Interface Implementation

- Interfaces are contracts for what a class can do, but they say nothing about the way in which the class must do it.

- Interfaces can be implemented by any class, from any inheritance tree.

- An interface is like a 100-percent abstract class, and is implicitly abstract whether you type the abstract modifier in the declaration or not.

- An interface can have only abstract methods, no concrete methods allowed.

- Interfaces are by default public and abstract—explicit declaration of these modifiers is optional.

- Interfaces can have constants, which are always implicitly public, static, and final.

- Interface constant declarations of public, static, and final are optional in any combination.

- A legal non abstract implementing class has the following properties:
    - It provides concrete implementations for all methods from the interface.
    - It must follow all legal override rules for the methods it implements.
    - It must not declare any new checked exceptions for an implementation method.
    - It must not declare any checked exceptions that are broader than the exceptions declared in the interface method.
    - It may declare runtime exceptions on any interface method implementation regardless of the interface declaration.
    - It must maintain the exact signature and return type of the methods it implements (but does not have to declare the exceptions of the interface).

- A class implementing an interface can itself be abstract.

- An abstract implementing class does not have to implement the interface methods (but the first concrete subclass must).

- A class can extend only one class (no multiple inheritance), but it can implement many.

- Interfaces can extend one or more other interfaces.

- Interfaces cannot extend a class, or implement a class or interface.

- When taking the exam, verify that interface and class declarations are legal before verifying other code logic.