CCT College Dublin

Assessment Cover Page

To be provided separately as a word doc for students to include with every submission

Module Title:	Object Oriented Constructs (OOC) Databases (DB)
Assessment Title:	Integrated Databases and OCC
Lecturer Name:	Sam Weiss James Garza
Student Full Name:	Monica Elena Garcia Zamora. Israel Betancourt Hernandez
Student Number:	2022450 2022340
Assessment Due Date:	20/12/2023
Date of Submission:	25/12/2023

Declaration

Introduction:

Databases and object-oriented constructors (OOC) are two fundamental pillars of modern software development. The seamless integration of these concepts is crucial for building robust, scalable, and maintainable applications. In this essay, we will explore the functions of database and object-oriented constructors and elucidate how they work together to create a harmonious relationship in software development.

Database Constructors:

A database constructor is a mechanism used to create and manipulate databases. Databases are organized collections of data that facilitate efficient data storage, retrieval, and management. Database constructors, often implemented through Data Definition Language (DDL) statements, define the structure of the database, including tables, relationships, and constraints.

The primary function of a database constructor is to:

1. **Define Schema:**
   - Specify the structure of the database, including tables, fields, and their data types.
   - Establish relationships between tables to represent the inherent connections within the data.
2. **Enforce Constraints:**
   - Set constraints to maintain data integrity, such as unique key constraints, foreign key constraints, and check constraints.
   - Ensure that the data adheres to predefined rules, preventing inconsistencies and errors.
3. **Manage Indexing:**
   - Create indexes to enhance data retrieval speed, optimizing database performance.
   - Indexing allows for efficient searching and sorting of data based on specific criteria.

Object-Oriented Constructors:

Object-oriented programming is a paradigm that uses objects, which encapsulate data and behavior, to model and design software. In OOP, constructors are special methods responsible for initializing objects by assigning values to their attributes. Object-oriented constructors play a vital role in creating instances of classes and preparing them for use in the application.

The functions of object-oriented constructors include:

1. **Object Initialization:**
   - Set initial values for object attributes when an instance of a class is created.
   - Ensure that objects start with a consistent and valid state.
2. **Encapsulation:**
   - Encapsulate data within objects, providing a clear boundary between an object's internal state and the external world.
   - Enhance modularity and maintainability by grouping related data and functionality.
3. **Inheritance and Polymorphism:**
   - Support inheritance by allowing derived classes to invoke the constructor of the base class.
   - Enable polymorphism, where objects of different classes can be treated uniformly through a shared constructor interface.

Integration of Database and Object-Oriented Constructors:

