

ciclo: DAW

MÓDULO DE programación

Tarea Nº 7

Alumno:

Borja Antonio Suárez Rodríguez

45768505T

*Los documentos, elementos gráficos, vídeos, transparencias y otros recursos didácticos incluidos en este contenido pueden contener imprecisiones técnicas o errores tipográficos. Periódicamente se realizan cambios en el contenido. Fomento Ocupacional FOC SL puede realizar en cualquier momento, sin previo aviso, mejoras y/o cambios en el contenido.*

*Es responsabilidad del usuario el cumplimiento de todas las leyes de derechos de autor aplicables. Ningún elemento de este contenido (documentos, elementos gráficos, vídeos, transparencias y otros recursos didácticos asociados), ni parte de este contenido puede ser reproducida, almacenada o introducida en un sistema de recuperación, ni transmitida de ninguna forma ni por ningún medio (ya sea electrónico, mecánico, por fotocopia, grabación o de otra manera), ni con ningún propósito, sin la previa autorización por escrito de Fomento Ocupacional FOC SL.*

*Este contenido está protegido por la ley de propiedad intelectual e industrial. Pertenecen a Fomento Ocupacional FOC SL los derechos de autor y los demás derechos de propiedad intelectual e industrial sobre este contenido.*

*Sin perjuicio de los casos en que la ley aplicable prohíbe la exclusión de la responsabilidad por daños, Fomento Ocupacional FOC SL no se responsabiliza en ningún caso de daños indirectos, sean cuales fueren* *su naturaleza u origen, que se deriven o de otro modo estén relacionados con el uso de este contenido.*

*© 2020 Fomento Ocupacional FOC SL todos los derechos reservados.*

Contenido

[1. Documentos que se adjuntan a este informe. 2](#_Toc52179159)

[2. Resto de epígrafes que componen el desarrollo de este informe. 2](#_Toc52179160)

(Una vez realizado el informe, no olvidar actualizar esta tabla del índice **(F9 + Actualizar toda la tabla)**, con el fin de que se actualicen todos los epígrafes y números de página)

1. Documentos que se adjuntan a este informe.

A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Imágenes.
3. Resto de epígrafes que componen el desarrollo de este informe.

En esta tarea se van a realizar un conjunto de clases que permitan modelar las distintos tipos de cuentas que una persona puede tener en un banco, haciendo uso del mecanismo de herencia. Los puntos que se deben tratar para resolver la tarea son:

RA07\_g) Se han realizado programas que implementen y utilicen jerarquías de clases.

Crear un proyecto en NetBeans denominado, "CuentasBancarias".

Dentro de dicho proyecto, crear un paquete denominado "modeloBancario".

Dentro del paquete "modeloBancario", crear una clase denominada Cliente, que modele los distintos clientes del banco que tienen una cuenta asociada para almacenar su dinero. Las características de la clase Cliente son:

Atributos (Todos los atributos de la clase Cliente deben tener visibilidad privada):

**idCliente**: número entero que representa el identificador único del cliente dentro del banco.

**nombre**: cadena de caracteres que representa el nombre del cliente del banco.

**direccion**: cadena de caracteres que representa la dirección donde vive el cliente del banco.

**teléfono**: cadena de caracteres que representa el teléfono que permite contactar con el cliente.

**Métodos** (Todos los métodos de la clase Cliente deben tener visibilidad pública):

constructor sin parámetros: constructor que inicializa todos los atributos de tipo cadenas de caracteres al valor null y los números enteros a 0.

constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

RA07\_b) Se han utilizado modificadores para bloquear y forzar la herencia de clases y métodos.

Dentro del paquete "modeloBancario", crear una clase abstracta denominada "Cuenta", que modele las distintas cuentas bancarias que mantiene la entidad financiera donde los clientes depositan su dinero.

Atributos (Todos los atributos de la clase Cuenta deben tener visibilidad protegida)

numeroDeCuenta: número entero que representa el identificador único asociado a cada una de las cuentas del banco.

saldo: número real que representa la cantidad de dinero almacenado en dicha cuenta.

titular: atributo de tipo Cliente que representa la persona que está asociada a dicha cuenta.

Métodos (Todos los métodos de la clase Cuenta deben tener visibilidad pública)

constructor sin parámetros: constructor que inicializa el cliente titular de la cuenta a null, y el saldo y el número de cuenta a cero.

constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes.

