CSE 1322L - Lab 5

Introduction

In this lab, you create software for the world's smallest bank. This bank can have only one customer, who will have a checking account and a savings account. The account types have some specific rules:

Checking Accounts:

- Starts empty
- Allows unlimited deposits and withdrawals for free.
- Provides no interest payments.
- If the account balance drops below $0, the customer is charged a $20 overdraft fee every time they try to withdraw.

Saving Accounts:

- Starts with a balance of $500
- Must maintain a $500 balance at all times. Anytime the customer tries to withdraw, if the resulting balance is below $500, they are charged $10.
- Earns 1.5% interest every year
- The first 5 deposits are free, after that there is a fee of $10 per deposit.

You will provide the bank teller with a simple menu which will allow them to make changes to the customer's accounts.

<u>You will be required to submit a UML diagram, along with your code. Check the last pages for details.</u>

Requirements

The features described below must be in your program:

- Account class
    - Has 3 fields: an integer field called "accountNumber", a static integer field called nextNumber (initialized at 10001), and a double field called accountBalance
    - **Account():** assigns nextNumber to accountNumber, increments nextNumber by 1, and initializes accountBalance with 0.
    - **Account(double)**: assigns nextNumber to accountNumber, increments nextNumber by 1, and initializes accountBalance with the argument passed.
    - **double withdraw(double)**: decrements the balance using the argument, returning the new balance

- **double deposit(double)**: increments the balance using the argument, returning the new balance
- **double getAccountBalance()** and **int getAccountNumber()**: Getters for their respective fields
- **String toString():** An override of the toString() method. Must return the following string:

  Account #X, balance $Y

  Where "X" is the account number and "Y" is its balance

- Checking class
  - Is a subclass of Account
  - **Checking(double)**: passes the argument to its superclass overloaded constructor
  - **double withdraw(double):** override of its superclass method. Uses its superclass' withdraw() to decrease the balance using the argument. If the new balance is below $0, prints "Charging an overdraft fee of $20 because account is below $0" and then decreases the balance by another $20, returning the new balance
  - **String toString():** An override of the toString() method from its superclass. Must return the following string:

    Checking Account #X, balance $Y

    Where "X" is the account number and "Y" is its balance

- Savings class
  - Is a subclass of Account
  - Has an integer field numberOfDeposits, initialized at 0
  - **Savings(double)**: passes the argument to its superclass overloaded constructor
  - **double withdraw(double)**: override of its superclass method. Uses its superclass' withdraw() to decrease the balance using the argument. If the new balance is below $500, prints "Charging a fee of $10 because you are below $500" and then decreases the balance by another $10, returning the new balance
  - **double deposit(double)**: override of its superclass method. Uses its superclass deposit() to increase the balance. It then increments the number of deposits on this savings account by 1 and prints "This is deposit X to this account", where "X" is the number of deposits done so far. If this number now exceeds 5, print "Charging a fee of $10", decrementing the balance by $10. Finally, return the new balance

- **double addInterest()**: Prints out "Customer has earned $X in interest", where X is the amount of interest to be applied. It then uses the superclass deposit() to add the interest to the account, returning the new balance
  - Note that the interest is always 1.5%
  - For example, if the customer currently has $1000 and this method is called, it will print "Customer has earned $15 in interest" and then return 1015
- **String toString():** An override of the toString() method from its superclass. Must return the following string:

  Savings Account #X, balance $Y

  Where "X" is the account number and "Y" is its balance

- Driver
  - Create a Checking and a Savings account with balances of $0 and $500, respectively.
  - In a loop, implement the menu below
    1. **Withdraw from Checking**: Prompts the user for an amount to withdraw, then calls the Checking object's withdraw(), printing out the new balance
    2. **Withdraw from Savings**: Prompts the user for an amount to withdraw, then calls the Savings object's withdraw(), printing out the new balance
    3. **Deposit to Checking:** Prompts the user for an amount to deposit, calls the Checking object's deposit(), printing out the new balance
    4. **Deposit to Savings:** Prompts the user for an amount to deposit, calls the Savings object's deposit(), printing out the new balance
    5. **Balance of Checking:** Prints the current balance of Checking using the appropriate getter
    6. **Balance of Savings:** Prints the current balance of Savings using the appropriate getter
    7. **Award Interest to Savings**: Calls the Savings' object addInterest(), printing out the new balance
    8. **Quit**: Terminates the program