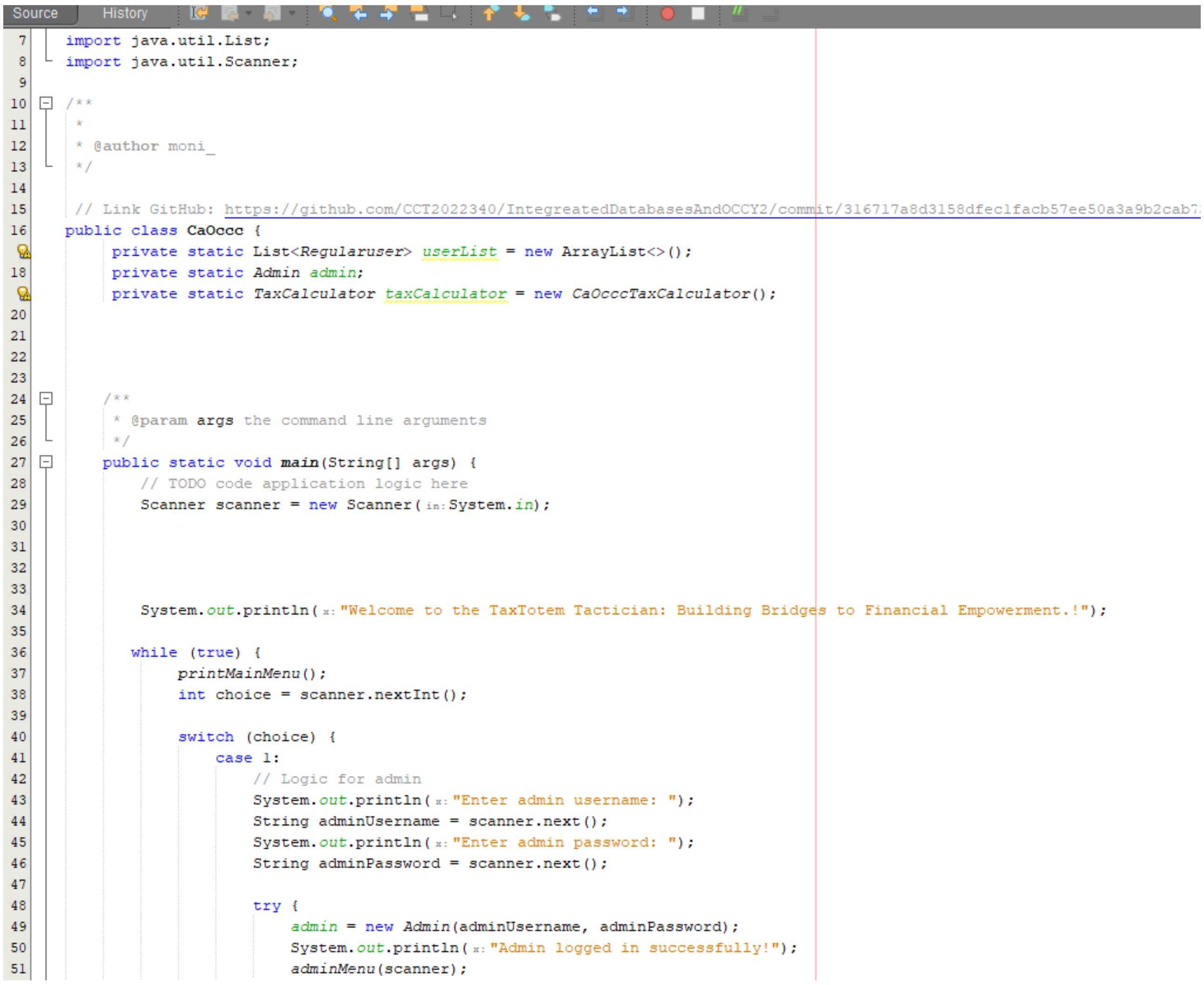
The synergy between database and object-oriented constructors is essential for building modern, data-driven applications. Object-relational mapping (ORM) frameworks, such as Hibernate in Java or Entity Framework in .NET, facilitate the seamless integration of databases with object-oriented programming.

1. **ORM Mapping:**
   - ORM frameworks map database tables to object classes, automating the translation between relational data and object-oriented structures.
   - The ORM framework often provides mechanisms to define relationships, constraints, and other database constructs using object-oriented concepts.
2. **Consistent State:**
   - Object-oriented constructors ensure that objects are initialized with valid data, promoting consistency in the application's state.
   - Database constructors define and enforce constraints that align with the object's integrity, ensuring a coherent representation of data.
3. **Efficient Querying:**
   - ORM frameworks generate SQL queries based on object-oriented constructs, allowing developers to interact with the database using familiar object-oriented syntax.
   - This integration facilitates efficient querying and reduces the need for manual translation between object-oriented and relational paradigms.
   -

In conclusion, the collaboration between database and object-oriented constructors is vital for creating robust, scalable, and maintainable software applications. While database constructors define the underlying structure and constraints of the data storage, object-oriented constructors initialize and encapsulate the application's logic. The integration of these constructs through ORM frameworks bridges the gap between relational databases and object-oriented programming, enabling developers to harness the benefits of both paradigms in a unified and cohesive manner. This synergy lays the foundation for modern software development, where data and functionality seamlessly coexist to meet the ever-evolving demands of the technological landscape.

