2. a) An interface defines the signature operations of an entity, it also sets the communication boundary between two entities, in this case two pieces of software. It generally refers to an abstraction that an asset provides of itself to the outside.

The main idea of an interface is to separate functions from implementations. Any request that matches the signature or interface of an object may also be sent to that object, regardless of its implementation.

Since it does not matter which implementation of a specific class is used, a class can be exchanged easily without changing the code of the calling class.

The concept of an interface is fundamental in most object oriented programming languages. In some, objects are known only through their interfaces so there is no way to access an object without going through its interface.

b) We can use **Factory pattern** for this solution. Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

We can another use **Prototype pattern** refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves implementing a prototype interface which tells to create a clone of the current object. This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, returns its clone on next request and update the database as and when needed thus reducing database calls.

**Command pattern** is a data driven design pattern and falls under behavioral pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

3. I used Command pattern in three functionalities. Command pattern is a data driven design pattern and falls under behavioral pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.