Deliverables

- Lab5.java (driver)
- Account.java
- Checking.java
- Savings.java
- uml.pdf (UML diagram)

Considerations

- While you will not lose points for putting all of your classes in a single file, you should prefer to place them in separate files as that usually improves legibility
- You don't need to worry about rounding your values to 2 decimal places as long as your math is correct. Even if your output says "2.5" and the sample output says "2.50", you will still get full credit
- Similarly, you don't need to worry about rounding errors as long as your math is correct. Even if your output says "2.49999999" and the sample output says "2.50", you will still get full credit
- When overriding toString() for both the Checking and Savings classes, it'll be easier to call Account's toString() and just add whatever you need, returning the result

Sample Output (user input in red)

```
[Banking System]

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 3

How much would you like to deposit to Checking? $500
Current balance of Checking is $500.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 4

How much would you like to deposit to Savings? $2000
This is deposit 1 to this account
Current balance of Savings is $2500.00
```

```
1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 7

Customer earned $37.50 in interest
Current balance of Savings is $2537.50

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 1

How much would you like to withdraw from Checking? $250.50
Current balance of Checking is $249.50

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 1

How much would you like to withdraw from Checking? $49.50
Current balance of Checking is $200.00

1. Withdraw from Checking
2. Withdraw from Savings
```

```
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 1

How much would you like to withdraw from Checking? $300
Charging an overdraft fee of $20 because account is below $0
Current balance of Checking is $-120.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 2

How much would you like to withdraw from Savings? $1537.50
Current balance of Savings is $1000.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 2

How much would you like to withdraw from Savings? $510
Charging a fee of $10 because you are below $500
Current balance of Savings is $480.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
```

```
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 4

How much would you like to deposit to Savings? $50
This is deposit 2 to this account
Current balance of Savings is $530.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 4

How much would you like to deposit to Savings? $100
This is deposit 3 to this account
Current balance of Savings is $630.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 4

How much would you like to deposit to Savings? $200
This is deposit 4 to this account
Current balance of Savings is $830.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
```

```
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 4

How much would you like to deposit to Savings? $400
This is deposit 5 to this account
Current balance of Savings is $1230.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 4

How much would you like to deposit to Savings? $70
This is deposit 6 to this account
Charging a fee of $10
Current balance of Savings is $1290.00

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 9

Invalid option.

1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
```

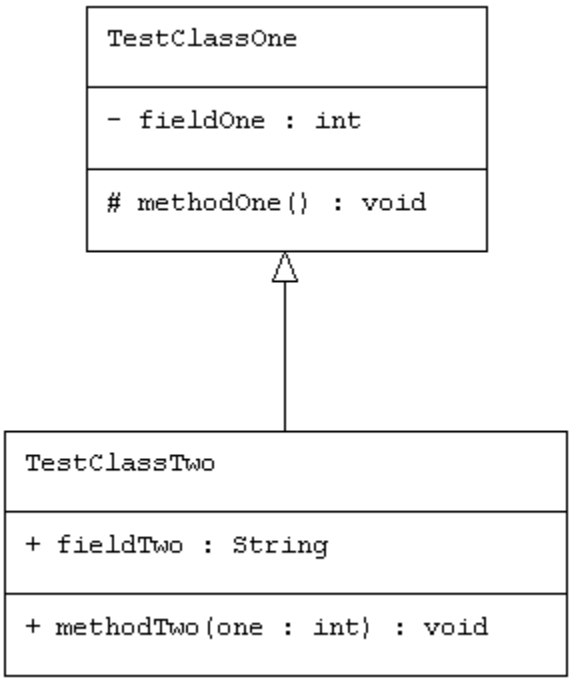
```
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 5

Checking Account #10001, balance: $-120.00


1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 6

Savings Account #10002, balance: $1290.00


1. Withdraw from Checking
2. Withdraw from Savings
3. Deposit to Checking
4. Deposit to Savings
5. Balance of Checking
6. Balance of Savings
7. Award Interest to Savings
8. Quit
Select option: 8

Shutting off...
```

UML Diagram

Your UML diagram must be submitted in its own file, preferably as a PDF; image files are also acceptable. Note that if your grader is unable to open the file or if they are unable to verify that your diagram is correct due to poor formatting, it will not be graded.

