Uganda Christian University

Department of Computing and Technology

Course: CSC2105: Object-Oriented Programming

Test 1

Submission Format: Upload a single Jupyter Notebook file (`.ipynb`) named

`AccessNumber_Test1.ipynb`
(e.g., `b2.._Test1.ipynb`)

Notebook Requirements

1. **Organize your notebook properly:**
    - Use **Markdown cells** to write your theory answers and explain your design choices.
    - Use **Code cells** to write and run your Python code.
    - Each question should start with a Markdown heading like:
      `## Question 1 — Encapsulation in Daily Reality`
2. Include your **Student Access and Registration Numbers, full name** in the **first Markdown cell**.
3. Use **Python 3.x only**. Code must run offline in Jupyter (no imports outside the standard library).
4. Do not share or collaborate. Submissions with identical logic or variable naming will both receive **zero**.
5. Comment all code explaining *why* validations or rules exist, not just what they do.
6. Ensure all Markdown answers are clear and written in your **own words**.

Total Marks: 100 Language: Python 3.x

Scope: Weeks 1 – 3 (Classes, Objects, Encapsulation, Validation, UML basics)

Question 1 (25 marks): Encapsulation in Daily Reality

In a Markdown cell:

1. Explain what encapsulation is and how it differs from data hiding.
2. Describe a real-life case from your family, community, or village where encapsulation could prevent misuse of data or resources (≤150 words).

In another Markdown cell:

- Draw (in text) a small ASCII UML diagram showing the class design for your example.

Then in a Code cell:

- Implement your class using classic getters/setters (no `@property`).
- Add one validation rule that makes sense in your example (and explain why it matters in a comment).
- Demonstrate your code with at least **two runs** — one valid, one invalid.

Question 2 (25 marks): Alpha MIS Simulation

In a Markdown cell:

- Describe briefly how encapsulation can be applied in a university management system like Alpha MIS.

Then in a Code cell:

- Implement a class representing one small part of Alpha MIS (e.g., course registration, meal plan, or fees record).
- Use encapsulation (`public`, `_protected`, and `__private` attributes).
- Add at least **two validation checks** relevant to your faculty (e.g., credit limit, fees threshold).
- Include a `summary()` or `report()` method printing your Access Number, faculty, and timestamp.

Marks: Design (8), Validations (8), Encapsulation (5), UCU Context (4)

Question 3 (25 marks): Hostel Visitor Audit

In a Markdown cell:

- Briefly describe how a digital visitor log could help maintain accountability in UCU hostels.

Then in a Code cell:

- Implement a class that stores only the **latest visitor entry** in a **dictionary** (no lists or tuples).
- Validate that names contain only letters and spaces.
- Create methods: `record()`, `update()`, `show_line(StudentID)`
- Raise and handle at least one exception for invalid data.
- Print a clear, formatted audit line with Student ID, hostel name, and timestamp.

Question 4 (25 marks): Creative Encapsulation Challenge

In a Markdown cell:

- Choose one real-world system you interact with regularly (mobile money, school bursar, SACCO, market stall, church tithe records, etc.).
- Explain in ≤100 words how encapsulation can help maintain trust or prevent fraud in that system.

Then in Code cells:

- Design **two classes** that interact via encapsulation.
    - One stores private/confidential data and exposes only safe methods to update or view it.
    - The other interacts with it externally without breaking encapsulation.
- Enforce at least one **district-specific numeric rule** (e.g., max transaction, minimum float, limit per day).
- Demonstrate how your design prevents direct modification of internal data.