**Week 10**

Q1 Write a report on the fundamental concepts of object orientation.

Object-Oriented Programming (OOP) is a paradigm that organizes software design around data, or **objects**, rather than functions and logic. The primary purpose of OOP is to increase the modularity, reusability, and maintainability of code by creating objects that interact with one another. These objects represent entities from the real world, encapsulating both the state (data) and behavior (methods). Below are the fundamental concepts of object orientation:

**1. Objects**

Objects are instances of classes and are the basic units of OOP. An object consists of:

* **Attributes**: These represent the properties or characteristics of an object (also known as fields or variables).
* **Methods**: These are functions associated with the object, defining its behavior or actions.

**2. Classes**

A **class** is a blueprint or template from which objects are created. It defines a set of attributes and methods that an object will have. While a class is a static definition, an object is a dynamic instance created from that class.

**3. Encapsulation**

Encapsulation is the concept of bundling data (attributes) and methods that manipulate the data within a single unit or object. It also restricts direct access to some of the object’s components, typically using access modifiers like private, protected, and public.

Encapsulation helps in:

* **Data Protection**: By hiding internal state from the outside, we protect the integrity of the data.
* **Modularity**: Changes within the class do not affect other parts of the program, promoting modular design.

**4. Inheritance**

Inheritance is the mechanism by which one class can inherit attributes and methods from another class, creating a hierarchy. The class that is inherited from is called the **parent** (or base or superclass), and the class that inherits is called the **child** (or subclass).

Benefits of inheritance:

* **Code Reusability**: Common functionalities need to be written only once in the base class.
* **Extensibility**: New functionalities can be added in derived classes without modifying the base class.

**5. Polymorphism**

Polymorphism allows methods to be used in different forms. It enables objects of different types to be treated as objects of a common superclass. The two types of polymorphism are:

* **Compile-time polymorphism** (Method Overloading): Multiple methods in the same class with the same name but different parameters.
* **Runtime polymorphism** (Method Overriding): A subclass provides a specific implementation of a method that is already defined in its superclass.

**6. Abstraction**

Abstraction refers to the concept of hiding complex details and showing only the essential features of an object. It allows the focus on the functionality rather than the implementation. Abstract classes and interfaces in OOP help in achieving abstraction.

**7. Relationships Between Classes**

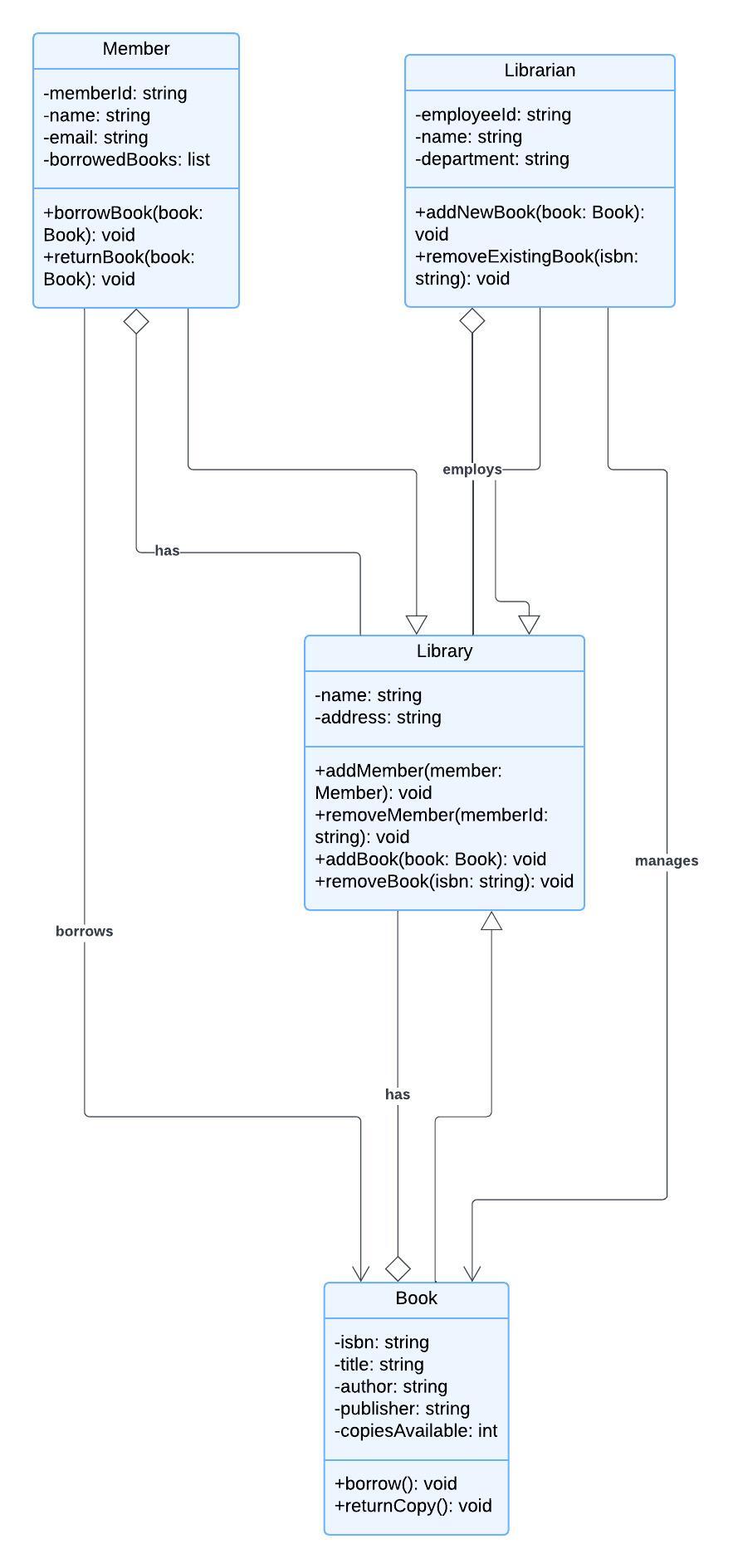
OOP supports different types of relationships between classes, such as:

