AFRICA UNIVERSITY
A United Methodist-Related Institution

"Investing in Africa's Future"

COLLEGE OF ENGINEERING AND APPLIED SCIENCES

NCIS 215: OBJECT-ORIENTED PROGRAMMING

END OF FIRST SEMESTER EXAMINATIONS

NOVEMBER 2024

LECTURER: MR BRAITON U MUKHALELA

TIME: 3 HOURS

INSTRUCTIONS

You are required to answer questions as instructed in each section

Start **each** question on a new page in your answer booklet

Answer *all* questions in Section **A** and any *three* from Section **B**

Credit will be awarded for logical, systematic and neat presentations

---

Section A :Total Marks: 40

Mini Project Exam Question: GUI-Based ATM with SMS Notification

Project Title: Development of a GUI-Based ATM System with SMS Notification

Objective: Create a simple ATM application with a graphical user interface (GUI) that allows users to perform basic transactions (withdrawal, deposit, balance inquiry) and sends SMS notifications via the Twilio API after each transaction.

Project Requirements and Marking Scheme

1. **User Interface (10 marks)**
    - Design a user-friendly GUI using a framework of your choice (e.g., Tkinter for Python, JavaFX for Java).
    - The interface should include:
        - Input fields for account number and PIN. (3 marks)
        - Buttons for transaction options: Withdraw, Deposit, Balance Inquiry, and Exit. (3 marks)
        - Display area for transaction confirmations and balance information. (4 marks)
2. **Basic ATM Functionalities (10 marks)**
    - Implement the following functionalities:
        - **Login:** Validate user credentials (account number and PIN). (3 marks)
        - **Withdraw:** Allow users to withdraw money, ensuring the amount does not exceed the available balance. (4 marks)
        - **Deposit:** Allow users to deposit money into their account. (3 marks)
        - **Balance Inquiry:** Display the current account balance. (3 marks)
3. **Twilio SMS Integration (5 marks)**
    - After each transaction (withdrawal, deposit), send an SMS notification to the user using the Twilio API. (3 marks)
    - Include transaction details (e.g., transaction type, amount, new balance) in the SMS message. (2 marks)
4. **Data Storage (5 marks)**
    - Use a simple data structure (like a dictionary or a JSON file) to store user account information, including account numbers, PINs, and balances.

      (3 marks)

    - Ensure that the data persists across sessions (e.g., by saving to a JSON file).

      (2 marks)

5. **Error Handling (5 marks)**
    - Implement error handling for:
        - Invalid user credentials. (2 marks)
        - Insufficient funds for withdrawals. (2 marks)
        - Invalid transaction amounts (e.g., negative numbers). (1 mark)

6. **Documentation (5 marks)**
    - Include comments in your code explaining key sections. (3 marks)
    - Prepare a brief user manual detailing how to use the ATM application.

(2 marks)

Bonus Features (Optional, not included in total marks)

- Implement multi-user support (multiple accounts).
- Add transaction history display.
- Enhance the user interface with improved design elements.

Submission Requirements

- Submit your source code files along with any dependencies.
- Provide a demonstration of the application functionality.

Section B: Total 60 marks

Short Practical Programming Tasks

Instructions: Write your code in java programming language and submit the source files.

1. **Task 1: Create a Class (15 points)**
    - Define a class named `Car` with the following attributes:
        - `make` (string)
        - `model` (string)
        - `year` (integer)
    - Include methods to:
        - Display car details.
        - Update the year of the car.
2. **Task 2: Implement Inheritance (15 points)**
    - Create a base class named `Animal` with a method `speak()`.
    - Define two subclasses: `Dog` and `Cat`, each overriding the `speak()` method to return a string representing the sound they make.
3. **Task 3: Demonstrate Polymorphism (15 points)**
    - Create a class named `Shape` with a method `area()`.
    - Implement subclasses `Circle` and `Rectangle`, each providing their own implementation of the `area()` method.
    - Write a function that takes a list of `Shape` objects and prints their areas.
4. **Task 4: Encapsulation and Abstraction (15 points)**
    - Define an abstract class `Employee` with a method `calculate_salary()`.
    - Create two subclasses: `FullTimeEmployee` and `PartTimeEmployee`, each implementing the `calculate_salary()` method.
    - Include private attributes for each class and provide public methods to set and get these attributes.
5. **Task 5: Advanced Topics (15 points)**
    - Implement operator overloading in your chosen programming language.

- Create a class `Vector` that represents a mathematical vector, and overload the `+` operator to add two vectors together.

Submission Guidelines

- Ensure that your code is well-commented.
- Submit all source files in a single ZIP folder.
- Include a README file explaining how to run your programs.

END OF EXAMINATION