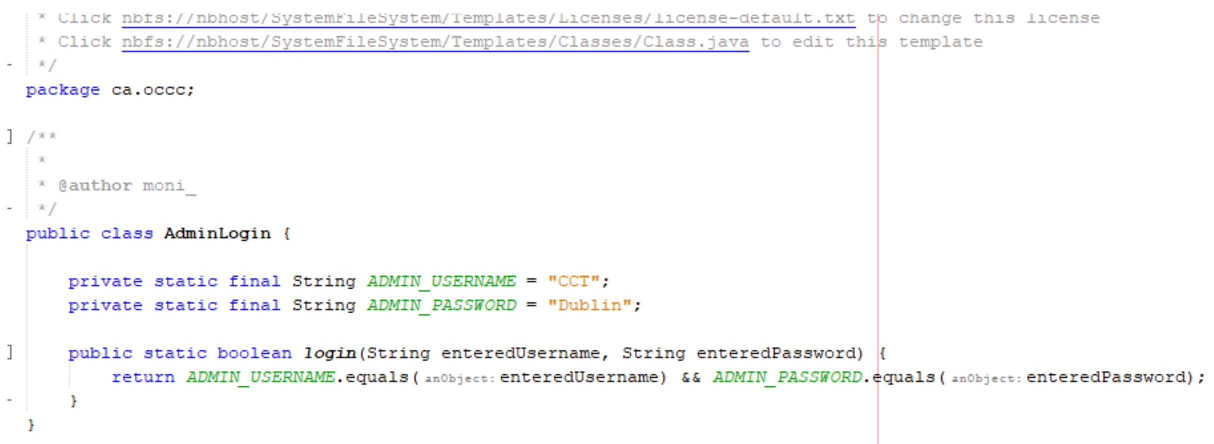
Structure:

This is the sample of the code, where the structure is based on Admin, new login and tax calculator. Admin can organize and manage stuff inside the program.



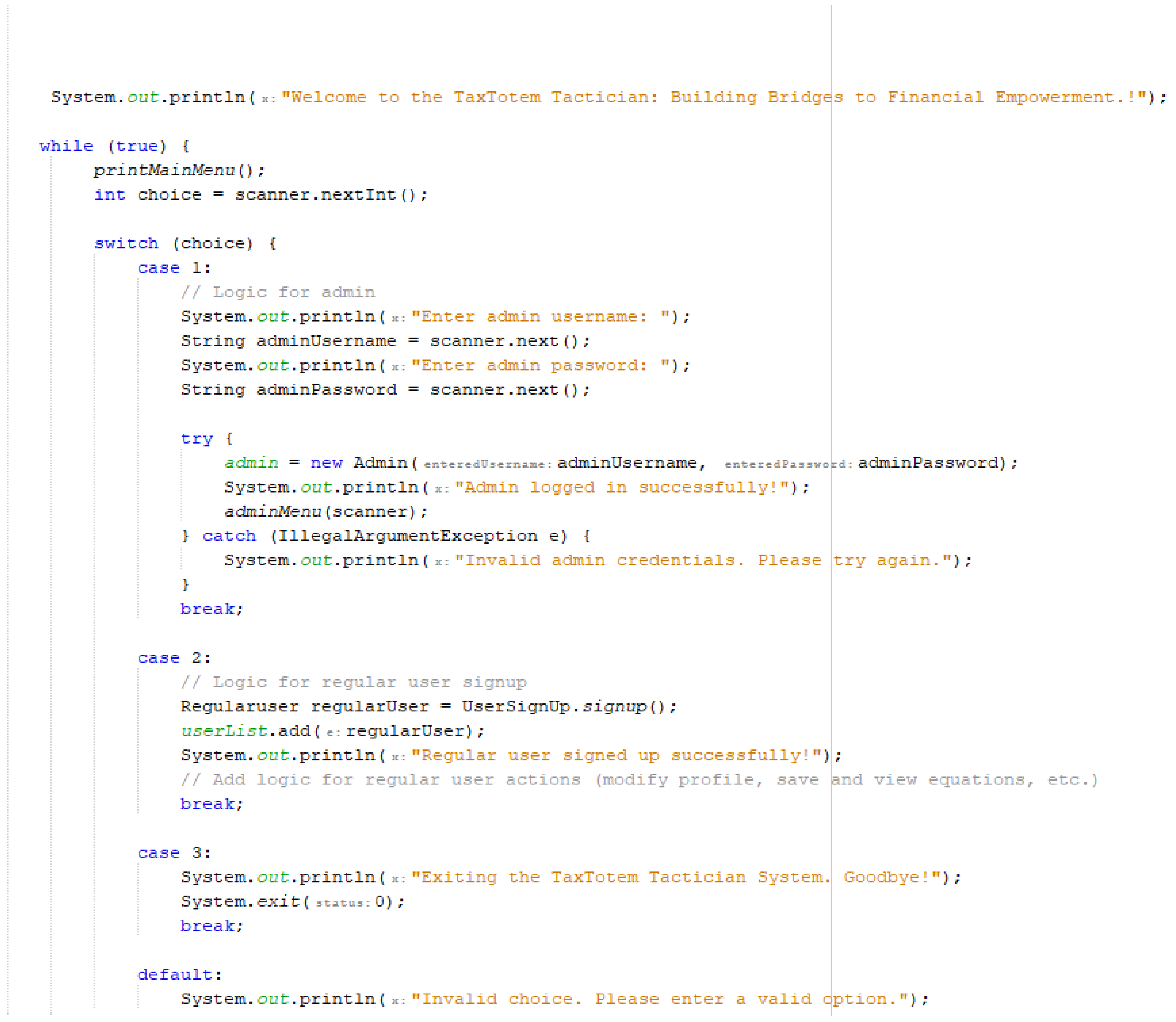
```java
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author moni_
 */

 // Link GitHub: https://github.com/CCT2022340/IntegreatedDatabasesAndOCCY2/commit/316717a8d3158dfec1facb57ee50a3a9b2cab7
public class CaOccc {
     private static List<Regularuser> userList = new ArrayList<>();
     private static Admin admin;
     private static TaxCalculator taxCalculator = new CaOcccTaxCalculator();



    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner scanner = new Scanner( in: System.in);



        System.out.println( x: "Welcome to the TaxTotem Tactician: Building Bridges to Financial Empowerment.!");

       while (true) {
            printMainMenu();
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    // Logic for admin
                    System.out.println( x: "Enter admin username: ");
                    String adminUsername = scanner.next();
                    System.out.println( x: "Enter admin password: ");
                    String adminPassword = scanner.next();

                    try {
                        admin = new Admin(adminUsername, adminPassword);
                        System.out.println( x: "Admin logged in successfully!");
                        adminMenu(scanner);
```

