7. Object Oriented Design

These questions require you to sketch out the classes and methods to implement technical problems. These problems give an interviewer insight into your coding style.

How to approach

Step 1: Handle Ambiguity

Object-Oriented Design (OOD) questions are often intentionally vague to test whether you'll make assumptions or if you'll ask clarifying questions. When asked OOD questions, you should enquire the 5W1H.
The different use cases will significantly impact your design.

Step 2: Define the Core Objects

For example, suppose we are asked to do OOD for a restaurant. Our core object might be things like Table, Guest, Party, Order, Meal, Employee, Server, Host, etc.

Step 3: Analyze Relationships

Now we want to analyze the relationships between the objects. Which objects are members of which objects? Do any objects inherit from any others? Are relationships many-to-many or one-to-many?

Using the above example, we may come up with the following design:

- Party should have an array of guests
- Server and Host inherit from Employee
- Each Table has one Party, but each Party may have multiple Tables
- There is one Host for the Restaurant

These are all just assumptions. Check with your interviewer how general purpose your design should be.

Step 4: Investigate Actions

Now, consider the key actions that the objects will take and how they relate to each other.

Design Patterns

While design patterns are mostly beyond the scope of an interview, Singleton and Factory Method design patterns are widely used in interviews.

Singleton Class

The Singleton pattern ensures that a class has only one instance and ensures access to the instance through the application. It can be useful in cases where you have a "global" object with exactly one instance. For example, we may want to implement Restaurant such that it has exactly one instance of Restaurant.

```
public class Restaurant {
  private static Restaurant _instance = null;
  protected Restaurant() {...}
  public static Restaurant getInstance() {
    if (_instance == null) {
      _instance = new Restaurant();
    }
    return _instance;
  }
}
```

Factory Method

The Factory Method offers an interface for creating an instance of a class, with its subclasses deciding with class to instantiate. You might want to implement this with the creator class being abstract and not providing an implementation for the Factory method.

```
public class CardGame {
  public static CardGame createCardGame(GameType type){
    if (type == GameType.Poker) {
      return new PokerGame();
    } else if (type == GameType.BlackJack) {
      return new BlackJackGame();
    }
    return null;
  }
}
```

Questions

Question 1: Deck of Cards: Design the data structures for a generic deck of cards. Explain how you would subclass the data structures to implement blackjack.

Assuming that the deck is a standard 52-card set, there are a few things that we can do:

- creating Enum Suit
- create an abstract class Card (because different values on the card may different things on different games, and we can just inherit this base card class) that has faceValue and suit attributes
- create class Hand to hold the cards (in an arraylist)

Full code on pages 304 - 307.

Note that on any OOD question, there are many ways to design the objects. Discuss trade-offs of different solutions. You should usually design for long-term code flexibility and maintenance.