The picture below shows what a class should look like in your diagram. Specifically:

- The box representing the class must be divided into 3 sections: name, fields, and behaviors
- The name must match the class name exactly
- The class members (fields and behaviors) must be preceded by the appropriate accessibility symbol (+ for public, - for private, and # for protected)
- A field must be listed as "name : type" (note the colon)
- A behavior must be listed as "name(name : type) : return_type"
- If two classes have a parent-child relationship, there must be an arrow pointing from the child to the parent



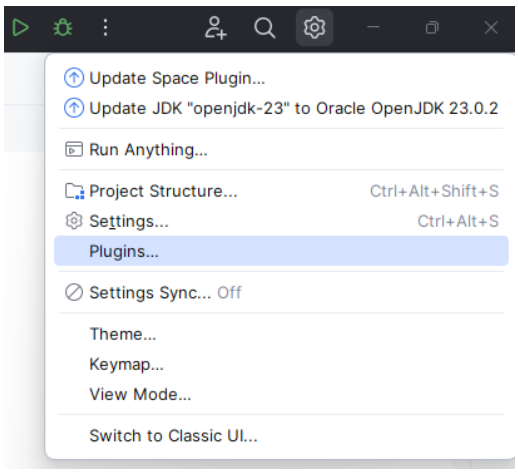
There are no restrictions on which program you use to generate the diagram. In fact, you could even hand-draw your diagram and upload a picture of it along with your code. **You are strongly discouraged from doing this.** Drawing your diagram manually, either by hand or by using any program besides the one listed below increases the chances of you missing something, which will lose you points. It's best to just write your code, and then have a program generate the diagram for you based on your code.

The next pages explains how to install and use the recommended UML diagram generator.

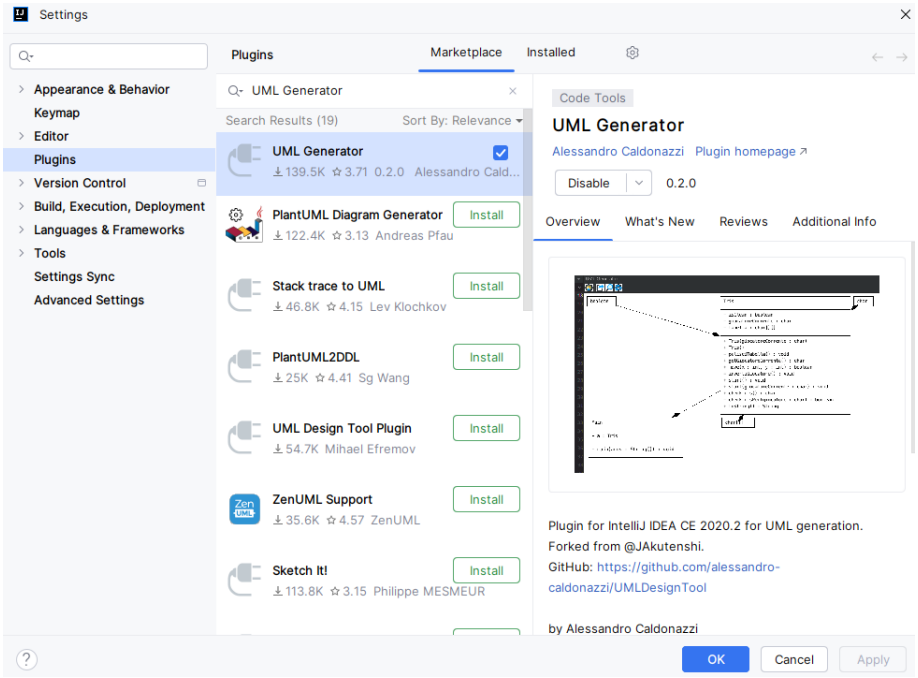
UML Generator

The recommended UML diagram generator is called “UML Generator”, developed by Alessandro Caldonazzi. Instructions on installing and using it can be found below:

- With IntelliJ open, open up your list of plugins by clicking the gear icon at the top right, followed by “Plugins”
    - If you haven’t updated IntelliJ in a while, the gear icon might be an arrow pointing upwards

