Beaconfire Inc, Home Work, Week1 Day3.

By (Ping) Nalongsone Danddank.
ndanddank@gmail.com
wechatID: ndanddank

Short Answer:

1. What is Object in Java and why we need object?

-> An Object is an instance of class in Java. Each object in Java has an identity, behavior, and state. And we need object to stores its state in fields(variables) and exposes its behavior through methods(functions).

https://docs.oracle.com/javase/tutorial/java/concepts/object.html

https://en.wikipedia.org/wiki/Object_(computer_science)

2. What is Inheritance and how many types of inheritance are supported by Java?

-> The ability to derive functions or features and state or field from another thing, and that can help us for reusing of code, then we can also add or modify its to be our own. There are 3 type of inheritance in Java like: single, Multilevel, Hierarchy.

3. What is the diamond problem in Java? And how can we resolve the problem?

-> it is an ambiguity when we call a method of super class that extends from 2 classes that have the same method, then when we call it, JVM will confuse or ambiguity to call which one. So that arise as a consequence of allowing multiple inheritance because Java does not support multiple inheritance.

https://www.tutorialspoint.com/what-is-diamond-problem-in-case-of-multiple-inheritance-in-java#:~:text=Then%2C%20if%20you%20call%20the,more%20than%20one%20other%20class.

4. What is Interface and what is abstract class? What are the differences between them?

-> Interface similar to class, can have variables and methods, but the methods that declared in interface are abstract by default, it means only signature without body implementation, However, In Java version 8 up, we can declare default method and static method those can not inheritance to subclass. Abstract class is also like normal class but there are at least one abstract method in the class and we need to put “abstract” keyword to in front class. There are class with a generic concept, not related to specific class. They define some behavior, then let the rest for subclasses to implement. The one difference between them is we can implements one or more then one for interface but only extends one class for abstract class. Another thing, we can define abstract class like normal class and let subclass implements at least one abstract method. But for interface we like to define like a blueprint for class, and let subclass implement all abstract method in the blueprint.

5. What is Polymorphism? And how Java implements it?

-> it means we can perform one action in different ways. In java, there are 2 types of polymorphism like: compile time -> convert .java to .class(byte code), and runtime -> when .class(byte code) executing or running.

6. What is the differences between overriding and overloading?

-> Overriding is determinded at running time, it happens when there are same method on super class and subclass, the method in subclass will override on supper class. Overloading is in compile time, so it happen when we have the same name of two or more method but they are different return type or arguments.

7. What is Encapsulation? How Java implements it? And why we need encapsulation?

-> Encapsulation is the way for hiding information. Like, we can let the class's fields or methods to be access by public, private, and other ways. We just put the keyword like "public, private, protect" or empty to the front of fields or method, so we can control the access ability from outside of class.

8. What is the difference between abstraction and encapsulation?

-> Abstraction is hiding how it does it. And Encapsulation is hiding what data from outside to access. Encapsulation = data hiding + abstraction.

9. What is toString() and why we need it?

-> for representing the object of instance class by String. We have to overriding the toString() method which come from Object supper class. So we need this method for formatting the object to be String to show the information of their fields. If we do not override this method and print it out, we will see the memory address of object that JVM created.

10. Can we use this keyword in constructor and why?

-> "this" is the keyword to represent the instance of class when we create an object. So when we use "this" in constructor to avoid confusing when we have the same name parameters to fields.

https://docs.oracle.com/javase/tutorial/java/javaOO/thiskey.html