|  |  |
| --- | --- |
|  | **Proyecto**  **Programación POO**  **Trabajo para 2 estudiantes.**  **Entidad Bancaria** |

El Proyecto simulará el manejo de una pequeña entidad bancaria: sus transacciones, los usuarios del sistema: cajeros y administradores y clientes.

Administrador del sistema, crear y manejar el sistema de información

Cajeros – Realizar transacciones bancarias de los clientes – Consignar y retirar

Cliente – realizar transacciones de consultar el saldo de la empresa, consignar retirar y transferir

Todos los usuarios tienen: nombre, dirección, teléfono, correo, cedula como identificador,

Los usuarios Administradores deben tener una etiqueta especial que los identifique como administradores.

Los cajeros deben tener sueldo y un código especial que será asignado a cada transacción bancaria

Los clientes pueden tener además cuentas corrientes y cuentas de ahorros.

La arquitectura del sistema deberá ser implementada bajo el patrón Modelo Vista Controlador (MVC).

Diagrama

Descripción generada automáticamente

Los administradores del sistema deben crear todas las tablas del sistema con las operaciones CRUD correspondientes: CRUD.

Create = Crear e insertar datos en las estructuras dinámicas en memoria.

Read = Leer o consultar datos de las estructuras.

Update = Actualizar o modificar datos de las estructuras.

Delete = Borrar datos de las estructuras.

Las operaciones CRUD deberán estar definidas a través de una interfaz que heredan todas las clases.

Insertar, borrar, actualizar, ordenar, mostrar todos y mostrar un solo dato debe ser implementado por el usuario, administrador.

Crear Usuarios, Crear cuentas corrientes, Crear cuentas de ahorros, crear áreas de la empresa y ejecutar todas las operaciones CRUD es función del usuario Administrador

Los cajeros pueden crear cuentas y realizar transacciones de los clientes sobre las cuentas y consultar datos de los clientes.

Los clientes no pueden crear cuentas solo realizar transacciones sobre las cuentas. Consignar, retirar y transferir.

Una cuenta de ahorros tiene: id del usuario, id de la cuenta, tipo de cuenta, y saldo.

Deben quedar registradas las transacciones que realizan los usuarios.

Un cliente tiene cuentas corrientes y cuentas de ahorros

Se podrán realizar los siguientes ordenamientos:

Los ordenamientos por id de los clientes se realizarán con el algoritmo en java de quicsort.

Los ordenamientos por nombre de los clientes se realizarán con el algoritmo en java de mersort.

Los ordenamientos de las cuentas se realizarán con el método sort de las colecciones.

Todas las búsquedas de un objeto se realizarán usando búsqueda binaria.

Todos los usuarios del sistema podrán consultar los saldos.

Retirar implica restar.

Consignar implica sumar.

Transferir restar de una cuenta y sumar a otra. Requiere cuenta origen y cuenta destino.

Se deberán usar estructuras de almacenamiento físico en disco: Bases de Datos o Archivos.

**Primera entrega – Proyecto 1**:

* Se deberán implementa todas las operaciones CRUD de todas las clases identificadas en el proceso. Insertar, Buscar por id usando búsqueda binaria en todas las clases identificadas.
* Se crearán constructores para instanciar los objetos identificados y los getter y los setter
* Se usará interfaz modo texto
* Se crearán los menús en modo texto usando switch case.
* Se verificarán que los datos no están repetidos usando el método equals().
* Se imprimirán usando el método toString().

**Segunda entrega – Proyecto 2**

* Implementar todas las operaciones CRUD usando interfaz gráfica
* Se realizarán los demás ordenamientos que requiere el problema.
* Se realizarán las validaciones de campos pertinentes.
* Se realizarán los menús
* Se crearán todas las consultas y transacciones del banco solicitadas en el sistema.
* Crear el diagrama de clases del sistema usando una herramienta que maneje UML.

**Tercera Entrega – Proyecto Final**

* Deberán crear una pantalla de inicio que permita autenticación por usuarios, se manejarán los perfiles definidos en el problema
* Almacenar los datos y realizar las operaciones CRUD en medios de almacenamiento físico.
* Crear las excepciones usando expresiones regulares de:
  + Correo electrónico
    - Validar la estructura del correo
  + Fechas
    - Validar la estructura de una fecha
  + contraseñas de usuarios:
    - Deben existir en el medio de almacenamiento y tener comenzar con 1 letra mayúscula, deben tener por lo menos 1 dígito y un carácter especial y una longitud mímica de 8 caracteres.
  + Nombres de usuarios: El nombre del usuario no puede tener menos ni más de 8 caracteres no debe incluir caracteres especiales y deberá ser validado a través de una expresión regular, usando una excepción.
  + Direcciones de los clientes deben ser validadas calle, carrera, y la estructura.
  + El menú principal del sistema que le cargue al administrador deberá tener una opción adicional que permita crear los demás usuarios.

**Funcionalidad**

* 1. El programa deberá ser lanzado desde un icono del escritorio sin usar el ambiente de desarrollo, NetBeans o BlueJ.
  2. El programa no deberá cerrarse cuando cargue las pantallas, estas deben retornar al menú principal.
  3. El programa deberá manejar todas las excepciones sin terminar abruptamente.
  4. El sistema manejará los conceptos de herencia, polimorfismo, relaciones entre clases, sobrecarga de métodos, constructores y encapsulamiento

**Apoyos para expresiones regulares**

[**http://web.ontuts.com/snippets/10-expresiones-regulares-imprescindibles-en-desarrollo-web/**](http://web.ontuts.com/snippets/10-expresiones-regulares-imprescindibles-en-desarrollo-web/)

[**http://puntocomnoesunlenguaje.blogspot.com.co/2013/07/ejemplos-expresiones-regulares-java-split.html**](http://puntocomnoesunlenguaje.blogspot.com.co/2013/07/ejemplos-expresiones-regulares-java-split.html)

[**http://www.ocpsoft.org/opensource/guide-to-regular-expressions-in-java-part-1/**](http://www.ocpsoft.org/opensource/guide-to-regular-expressions-in-java-part-1/)

[**https://examples.javacodegeeks.com/core-java/util/regex/matcher/validate-password-with-java-regular-expression-example/**](https://examples.javacodegeeks.com/core-java/util/regex/matcher/validate-password-with-java-regular-expression-example/)