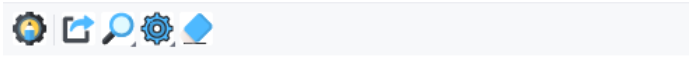


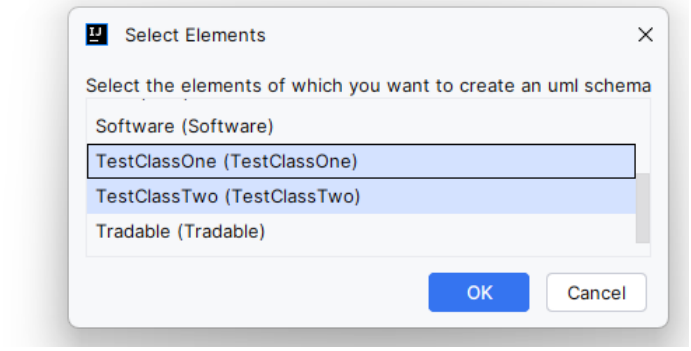
- While in the Plugins menu on the left, select “Marketplace” at the top
- Search for “UML Generator”, developed by Alessandro Caldonazzi



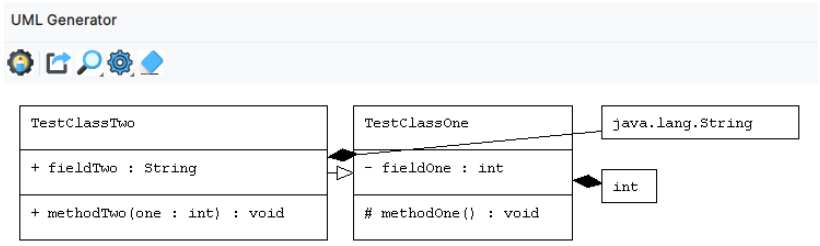
- Click on install. Once it’s done, you may be asked to restart IntelliJ. Please do so.

- On the far right, you will now see a UML button. Clicking on it will show you a blank window with some buttons
- To generate your diagram, click the gear icon with the pencil in the middle (the first button). Then, select all the classes that are part of your diagram using the control key. Finally, click ok.

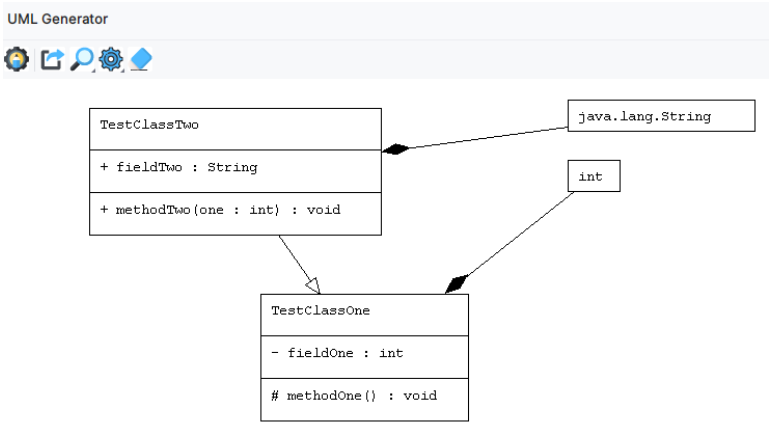




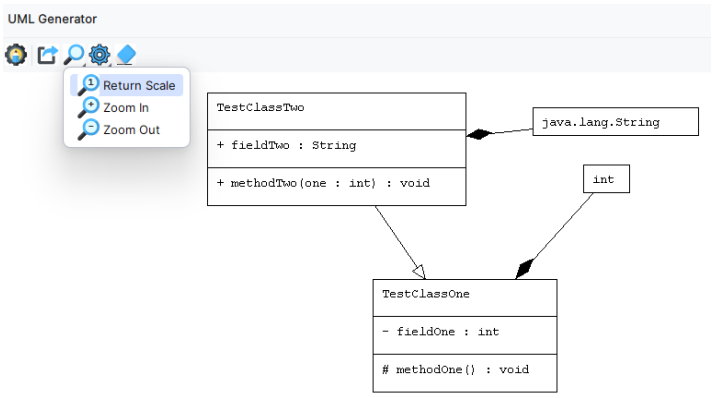
- Your initial diagram will likely look disorganized.



- You can drag its entities with your mouse, to make your diagram more legible



- You can zoom in and out using the magnifier glass button. You can pan around the image using the scroll bars at the right and bottom



- Once your diagram is presentable, click save (the second button). Select a place to save, and give your diagram a name. **Do not use the name provided in the name box, as that will make your diagram fail to save**.
- Don't forget to submit your diagram along with your code to Gradescope.