Ödev - 1

Q1 - Why do we need to use OOP? Some major OOP languages?

A: OOP helps developers to reduce problems into smaller pieces and enables data-oriented programming which can be manipulated with objects' methods later. The four fundamental concepts of OOP provide opportunities for data hiding, split up projects into smaller pieces, and scale up much easier. The most-known OOP-based languages are; Java, C++, C#.

Q2 - Interface vs abstract class?

A: Abstract class can have both abstract and non-abstract methods while interfaces can only have abstract methods. Further, a class can extend only one other abstract class meanwhile it can implement multiple interfaces. Interfaces provide and define merely the state of the class on the other hand abstract class allows you to characterize the functionality of the methods and override them if unfavorable.

Q3 - Why do we need equals and hashcode? When to override?

A: Object.hashCode() needs to be overriden in cases Object.equals() is overriden. The reason for this is that hash based collections relies on Object.hashCode() which will be inaccurate for cases that Object.equals() is modified.
Override is to be executed when the inherited method does not function as we want it to or needs modification for the sub-class.

Q4 - What is the diamond problem in Java? How to fix it?

A: The diamond problem is simply not being able to inherit multiple parent classes. To overcome this issue, interfaces can be implemented and methods can be overridden.

Q5 - Why do we need a garbage collector? How does it run?

A: Garbage collection is an automated procedure that clears up the memory from unused objects.

Q6 - Java “static” keyword usage?

A: Static variables or methods are the types themselves instead of belonging to an instance. They are initiated or initialized before any instances are created, and shared across all the instances. It’s mostly used for scenarios where the static variable does not rely on objects, instead, it is to be shared across all objects of the type.

Q7 - Immutability meaning? Where, how and why to use it?

A: An immutable object is an object whose entire state will remain persistent and continuous in time after it is initially created. A consistent and conflict-free state of the program can be achieved using immutability.

Q8 - Composition and aggregation meanings and differences?

A: Both composition and aggregation are “has-a” or “belongs-to” relationships. The difference between them is that composition is a stronger relationship unlike aggregation is looser of a relationship. In a composition type of relationship, one object is completely dependent on another one. As an example, a building has rooms and if it is destroyed, the rooms cannot exist. In an aggregation kind of, the dependency is looser. The objects rely on each other to be fully functional yet the lack of one of them does not have fatal considerations on other’s existence.

Q9 - Cohesion and coupling meanings and differences?

A: Cohesion represents the functional relationships of the module. The higher the cohesion is, the better design the software has. In contradiction, Coupling means the interdependence of modules. The lower the coupling is, the better design the software has.

Q10 - Heap and stack meanings and differences?

A:

Q11 - Exception meaning and type of exceptions?

A:

Q12 - Summarize “clean code”?

A: It is easily readable, modifiable, and maintainable code. It is supposed to be understood when read by another person.

Q13 - What is the method hiding in Java?

A: Method hiding is when a child class overrides a static method in its parent class.

Q14 - What is the difference between abstraction and polymorphism in Java?

A: Abstraction is reducing the complexity of implementation and hiding essential structures. Polymorphism is when classes have an inheritance association, mostly those classes have an “IS-A” relationship.