9 Lecture - CS506

Important Subjective

Certainly, here are 10 subjective short questions along with their answers related to Abstract Classes and Interfaces:

****Question 1: What is an abstract class, and why would you use it in Java?****

****Answer:** An abstract class is a class that cannot be instantiated and can have both abstract (unimplemented) and concrete (implemented) methods. It serves as a blueprint for other classes to inherit from, enabling code reuse and enforcing a common structure for subclasses.**

****Question 2: Explain the concept of multiple inheritance and how Java deals with it using interfaces.****

****Answer:** Multiple inheritance refers to a class inheriting from more than one class. Java avoids the complications of multiple inheritance by using interfaces. A class can implement multiple interfaces, enabling it to inherit method signatures from multiple sources without inheriting implementation details.**

****Question 3: How does an abstract class differ from an interface in Java?****

****Answer:** An abstract class can have both abstract and concrete methods, while an interface can only have abstract methods. A class can implement multiple interfaces, but it can only extend a single abstract class. Abstract classes can have instance variables, while interfaces can only have constants.**

****Question 4: Can an abstract class have a constructor? Explain.****

****Answer:** Yes, an abstract class can have a constructor. The constructor is called when an instance of a concrete subclass is created. It can initialize instance variables or perform other necessary setup operations.**

Question 5: What are default methods in Java interfaces?

Answer: Default methods are methods in interfaces that have a default implementation. They allow adding new methods to existing interfaces without breaking the implementing classes. Concrete classes implementing the interface can choose to override or use the default implementation.

Question 6: How do you achieve method overloading and overriding in interfaces?

Answer: Interfaces support method overloading by allowing multiple methods with the same name but different parameter lists. Method overriding in interfaces is similar to class inheritance; a subclass implementing an interface must provide a method with the same name, parameters, and return type as the parent interface.

Question 7: Explain the importance of the "implements" keyword in Java interfaces.

Answer: The "implements" keyword is used to signify that a class is adopting the contract defined by an interface. It ensures that the class provides implementations for all the methods declared in the interface, establishing a strong relationship between the class and the interface.

Question 8: Can an interface extend a class in Java? Why or why not?

Answer: No, an interface cannot extend a class in Java. An interface only defines method signatures, and it's not meant to inherit or extend class implementations. However, an interface can extend another interface to inherit its method signatures.

Question 9: What is the purpose of a marker interface?

Answer: A marker interface is an interface that does not declare any methods but is used to provide metadata about a class implementing it. Examples include the `Serializable` interface in Java, which indicates that a class can be serialized.

Question 10: In which situations would you prefer using an abstract class over an interface, and vice versa?

Answer: Use an abstract class when you want to provide a base implementation and share code among related subclasses. Use an interface when you want to define a contract that multiple unrelated classes can implement, enabling them to fulfill a common role without inheritance constraints.