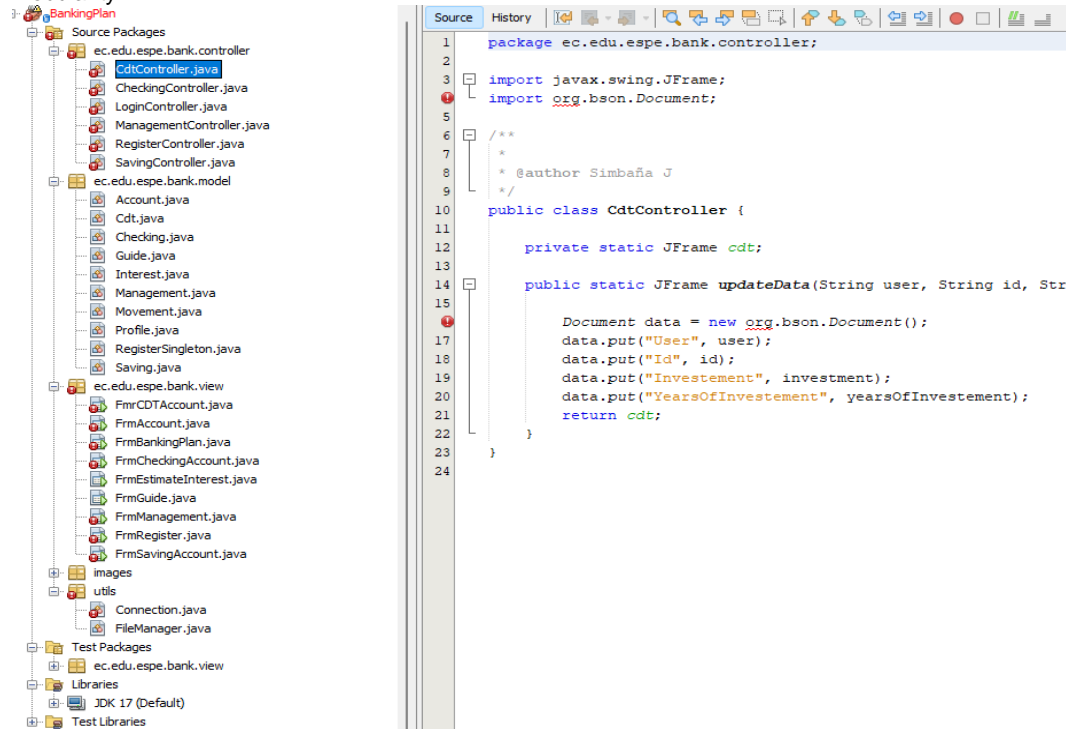


Video: <https://youtu.be/OflyTMYa23U>

Modularity

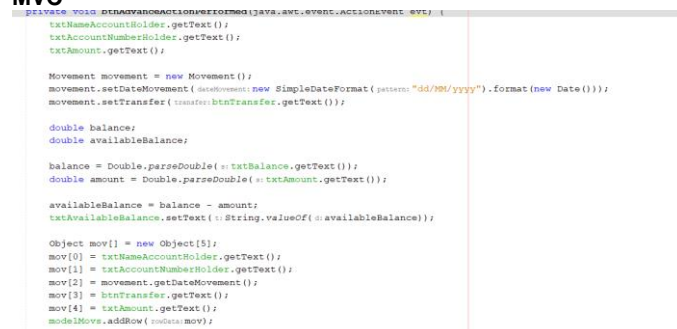


Clean code



button does not call function

MVC



- Code in the btn

```

Source Packages
├── ec.edu.espe.bank.controller
│   ├── CdtController.java
│   ├── CheckingController.java
│   ├── LoginController.java
│   ├── ManagementController.java
│   ├── RegisterController.java
│   ├── SavingController.java
│   └── ec.edu.espe.bank.model
│       ├── Account.java
│       ├── Checking.java
│       ├── Guide.java
│       ├── Interest.java
│       ├── Management.java
│       ├── Movement.java
│       ├── Profile.java
│       ├── RegisterSingleton.java
│       └── Saving.java
├── ec.edu.espe.bank.view
├── images
├── utils
├── Connection.java
├── FileManager.java
├── Test Packages
├── Libraries
├── Test Libraries
├── Team06ApplicationOfHumanResources
├── Team06ApplicationOfHumanResources
└── TestAlgorithm

