuando se verifique la cantidad del sobre adjunto.

Menú principal:

1 - Ver mi balance

2 - Retirada efectivo

3 - Depositar efectivo

4 - Salir

Introduzca una opción:

1

Información de balance:

- Balance disponible

\$960,00

- Total balance:

\$1.260,00

Menú principal:

1 - Ver mi balance

2 - Retirada efectivo
3 - Depositar efectivo
4 - Salir
Introduzca una opción:
4

Saliendo del sistema

Gracias por usar el Cajero

Introduzca una opción:
2
Cuando hay fondos insuficientes
Menú de retiro:
1 - 20€
2 - 40€
3 - 60€
4 - 100€
5 - 200€
6 - Cancelar transacción
5

Fondos Insuficientes en su cuenta.

Por favor seleccione una cantidad menor.
Cuando no hay suficiente efectivo
Introduzca una opción:
1

Información de balance:
- Balance disponible
\$200,00
- Total balance:
\$400,00
Menú principal:
1 - Ver mi balance
2 - Retirada efectivo
3 - Depositar efectivo
4 - Salir
Introduzca una opción:
2

Menú de retiro:
1 - 20€
2 - 40€
3 - 60€
4 - 100€
5 - 200€
6 - Cancelar transacción
4

Efectivo insuficiente en Cajero

Por favor, seleccione una cantidad menor.

Dada una matriz **n x n** de tipo String, crea un programa que procese la matriz de forma que, dada una matriz de entrada, resulten las siguientes matrices:

- función: **matrizAleatoria()**: calcula la matriz donde todos sus elementos se ordenan de forma aleatoria mediante el primer carácter de cada elemento de forma alfabética. (1.25p)
- función: **matrizOrdenadaPorFilas()**: calcula la matriz donde sus filas están ordenadas por la primera inicial de la cadena de manera alfabética.(1.25p)
- función: **matrizTriangularAbajo()**: calcula la matriz triangular inferior rellenando con asteriscos por encima. (0.75)
- función: **matrizTriangularArriba()**: calcula la matriz triangular superior rellenando con asteriscos por debajo. (0.75)

```
*****Matriz normal: *****

ab fg sT DD aa
zz hi ba AE Mn
gK AC BA AF AA
CC lT FA GH DA
ZM sF oP mK am
*****Matriz Aleatoria: *****

ab DD fg sT aa
zz hi Mn ba AE
AA gK AF BA AC
FA lT GH DA CC
oP sF am ZM mK
*****Matriz Ordenada Por filas: *****

aa ab DD fg sT
AE ba hi Mn zz
AA AC AF BA gK
CC DA FA GH lT
am mK oP sF ZM
*****Matriz Diagonal Abajo: *****

aa * * * *
AE ba * * *
AA AC AF * *
CC DA FA GH *
am mK oP sF ZM
*****Matriz Diagonal Arriba: *****

aa ab DD fg sT
* ba hi Mn zz
* * AF BA gK
* * * GH lT
* * * * ZM
```

Se necesita introducir una matriz de cadenas y que el programa pueda imprimir la matriz aleatoria, ordenada por filas, ordenada por columnas, la matriz triangular inferior y superior.

Nombre de la clase: **Ejercicio2Trim2.java**

Rúbrica:

Ejercicio1: 100% especificación, implementación y funcionamiento correcto de cada clase. En caso contrario de estar incompleta en su especificación, implementación y no funcione, la valoración será de 0%. Si no funciona pero su especificación e implementación son correctas hasta un 50%.

Ejercicio2: 100% especificación, implementación y funcionamiento correcto de cada método. En caso contrario de estar incompletos en su especificación, implementación y no funcione, la valoración será de 0%. Si no funciona pero su especificación e implementación son correctas hasta un 50%.