* **Association**: A general relationship where one object uses or interacts with another.
* **Aggregation**: A “whole-part” relationship where one object is composed of one or more objects, but the part can exist independently of the whole.
* **Composition**: A stronger “whole-part” relationship where the parts are dependent on the whole (if the whole is destroyed, the parts are destroyed too).

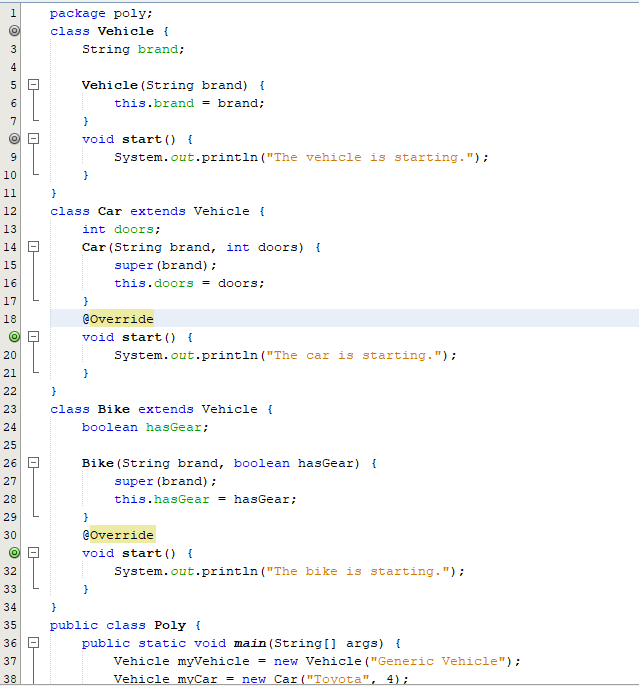
**Conclusion**

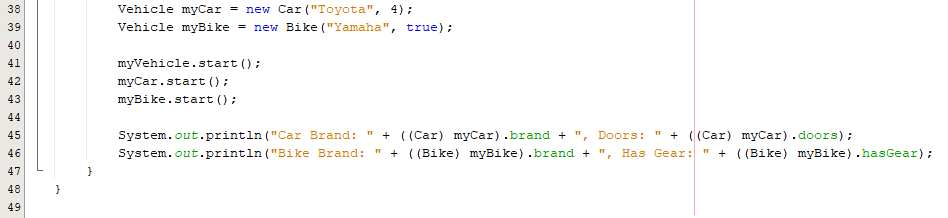
Object orientation is a powerful paradigm that improves the design and structure of software systems. By focusing on objects that represent real-world entities, encapsulating data and behaviors, and using concepts like inheritance and polymorphism, OOP enhances modularity, maintainability, and scalability. It encourages developers to write reusable and flexible code, making it easier to manage complex systems. Through abstraction and encapsulation, OOP ensures that implementation details are hidden, while inheritance and polymorphism promote code reuse and flexibility in designing systems.

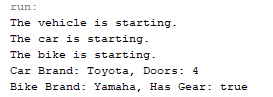
Q2. Create a UML class diagram for a simple system.



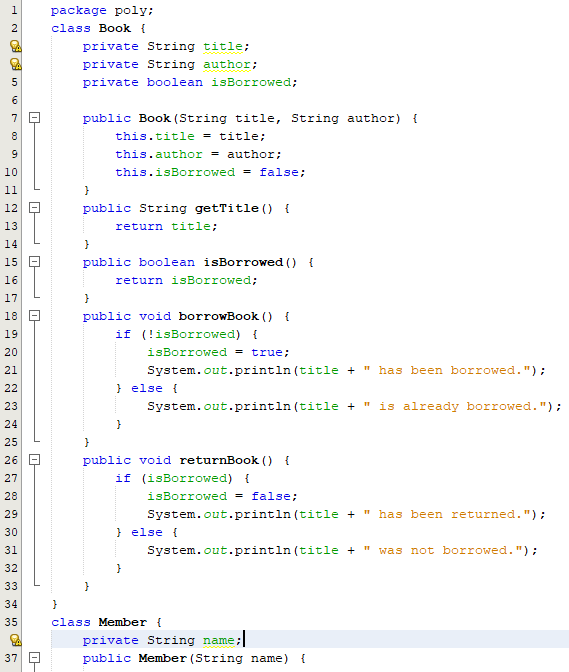
Q3. Write a C++/Java program to demonstrate inheritance and polymorphism.