This code is for let the administrator log in to the administrator menu, in case the credentials are wrong, the administrator would not be allowed to access the menu.



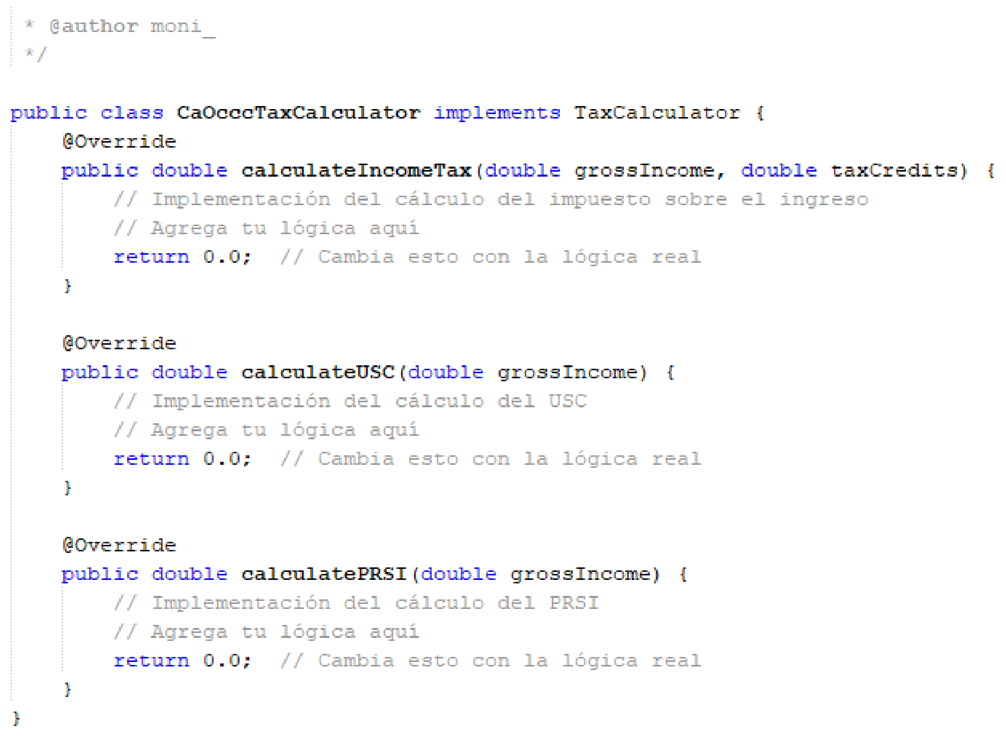
```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package ca.occc;

/**
 *
 * @author moni_
 */
public class AdminLogin {

    private static final String ADMIN_USERNAME = "CCT";
    private static final String ADMIN_PASSWORD = "Dublin";

    public static boolean login(String enteredUsername, String enteredPassword) {
        return ADMIN_USERNAME.equals( anObject: enteredUsername) && ADMIN_PASSWORD.equals( anObject: enteredPassword);
    }
}
```