**getNumeroDeCuenta**: método que permite obtener el número de cuenta.

**getSaldo**: método que permite obtener el saldo de la cuenta.

**getTitular**: método que permite obtener el titular de la cuenta

**setNumeroDeCuenta**: método que recibe un parámetro entero que representa el nuevo número de cuenta que se desea asignar y asigna parámetro el valor de dicho al atributo numeroDeCuenta.

**setSaldo**: método que recibe un parámetro entero que representa el nuevo saldo que se desea asignar y asigna parámetro el valor de dicho al atributo saldo.

**setTitular**: método que recibe un parámetro Cliente que representa el nuevo titular que se desea asignar y asigna parámetro el valor de dicho al atributo titular.

**ingresar**: recibe un parámetro entero que representa la cantidad que se desea ingresar en la cuenta. El método incrementará el saldo en la cantidad recibida como parámetro.

**retirar**: método abstracto que permitirá sacar una cantidad de la cuenta (si hay saldo disponible para ello), no se implementará ya que dependerá del tipo de cuenta, por tanto su implementación recaerá en las clases hijas.

**actualizarSaldo**: método abstracto que actualizará el saldo de la cuenta, dependiendo del tipo de interés de cada una de las cuenta, por tanto su implementación recaerá en las clases hijas.

RA07\_c) Se ha reconocido la incidencia de los constructores en la herencia.

Tanto las clases padre (superclases) con las hijas (subclases), deben de tener implementados los constructores.

RA07\_d) Se han creado clases heredadas que sobrescriban la implementación de métodos de la superclase.

Dentro del paquete "modeloBancario", crear una clase denominada "CuentaCorriente", que herede de la clase “Cuenta”, que modele un tipo de cuenta con un interés fijo del 1.5%.

Atributos (Todos los atributos de la clase CuentaCorriente deben tener visibilidad protegida):

interesFijo: constante real cuyo valor es 0.15.

Métodos (Todos los métodos de la clase CuentaCorriente deben tener visibilidad pública)

constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes. Dicho constructor debe hacer uso del constructor de la clase padre

“Cuenta”.

Implementación de los métodos abstractos retirar y actualizarSaldo.

Dentro del paquete "modeloBancario", crear una clase denominada "CuentaAhorro", que herede de la clase “Cuenta”, que modele un tipo de cuenta con un interés variable y un saldo mínimo necesario.

Atributos (Todos los atributos de la clase CuentaAhorro deben tener visibilidad protegida).

interesVariable: número real que representa el tipo de interés que se aplica a la cuenta.

saldoMinimo: número real que representa el dinero mínimo que debe haber en la cuenta.

Por tanto, el valor del atributo saldo siempre debe ser mayor o igual que saldoMinimo

Métodos (Todos los métodos de la clase CuentaAhorro deben tener visibilidad pública)

constructor con parámetros: constructor que tienen tantos parámetros como atributos tiene la clase, y que inicializa cada uno de los atributos con el valor de los parámetros correspondientes. Dicho constructor debe hacer uso del constructor de la clase padre “Cuenta”.

Implementación de los métodos abstractos retirar y actualizarSaldo. Esta cuenta tiene como atributos el interés variable a lo largo del año y un saldo mínimo necesario. Al retirar dinero el saldo debe ser mayor o igual que el saldoMinimo.

RA07\_e) Se han diseñado y aplicado jerarquías de clases.

Crear un entorno que importe el paquete "modeloBancario" y que permita probar las distintas clases Cuentas implementadas.

RA07\_f) Se han probado y depurado las jerarquías de clases.

En el informe se adjuntarán capturas que demuestren el correcto funcionamiento de la aplicación.

RA07\_h) Se ha comentado y documentado el código.

Realizar los comentarios oportunos en el código, que aclaren la funcionalidad de los métodos desarrollados.

Contestar a nivel teórico las siguientes cuestiones:

RA07\_a) Identificar los conceptos de herencia, superclase y subclase.

Explicar que son los conceptos enumerados.

**Herencia:**

**Podemos definir la herencia como la capacidad de crear clases que adquieren de manera automática los miembros (atributos y métodos) de otras clases que ya existen, pudiendo al mismo tiempo añadir atributos y métodos propios.**

**Superclase:**

**En el lenguaje de Java, una clase que se hereda a otras se denomina superclase, sería lo que llamamos clase padre.**

**Subclase:**

**La clase que hereda se llama subclase o clase hija. Por lo tanto, una subclase es una versión especializada de una superclase. Hereda todas las variables y métodos definidos por la superclase y agrega sus propios elementos únicos.**