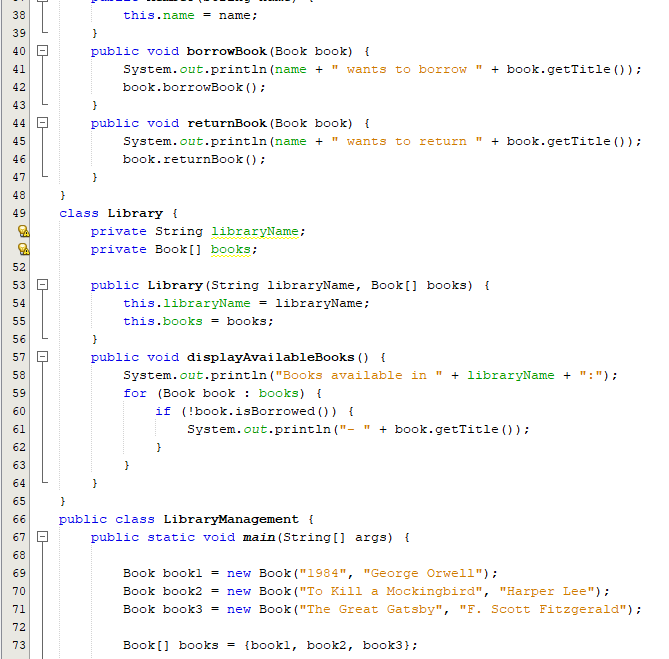
****

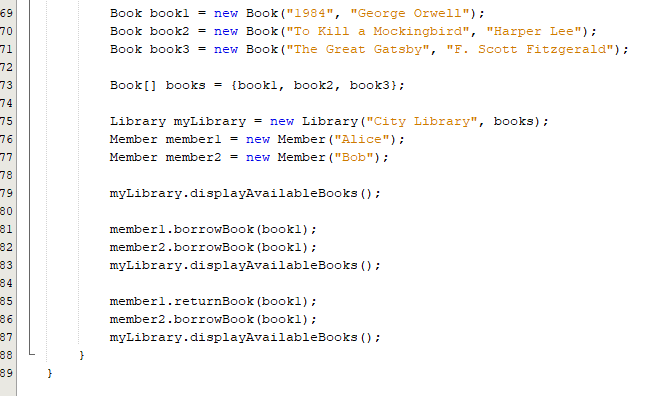
****

****

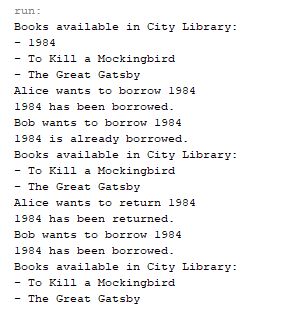
Q4. Implement a C++/Java program to simulate object-oriented modeling.

****

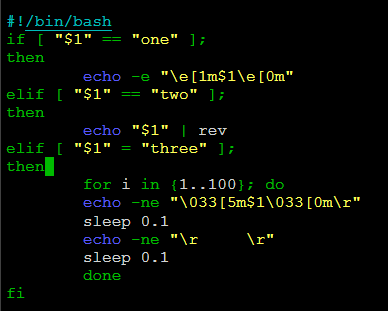
****

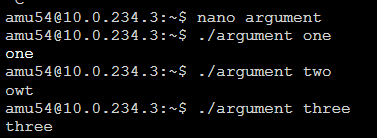
****

**Output**

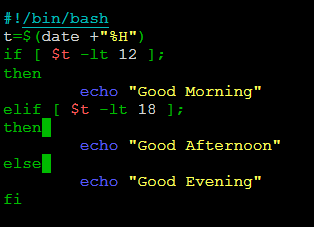
****

Q5. A shell script can receive an argument ‘one’, ‘two’, or ‘three’. If the argument supplied is ‘one’ display it in bold, if it is ‘two’ display it in reverse order and if it is ‘three’ make it blink on the screen. If a wrong argument is supplied report it. (Hint: Use an elif statement)





Q6. Write a shell script to display the message “Good Morning” / “Good Afternoon” / “Good Evening” depending upon the current time.



**Q.6 output.png**

Q7. Write a shell script program to calculate overtime pay of 10 employees. Overtime is paid at the rate of Rs. 12.00 per hour for every hour worked above 40 hours. Assume that the employees do not work for fractional part of an hour.