```

```

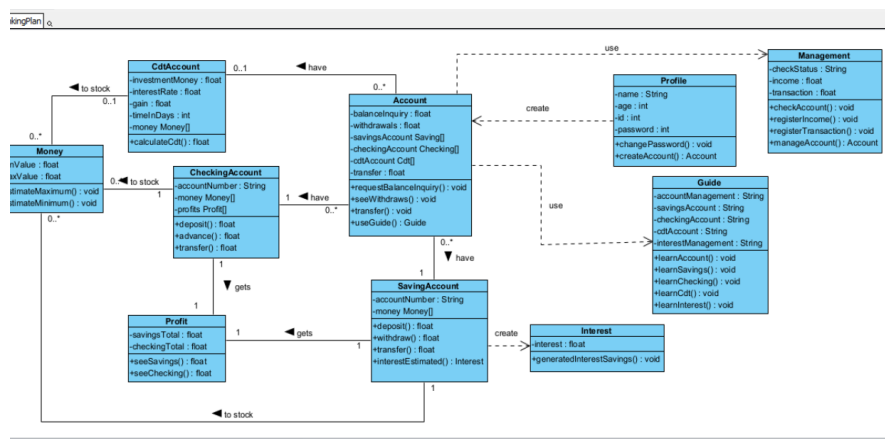
1 package ec.edu.espe.bank.controller;
2
3 import javax.swing.JFrame;
4 import org.bson.Document;
5
6 /**
7  *
8  * @author Simbaña J
9  */
10 public class RegisterController {
11
12     private static JFrame register;
13
14     public static JFrame updateData(String user, String id, String birthdate, String password) {
15
16         Document data = new org.bson.Document();
17         data.put (key: "User", value: user);
18         data.put (key: "Id", value: id);
19         data.put (key: "Birthdate", value: birthdate);
20         data.put (key: "Password", value: password);
21         return register;
22     }
23
24 }

