CP472: Assignment 3
Total Points = 46

In this programming assignment, you will implement a simple **banking system in Python and Java** that allows for account creation, deposit, withdrawal, and displaying account details. Focus on the application of **OOP concepts** such as classes and objects, encapsulation and data hiding, inheritance, and polymorphism.

Tasks:

1. **Python Implementation:**

- Implement the banking system in Python, ensuring that all OOP concepts are appropriately applied.
- Test the system by creating instances of SavingsAccount and CheckingAccount, performing transactions, and displaying account details.

2. **Java Implementation:**

- Implement the same banking system in Java, focusing on the same OOP concepts.
- Test the system in Java by creating objects of the subclasses and performing similar operations as in the Python implementation.

Main Requirements:

1. **Classes and Objects:**

- Create a BankAccount class with attributes such as accountNumber, accountHolderName, and balance.
- Include methods for deposit(amount), withdraw(amount), and displayAccountDetails().

2. **Encapsulation and Data Hiding:**

- Ensure that sensitive information can only be accessed or modified through appropriate methods.

3. **Inheritance:**

- Extend the BankAccount class to create two subclasses: SavingsAccount and CheckingAccount. Each subclass can have additional attributes or methods, such as a minimum balance for savings accounts.

4. **Polymorphism:**

- Demonstrate polymorphism by overriding a method in the subclass. For example, the withdraw method in SavingsAccount might include a check for a minimum balance, while the CheckingAccount version does not.

5. **Error Handling:**

- Handle common errors or exceptions in both the Python and Java implementations. For example, what should happen if a withdrawal amount exceeds the account balance? Other examples include handling invalid input types, negative deposit or withdrawal amounts, or attempting to create an account with an existing account number (if you are not autogenerating the account numbers).

6. **Driver Program, Testing and Validation:**

- Write a driver program where you create objects of your classes and call appropriate methods interactively. For example, when creating a new account, ask the user about the type of account, account holder name, and initial deposit amount.
- Include testing code to validate the functionality of your banking system. The testing code could be similar to the one provided to you for your assignments in CP213, and CP264 such as testSingleStack(), testSingleQueue(). You may name your testing functions testSavingsAccount(), testCheckingAccount() etc.

7. **Report:**

- Write a brief report (1 page) comparing the OOP concepts in both languages based on your experience.
- Include a UML diagram in the report to visualize the class hierarchy and relationships. The UML diagram must be clear and legible.

Additional Requirements:

- The requirements stated above are only a subset of requirements, and you are required to add "some" more features. Examples of these features include:
    - Registered savings account(s) such as TFSA and RRSP
    - Debit and credit cards, ATM machines, and online banking
    - Loans, and mortgages.
- You have the flexibility to choose which features you want to include. The minimum requirement is that you have at least six classes (including the three major classes BankAccount, SavingsAccount, and CheckingAccount) where every class is related to at least one other class either through an inheritance or a composition relationship.
- You are encouraged to understand the working of a typical banking system before designing your class hierarchy and deciding about the class members. The following links could be helpful in doing so.
    - https://www.nerdwallet.com/ca/banking/types-of-bank-accounts
    - https://www.securities-administrators.ca/investor-tools/understanding-your-investments/types-of-savings-plans/
    - https://www.investopedia.com/terms/m/mortgage.asp
    - https://www.investopedia.com/articles/personal-finance/050214/credit-vs-debit-cards-which-better.asp

Marking Scheme:

		Marks
Main Requirements	Creation of classes with appropriate data members and member functions	10
	Correct and appropriate use of data hiding	4
	Correct and appropriate use of inheritance	4
	Correct and appropriate use of polymorphism	4
	Error handling	4
	Driver program including the testing	6
	Report	4
	Code readability (naming conventions, formatting, comments)	2
Additional Requirements	Have three additional classes and appropriate use of OOP features for those classes	8
Bonus Marks	More than six classes with appropriate use of OOP features for those classes (roughly 1 bonus mark for each additional class up to a maximum of 4)	(4)
	Total	**46**

Deliverables

Submit a zip file containing:

- The source code files for Python (.py), and Java (.java) programs.
- Include a README file if your code has any dependencies (e.g., 3rd party libraries).
- A PDF file named "Assignment 3_Report" as specified in Task 7.

Instructions and Comments

- Follow best practices for code readability, such as consistent naming conventions, code formatting, and the use of comments.
- Where applicable, the marks are equally divided between the Java and Python implementations. For example, "Correct and appropriate use of inheritance" will have 2 marks each for Java and Python.