Here is the user sign up for database and tax calculator where user can just add information into the database.



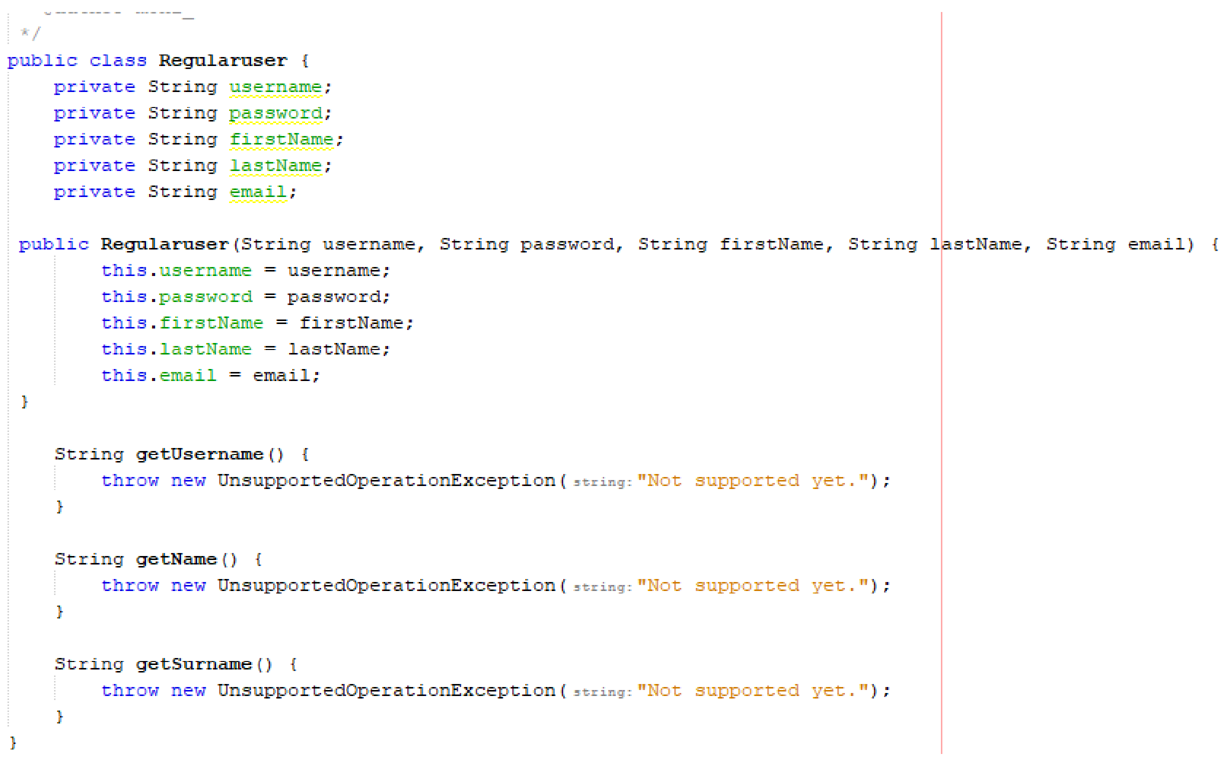
```
    System.out.println( x: "Welcome to the TaxTotem Tactician: Building Bridges to Financial Empowerment.!");

   while (true) {
        printMainMenu();
        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                // Logic for admin
                System.out.println( x: "Enter admin username: ");
                String adminUsername = scanner.next();
                System.out.println( x: "Enter admin password: ");
                String adminPassword = scanner.next();

                try {
                    admin = new Admin( enteredUsername: adminUsername,  enteredPassword: adminPassword);
                    System.out.println( x: "Admin logged in successfully!");
                    adminMenu(scanner);
                } catch (IllegalArgumentException e) {
                    System.out.println( x: "Invalid admin credentials. Please try again.");
                }
                break;

            case 2:
                // Logic for regular user signup
                Regularuser regularUser = UserSignUp.signup();
                userList.add( e: regularUser);
                System.out.println( x: "Regular user signed up successfully!");
                // Add logic for regular user actions (modify profile, save and view equations, etc.)
                break;

            case 3:
                System.out.println( x: "Exiting the TaxTotem Tactician System. Goodbye!");
                System.exit( status: 0);
                break;

            default:
                System.out.println( x: "Invalid choice. Please enter a valid option.");
```

