**Why we need to use OOP? Some major OOP languages?**

In time, the need to transfer real-life objects to the computer environment arose. With the transfer of each object to the computer environment with attributes such as color, state, shape, and function, the processing load has increased. OOP, on the other hand, is a programming method that allows data to be stored correctly, easily accessible, and easily manipulated using class logic. Commonly used languages such as Java, Python, C++, C# are programming languages that can be given as examples of this programming method.

**Interface vs Abstract Class?**

Interface and Abstract classes are abstraction techniques that can be used interchangeably.

Abstract Class;

It can contain a constructor and a destructor.

Methods can have implementations.

can contain static methods.

Can express what the class derives from (is a).

It should be used to impart common classroom behavior.

Interface;

It cannot contain a constructor or a destructor.

Cannot contain overridable static methods.

Can express what abilities the class has. (can do).

The common skill method should be used for gaining.

**Why we need equals and hashcode? When to override?**

The Equals method checks if the contents of the 2 values are the same. Hashcode checks if their addresses are the same.

We use the equals() method to know that 2 values hold the same value, but because the places where these 2 data are held are different, we need to override hashcode().

**Diamon problem in Java? How to fix it?**

The Diamond problem is when 2 classes inherited from one Class can be created at the same time by inheriting from another class, and the methods of these 2 main classes are mixed.

For example, let's create the "hairs on the paw" method of the "cat" and "dog" classes, which are inherited from the "stray animals" class. Then confusion will arise when we call the "hairs on the paw" method for the cat and dog class in the "pets" class. While stray animals don't have feathers on their paws, they do in pets.

We can have the same default methods (same name and signature) in two different interfaces and, from a class, you can implement these two interfaces. If we do so, we must override the default method from the class explicitly specifying the default method along with its interface name.

**Why we need Garbage Collector? How does it run?**

The Garbage Collector takes control of the buffer. Garbage Collector in Java allows you to delete unnecessary data and free up memory space.

Garbage Collector initially stores it in the "eden" region allocated in memory, when this region is full, the referenced data in the "eden" region is moved to the "S0 survived" region with the "minor garbage collector" method, and the others are deleted. When the "eden" is full, the "minor garbage collector" runs again, but the referenced data from the "S0 Survived" region is moved to S1 and the referenced data in "eden" is also moved to S1 and S0 is deleted. By repeating this process, the data with the reference survives and the ones without the reference die.

**Java “static” keywords usage?**

The users can apply static keywords with variables, methods, blocks, and nested classes. The static keyword belongs to the class than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.

When a variable is declared as static, then a single copy of the variable is created and shared among all objects at the class level. Static variables are, essentially, global variables. All instances of the class share the same static variable.

**Immutability meaning? Where, How and Why to use it?**

Immutable objects are immutable objects. You create them, but you cannot change them. Instead, if you want to replace an immutable object, you must clone it and modify the clone while creating it. Immutable objects are useful in multi-threaded environments and streams. Relying on immutable objects is great. There may be errors caused by a thread modifying another thread's object. Immutable objects would have solved all of these problems.

To give an example of common ones in Java, String and all primitive wrappers (Integer, Long, Double, Byte….), BigDecimal, BigInteger are immutable classes.

**Composition and Aggregation means and differences?**

*Aggregation* implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

*Composition* implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate from a House.

**Cohesion and Coupling means and differences?**

Cohesion refers to what the class (or module) can do. Low cohesion would mean that the class does a great variety of actions - it is broad, unfocused on what it should do. High cohesion means that the class is focused on what it should be doing, i.e. only methods relating to the intention of the class.

As for coupling, it refers to how related or dependent two classes/modules are toward each other. For Low Coupled classes, changing something major in one class should not affect the other. High coupling would make it difficult to change and maintain your code; since classes are closely knit together, making a change could require an entire system revamp.

Good software design has high cohesion and low coupling.

**Heap and Stack means and differences?**

A stack is a system where data is stored independently of the index. For example, a Stack such as "{1,2,3,4}" and 1 is the first in and 4 is the last element, in this case, we cannot reach other elements without removing element 4, so there is a "first in last out" situation. Data in the stack is stored sequentially in memory. The advantage of this structure is that it is fast and low cost.

Heap, on the other hand, can be stored in different places in memory. Instead of saying "a=1,b=2,c=3,d=4", it can be thought of as a definition like "{1,2,3,4}". This notation is not quite correct, because Heap is an "iterable" system. To variables can be accessed with the index numbers.

**Exception means? Type of Exceptions?**

The exception is a method by which we can predict the situations in which we may get errors in the program, ignore the situations where the program will get an error, or notify us. For example, it can be used in a structure that can ignore our Index Error when we call an object with an index more than the number of elements of a list. There are two types of exceptions; Check and Uncheck.

**How to summarize “clean code” as short as possible?**

It is easy to read, understandable by the team, written in accordance with the naming conventions (CamelCase).

**What is the method of hiding in Java?**

Method hiding can be defined as, "if a subclass defines a static method with the same signature as a static method in the superclass, in such a case, the method in the subclass hides the one in the superclass." The mechanism is known as method hiding. It happens because static methods are resolved at compile time.

**What is the difference between abstraction and polymorphism in Java?**

Abstraction ensures that the correct method works, taking care of the Classes of the methods with the same name. For example, we can get the "meow" output from the cat class with the sound() method, as well as the "baff" output with the sound() method in the dog class.

Polymorphism, on the other hand, lets there be 2 methods with the same name and one of them takes 2 parameters and the other 1 parameter. If we call it with how many parameters, the related method will run automatically.