**Name: KRISH PANCHAL**

**PRN: 22070122102**

**Year: 2022-26**

**CodeX OOPS Assignment**

**Q1. What is the fundamental concept of Object-Oriented Programming?**

Ans: The fundamental concept of ​​Object-Oriented Programming (OOP) is to organize and organize software with objects. Objects represent real-world objects and concepts, and contain both data (attributes) and methods (functions) that operate on that data. OOP promotes the idea of ​​modeling software after the way we think and interact with the real world, making it easier to design, understand and maintain complex systems

**Q2. How does OOP differ from Procedural programming paradigms?**

Ans: OOP differs from Procedural programming in many ways:

Data abstraction: OOP focuses on abstract data types (objects) and stores their contents, whereas program design exposes data and functions separately

Encapsulation: OOP contains data and methods in objects, ensuring data integrity. Procedural programming does not have this level of encapsulation.

Inheritance: OOP allows the creation of new classes by deriving properties and methods from existing classes, and encourages code reuse. Procedural programming is task-based and has no built-in property management.

Polymorphism: OOP allows objects of different classes to be treated as objects of a common hyperclass. This allows for dynamic method dispatch and flexibility in the use of resources. Procedural programming does not have this feature.

**Q3. Explain the four main principles of OOP: encapsulation, inheritance, polymorphism, and abstraction?**

There are four basic principles of OOP:

Encapsulation: Encapsulation consists of data (attributes) and methods (functions) that act on that data in a single unit called an object This helps in storing data and ensures that the internal state of the object can be accessed and modified by what well-defined so interfaces are used.

Properties: Properties are tools that allow you to create new classes by accessing properties and methods from existing classes. It encourages repetitive rules and class-based planning.

Polymorphism: Polymorphism enables you to handle different objects as objects of a common hyperclass. This enables flexibility with dynamic method dispatch, where the appropriate method of execution is specified at runtime.

Abstraction: Abstraction seeks to simplify a complex reality by creating classes based on the expression of important attributes and actions. It hides unnecessary information and highlights relevant parts of the object.

**Q4. What is a class, and how is it related to objects in OOP?**

Ans: Class in OOP is a blueprint or template for doing things. It explains the structure and behavior of things. A class defines the attributes (data) that an object of that class will have and the methods (operations) that can be performed on those attributes. Objects are instances of classes and are created based on class definitions. In simple terms, a class defines the attributes and behaviors that an object should have, and objects are literal instances of those attributes and behaviors.

**Q5. Provide an example of a class and its objects in a real-life context and explain?**

Ans: Real life example: "Bank Account".

**Class:** The "Bank Account" class can be thought of as a template for creating individual bank accounts. It describes the structure and actions of the bank account, including "account number", "account owner name", "balance", "deposit", "withdrawal", "check\_balance" and other options

**Objects:** Instances of the "Bank Account" class represent individual bank accounts. Each account item has its own unique account number, account owner name, and balance. For example, you can create items such as "John's Savings Account," "Alice's Checking Account," and "Bob's Retirement Fund" for different customers' bank accounts.

**Explanation:** In this example, the "Bank Account" class provides common templates for all bank accounts. Objects in this class represent specific instances of bank accounts, each with different characteristics and options. The class teaches deposits and withdrawals, balance management, and accounting information. Each account object contains its own unique information, such as the account owner’s name and account number. By creating objects from the "Bank Account" class, you can effectively manage and communicate with individual bank accounts while complying with common policies and activities

**Q6. How does inheritance work in OOP, and what are the advantages and disadvantages of using inheritance?**

Ans: Inheritance in OOP allows a class to inherit properties and methods from other classes, encourages code reuse and structures the classes. The benefits of inheritance are:

**Code reusability**: Inherited classes can reuse attributes and methods of the base class, reducing unnecessary code.

**Modularity**: Inheritance encourages modular design, making it easier to maintain and update code.

However, there are also some disadvantages of inheritance:

**Strong Connection:** Subclasses are tightly coupled with their parent class. Changes to the base class **can affect subclasses, and can cause unpredictable behavior.**

**Inadequate flexibility:** Succession can create a rigid class structure, making it difficult to adapt to changing needs.

**Complexity**: Deep legacy systems can be complex and difficult to understand.