# Banking App Pt.1

Due May 30th 9am

# Description:

Leveraging Java 8, create an application that simulates simple banking transactions

## Requirements:

* Build the application using **Java 8**
* **All interaction with the user** should be done through the **console** using the **Scanner class**

* Customers of the bank should be able to register with a username and password, and apply to open an account.
  + Customers should be able to apply for joint accounts

#### If same color text as other classes, then fields or methods are shared between classes.

#### Will need to create a class that other classes can inherit same fields/methods.

Registration Class:

* Register Customer()
  + Username:
  + Password:
  + Joint Account: Yes or No?
    - Yes. -> 2 Customers & 2 Accounts Total. With Ability to Access Each Other’s Account.

Customer Class:

* Set Customer Personal Information()
  + Customer Username:
  + Customer Password:
  + Customer Account Number:
  + Customer Joint Account
  + Customer First Name:
  + Customer Last Name:
  + Customer Address1:
  + Customer Address2:
  + Customer City:
  + Customer State:
  + Customer Zip:
  + Customer Country:
* View Customer Personal Information()
* Once the account is open, customers should be able to withdraw, deposit, and transfer funds between accounts
  + All basic validation should be done, such as trying to input negative amounts, overdrawing from accounts etc.

Account Class:

* View Customer Account Information()
  + View Transactions
* View Customer Account Balance()
* Withdraw from Customer Account()
* Deposit to Customer Account()
* Transfer to/from Customer Account()
* Employees of the bank should be able to view all of their customers information
  + This includes, account information
  + Account balances
  + Personal information
* Employees should be able to approve/deny open applications for accounts

Employee Class:

* View Customer Account Information()
  + View Transactions
* View Customer Account Balance()
* View Customer Personal Information()
* AccountApplicationDecision()()
  + Approve or Deny?
* Bank admins should be able to view and edit all accounts
  + This includes:
    - Approving/denying accounts
    - withdrawing, depositing, transferring from all accounts
    - canceling accounts

Bank Admin Class:

* View Customer Account Information()
  + View Transactions
* View Customer Account Balance()
* View Customer Personal Information()
* AccountApplicationDecision()
  + Approve or Deny?
* Withdraw from Customer Account()
* Deposit to Customer Account()
* Transfer to/from Customer Account()
* Cancel/Close Customer Account()
* **All information** should be **persisted** using **text files** and **serialization**
* **100% test coverage is expected using J-Unit**
  + You should be using TDD - **Test Driven Development**?
* Logging should be accomplished using Log4J
  + **All transactions** should be **logged**

### Java Serialization

* Java provides a mechanism called object serialization.
* Object serialization is:
  + where an **object** can be **represented** as a **sequence of bytes** that includes the:
    - **object’s data**
    - **object’s type**
* After a serialized object has been written into a file, it can be read from the file and deserialized.
  + Deserialization:
    - The type information and bytes that represent the object and its data can be used to recreate the object in memory.
* ObjectInputStream and ObjectOutputStream classes:
  + contains methods for serializing and deserializing an object
* **Serialization Example Using ObjectOutputStream:**
  + **//** Serialize an Object and send to output stream

**public final void writeObject(Object x) throws IOException**

* **Deserialization Example Using ObjectInputStream:**
  + **//** readObject() method retrieves the next Object out of the stream and deserializes it.

**//** The return value is Object, which then will be ***cast to the appropriate data type*.**

* + **public final Object readObject() throws IOException, ClassNotFoundException**

### Java Serialization Customer Class

package bankapp;

import java.io.Serializable;

@SuppressWarnings("serial")

public class Customer implements Serializable {

public String firstName;

public String address1;

public String city;

public String state;

public String phoneNumber;

public void checkCustomerAccount(){

System.out.println("Name: " + firstName + "\nAddress: " + address1 + ", " + city + ", " + state + "\nPhoneNumber: " + phoneNumber);

}

}

* **Requirements for Serialization:**
  + Meet 2 conditions:
    - The class must implement java.io.Serializable interface
    - All fields in class must be serializable.
      * If not, mark with keyword **transient**.

### Java Serialization & Deserialization BankAppDriver Class

package bankapp;

import java.io.\*;

public class BankAppDriver {

public static void main(String[] args){

// Will replace with Scanner

Customer c = new Customer();

c.firstName = "Jacqueline";

c.address1 = "123 ABC St";

c.city = "Dallas";

c.state = "TX";

c.phoneNumber = "1234567890";

serializeCustomer(c);

deserializeCustomer(c);

}

public static void serializeCustomer(Customer c) {

// Serialize

try{

// Make accounts directory if doesn't exist.

File directory = new File("accounts");

directory.mkdir();

// Create customer.ser file and serialize

FileOutputStream fileOut = new FileOutputStream("accounts/customer.ser");

ObjectOutputStream outStream = new ObjectOutputStream(fileOut);

outStream.writeObject(c);

outStream.close();

fileOut.close();

System.out.println("Serialized data is saved in /accounts/customer.ser");

} catch(IOException i){

i.printStackTrace();

}

}

public static void deserializeCustomer(Customer c) {

//Deserialize

System.out.println("Before c = null -> " + c.firstName);

// Makes c reference null to load data from file into c.

c = null;

try {

FileInputStream fileIn = new FileInputStream("accounts/customer.ser");

ObjectInputStream inStream = new ObjectInputStream(fileIn);

c = (Customer) inStream.readObject();

inStream.close();

fileIn.close();

} catch(IOException i) {

i.printStackTrace();

return;

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

System.out.println("Deserialized Customer: ");

System.out.println("Name: " + c.firstName);

System.out.println("Address: " + c.address1 + ", " + c.city + ", " + c.state);

System.out.println("Phone Number: " + c.phoneNumber);

}

}