- □ Default and `protected` members differ only when subclasses are involved:
  - □ Default members can be accessed only by classes in the same package.
  - □ `protected` members can be accessed by other classes in the same package, plus subclasses regardless of package.
  - □ `protected` = package + kids (kids meaning subclasses).
  - □ For subclasses outside the package, the `protected` member can be accessed only through inheritance; a subclass outside the package cannot access a `protected` member by using a reference to a superclass instance. (In other words, inheritance is the only mechanism for a subclass outside the package to access a `protected` member of its superclass.)
  - □ A `protected` member inherited by a subclass from another package is not accessible to any other class in the subclass package, except for the subclass' own subclasses.

Local Variables (OCA Objective 2.1)

- □ Local (method, automatic, or stack) variable declarations cannot have access modifiers.
- □ `final` is the only modifier available to local variables.
- □ Local variables don't get default values, so they must be initialized before use.

Other Modifiers—Members (OCA Objective 6.6)

- □ `final` methods cannot be overridden in a subclass.
- □ `abstract` methods are declared with a signature, a return type, and an optional `throws` clause, but they are not implemented.
- □ `abstract` methods end in a semicolon—no curly braces.
- □ Three ways to spot a nonabstract method:
  - □ The method is not marked `abstract`.
  - □ The method has curly braces.
  - □ The method MIGHT have code between the curly braces.
- □ The first nonabstract (concrete) class to extend an `abstract` class must implement all of the `abstract` class' `abstract` methods.
- □ The `synchronized` modifier applies only to methods and code blocks.
- □ `synchronized` methods can have any access control and can also be marked `final`.
- □ `abstract` methods must be implemented by a subclass, so they must be inheritable. For that reason:
  - □ `abstract` methods cannot be `private`.
  - □ `abstract` methods cannot be `final`.
- □ The `native` modifier applies only to methods.
- □ The `strictfp` modifier applies only to classes and methods.

Methods with var-args (OCP Only, OCP Objective 1.3)

- □ As of Java 5, methods can declare a parameter that accepts from zero to many arguments, a so-called var-arg method.
- □ A var-arg parameter is declared with the syntax `type... name`; for instance: `doStuff(int... x) { }`.
- □ A var-arg method can have only one var-arg parameter.
- □ In methods with normal parameters and a var-arg, the var-arg must come last.

Variable Declarations (OCA Objective 2.1)

- □ Instance variables can
  - □ Have any access control
  - □ Be marked `final` or `transient`
- □ Instance variables can't be `abstract`, `synchronized`, `native`, or `strictfp`.
- □ It is legal to declare a local variable with the same name as an instance variable; this is called "shadowing."
- □ `final` variables have the following properties:
  - □ `final` variables cannot be reassigned once assigned a value.
  - □ `final` reference variables cannot refer to a different object once the object has been assigned to the `final` variable.
  - □ `final` variables must be initialized before the constructor completes.
- □ There is no such thing as a `final` object. An object reference marked `final` does NOT mean the object itself can't change.
- □ The `transient` modifier applies only to instance variables.
- □ The `volatile` modifier applies only to instance variables.