The tax calculator will return double int depending on what the user type in



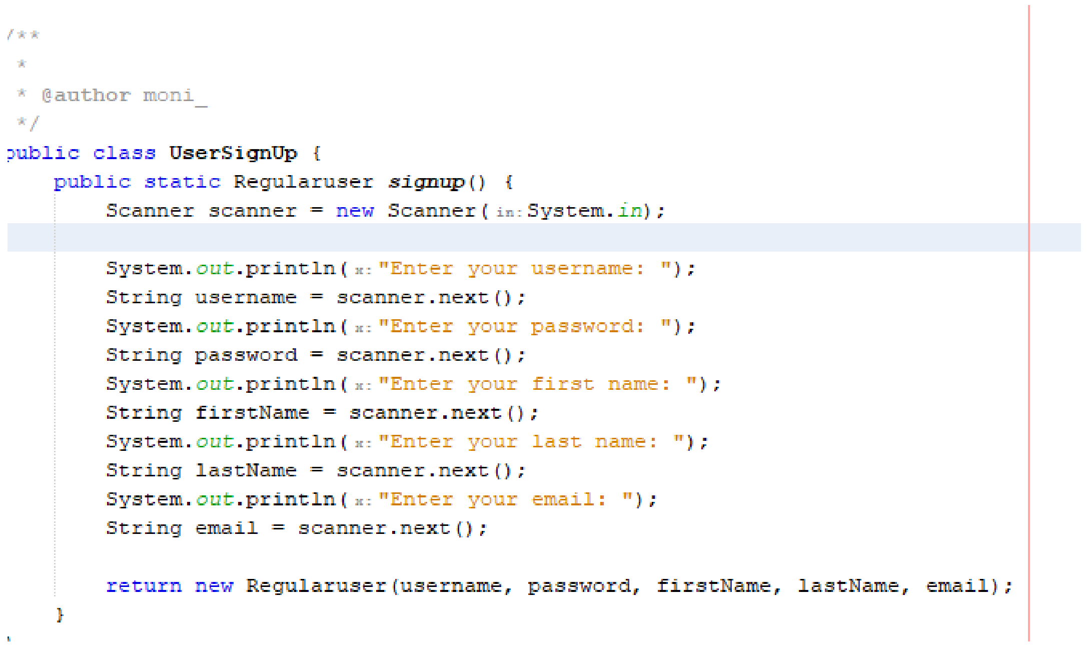
```java
 * @author moni_
 */

public class CaOcccTaxCalculator implements TaxCalculator {
    @Override
    public double calculateIncomeTax(double grossIncome, double taxCredits) {
        // Implementación del cálculo del impuesto sobre el ingreso
        // Agrega tu lógica aquí
        return 0.0;  // Cambia esto con la lógica real
    }

    @Override
    public double calculateUSC(double grossIncome) {
        // Implementación del cálculo del USC
        // Agrega tu lógica aquí
        return 0.0;  // Cambia esto con la lógica real
    }

    @Override
    public double calculatePRSI(double grossIncome) {
        // Implementación del cálculo del PRSI
        // Agrega tu lógica aquí
        return 0.0;  // Cambia esto con la lógica real
    }
}
```

The regular user is to register new users into database and add them into the same one.



