Khaula Molapo

C++ 2 wk 1

Year 2 Semester 2

Question 1:

In C++, object-oriented programming (OOP) allows programmers to model real-world situations using classes and objects. A bank account is a common example. We can create a BankAccount class with private data members such as accountNumber, balance, and accountHolderName. Public member functions like deposit, withdraw, and displayBalance let users interact with the account safely. Using constructors, we can initialize accounts when objects are created. Encapsulation ensures that sensitive information, like balance, is protected and cannot be accessed directly outside the class. Objects of BankAccount represent individual accounts, and we can perform operations independently. Inheritance can create specialized accounts, such as SavingsAccount or CheckingAccount, extending the main class. Polymorphism allows us to override functions for different account types, while abstraction keeps the interface simple. This approach makes programming organized, realistic, and easier to maintain, allowing secure operations on multiple accounts without errors.

Question 2

Encapsulation is an OOP principle that hides the internal details of a class from outside access and allows interaction only through public functions. In C++, private or protected variables store data, while public methods control access. This ensures sensitive information cannot be changed accidentally and keeps the system consistent. For example, a bank account class can hide the balance variable and allow deposits or withdrawals only through validated functions. Encapsulation improves security, reduces bugs, and makes code easier to read and maintain. By combining data and methods in a single class, encapsulation organizes the program logically. It also simplifies debugging and allows programmers to understand code behaviour easily.

Question 3

Writing a rectangle class in Visual Studio Code helped me understand core OOP concepts in C++. I defined private variables, length and width, and created public functions to calculate area and perimeter. Using constructors allowed me to initialize objects efficiently. Testing the program reinforced the importance of encapsulation because private variables could not be accessed directly. Writing this class showed how objects represent real-world entities and how functions interact with data. The exercise also taught me about class structure, object creation, and using methods effectively. Overall, it improved my understanding of OOP and made programming more organized, practical, and easier to manage.

Question 4

A C++ class diagram visually shows the structure of a class and its members. In Canva, I created a rectangle representing the class. The top section shows the class name, the middle section lists attributes or variables, and the bottom section contains methods or functions. For example, a Rectangle class diagram includes length and width as private attributes and calculateArea and calculatePerimeter as public methods. This diagram clarifies how data and functions are grouped. It helps programmers plan code, understand relationships, and organize classes before coding. Visual diagrams make learning and reviewing OOP concepts simpler and more effective for understanding class design.