```

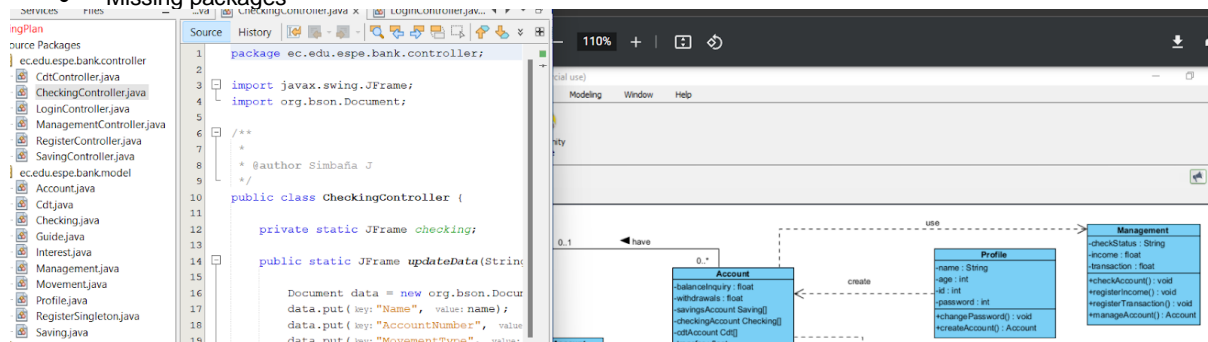
One design pattern

- Singleton

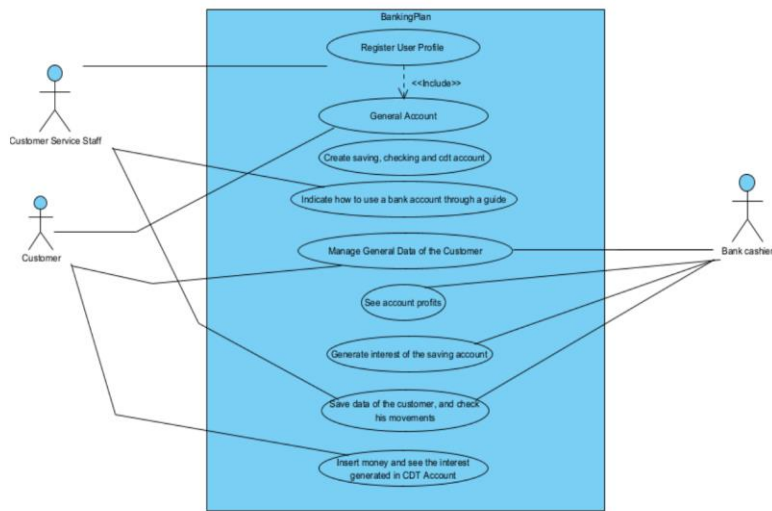
Diagrams consistency



- Missing packages



- Lack of classes and connections



Unit tests

Source Packages

- ec.edu.espe.bank.view
 - FrmCheckingAccountTest.java
 - FrmManagementTest.java
 - FrmSavingAccountTest.java

```

1 package ec.edu.espe.bank.view;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 /**
7  * @author Camila Teca, DEEE-ESPE
8  */
9 public class FrmSavingAccountTest {
10
11     public FrmSavingAccountTest() {
12     }
13
14     /**
15      * Test of deposit method, of class FrmSavingAccount.
16      */
17     @Test
18     public void testDeposit() {
19         System.out.println("deposit");
20         float balance = 23.45F;
21         float amount = 25.34F;
22         FrmSavingAccount instance = new FrmSavingAccount();
23         instance.deposit(balance, amount);
24         float expectedResult = 48.79F;
25         float result = (float) Double.sum(a:balance, b:amount);
26     }
27 }
            
```

Test Results

ec.edu.espe.bankview.FrmCheckingAccountTest x ec.edu.espe.bankview.FrmSavingAccountTest x ec.edu.espe.bankview.FrmManagementTest x

Tests passed: 0.00 %

No test passed, 2 tests caused an error. (1.346 s)

- ec.edu.espe.bankview.FrmManagementTest Failed
- testAdd caused an ERROR: Uncompilable source code - E
- testDeposit caused an ERROR: Uncompilable source code

GUI navigability

Source Packages

- ec.edu.espe.bank.controller
- ec.edu.espe.bank.model
- ec.edu.espe.bank.view
 - FrmCDTAccount.java
 - FrmAccount.java
 - FrmBankingPlan.java
 - FrmCheckingAccount.java
 - FrmEstimateInterest.java
 - FrmGuide.java
 - FrmManagement.java
 - FrmRegister.java
 - FrmSavingAccount.java
- images
- utils
 - Connection.java
 - FileManager.java
- packages
- aries
- libraries

```

127 private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
128     try {
129         JFrame register = RegisterSingleton.getInstance();
130         register.setVisible(b: true);
131     } catch (Exception e) {
132         e.printStackTrace();
133     }
134 }
135
136 private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
137     System.exit(status: 0);
138 }
139
140 private void btnLoginActionPerformed(java.awt.event.MouseEvent evt) {
141     txtPassword.getPassword();
142     anObject:UserTeam) && Password.equals(anObject:PasswordTeam
143     unt();
144     r(parentComponent: this, message: "Incorrect Data, Try Again!")
145     nalStreamConnection.java:568)
146     mConnection.java:447)
147     e(InternalStreamConnection.java:298)
148     mConnection.java:258)
149     ionDescription(InternalStreamConnectionInitializer.java:105)
150     alStreamConnectionInitializer.java:62)
151     on.mongodb.internal.connection.InternalStreamConnection.open(InternalStreamConnection.java:129)
152     at com.mongodb.internal.connection.DefaultStreamConnectionFactory.open(DefaultStreamConnectionFactory.java:117)
            
```

Welcome To Banking Plan

Name User: _____

Password: _____

Register Login Exit

GUI fields validation

Functionality

MongoDB

The screenshot displays the MongoDB Atlas web interface. On the left, a sidebar shows the database structure with a collection named 'CheckingAccount'. The main panel shows the 'BankingPlan.SavingAccount' database with a storage size of 20KB. A 'Find' button is visible. Below the database information, a 'QUERY RESULTS' section shows a single document:

```
{ "_id": ObjectId("6217e16dc5c0543b69724637"),  
  "Name": "Des",  
  "Account Number": "24680",  
  "Movement Type": "Deposit",  
  "Amount": "134.3" }
```

Overlaid on the right is a 'Checking Account' form. The form has a light blue header and contains the following fields and buttons:

- Name User:** Fausto
- Account Number:** 34567
- Balance:** 25
- Buttons:** OK, Clean

Below the form, there is a section for 'Movement Type' with buttons: Deposit, Advance, Transfer, and Return. To the right of these buttons is a text input field for 'Name of the account holder' with the value 'Des'. Below this is another text input field for 'Account Number' with the value '24680'. At the bottom of this section is a text input field for 'Amount' with the value '134.3' and a red asterisk indicating a validation rule: '* Use the point for tenths'. A 'Clean' button is located at the bottom right of this section.

At the bottom of the form, there is a table with the following data:

Name	Account	Date	Type	Amount
Des	24680	24/02/2022	Deposit	134.3