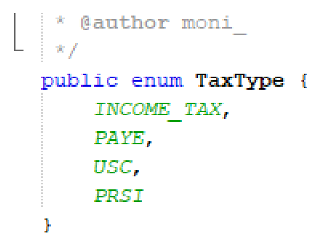
```java
 */
public class Regularuser {
    private String username;
    private String password;
    private String firstName;
    private String lastName;
    private String email;

 public Regularuser(String username, String password, String firstName, String lastName, String email) {
        this.username = username;
        this.password = password;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
 }

    String getUsername() {
        throw new UnsupportedOperationException(string: "Not supported yet.");
    }

    String getName() {
        throw new UnsupportedOperationException(string: "Not supported yet.");
    }

    String getSurname() {
        throw new UnsupportedOperationException(string: "Not supported yet.");
    }
}
```

New user sign up to add into database and calculate their taxes.



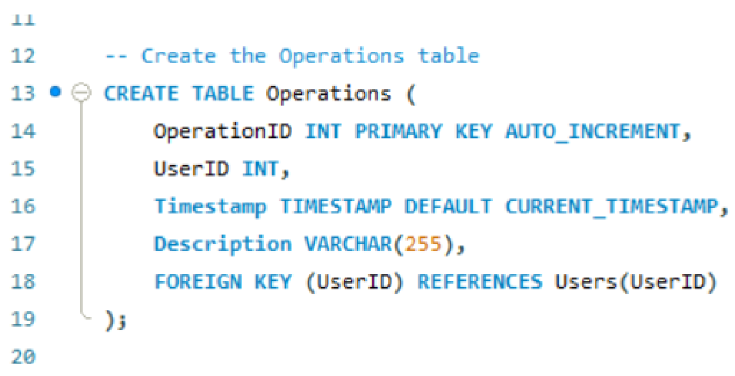
```
/**
 *
 * @author moni_
 */
public class UserSignUp {
    public static Regularuser signup() {
        Scanner scanner = new Scanner( in: System.in);

        System.out.println( x: "Enter your username: ");
        String username = scanner.next();
        System.out.println( x: "Enter your password: ");
        String password = scanner.next();
        System.out.println( x: "Enter your first name: ");
        String firstName = scanner.next();
        System.out.println( x: "Enter your last name: ");
        String lastName = scanner.next();
        System.out.println( x: "Enter your email: ");
        String email = scanner.next();

        return new Regularuser(username, password, firstName, lastName, email);
    }
```

The enum will let us know the tax type and calculate it.

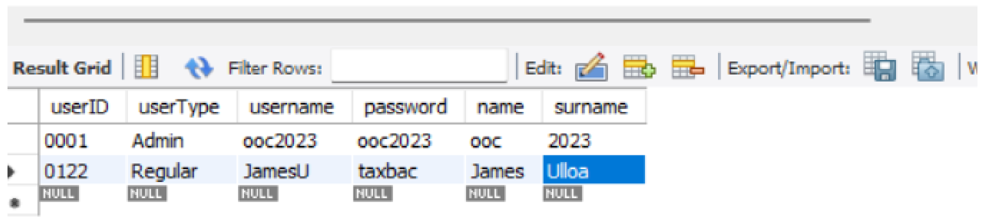


```
 * @author moni_
 */
public enum TaxType {
    INCOME_TAX,
    PAYE,
    USC,
    PRSI
}
```

Whenever we sign up for admin or new user, on the database will be stored and could be used for next time sign in



```
-- Create the Operations table
CREATE TABLE Operations (
    OperationID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Description VARCHAR(255),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```



Result Grid | Filter Rows: | Edit: | Export/Import:

userID	userType	username	password	name	surname
0001	Admin	ooc2023	ooc2023	ooc	2023
0122	Regular	JamesU	taxbac	James	Ulloa
NULL	NULL	NULL	NULL	NULL	NULL

References:

What is a database? (no date) *What Is a Database | Oracle Ireland*. Available at: https://www.oracle.com/ie/database/what-is-database/#:~:text=Is%20a%20Database%3F-,Database%20defined,database%20management%20system%20(DBMS). (Accessed: 27 December 2023).

Constructor (object-oriented programming) (2023) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming) (Accessed: 27 December 2023).

Advanced Data Structures Algorithms & System Design(HLD+LLD) by Logicmojo (no date) *Logicmojo*. Available at: https://logicmojo.com/constructor-in-oops (Accessed: 27 December 2023).

ADBMS 11 object structure and type constructor (2020) *PPT*. Available at: https://www.slideshare.net/VaibhavKhanna21/adbms-11-object-structure-and-type-constructor (Accessed: 27 December 2023).