- An `abstract` class cannot be instantiated.
- A single `abstract` method in a class means the whole class must be `abstract`.
- An `abstract` class can have both `abstract` and nonabstract methods.
- The first concrete class to extend an `abstract` class must implement all of its `abstract` methods.

Interface Implementation (OCA Objective 7.6)

- 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.
- Interface methods 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.
- Note: This section uses some concepts that we HAVE NOT yet covered. Don't panic: once you've read through all of Part I of the book, this section will make sense as a reference.

  A legal nonabstract implementing class has the following properties:
  - It provides concrete implementations for the interface's methods.
  - 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 (allowing for covariant returns) 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.
- 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.

Member Access Modifiers (OCA Objective 6.6)

- Methods and instance (nonlocal) variables are known as "members."
- Members can use all four access levels: `public`, `protected`, default, and `private`.
- Member access comes in two forms:
  - Code in one class can access a member of another class.
  - A subclass can inherit a member of its superclass.
- If a class cannot be accessed, its members cannot be accessed.
- Determine class visibility before determining member visibility.
- `public` members can be accessed by all other classes, even in other packages.
- If a superclass member is `public`, the subclass inherits it—regardless of package.
- Members accessed without the dot operator (`.`) must belong to the same class.
- `this.` always refers to the currently executing object.
- `this.aMethod()` is the same as just invoking `aMethod()`.
- `private` members can be accessed only by code in the same class.
- `private` members are not visible to subclasses, so `private` members cannot be inherited.