Overriding and Overloading (OCA Objective 6.3)

- Methods can be overridden or overloaded; constructors can be overloaded but not overridden.
- With respect to the method it overrides, the overriding method
  - Must have the same argument list
  - Must have the same return type, except that, as of Java 5, the return type can be a subclass, and this is known as a covariant return
  - Must not have a more restrictive access modifier
  - May have a less restrictive access modifier
  - Must not throw new or broader checked exceptions
  - May throw fewer or narrower checked exceptions, or any unchecked exception
- `final` methods cannot be overridden.
- Only inherited methods may be overridden, and remember that private methods are not inherited.
- A subclass uses `super.overriddenMethodName()` to call the superclass version of an overridden method.
- Overloading means reusing a method name but with different arguments.
- Overloaded methods
  - Must have different argument lists
  - May have different return types, if argument lists are also different
  - May have different access modifiers
  - May throw different exceptions
- Methods from a superclass can be overloaded in a subclass.
- Polymorphism applies to overriding, not to overloading.
- Object type (not the reference variable's type) determines which overridden method is used at runtime.
- Reference type determines which overloaded method will be used at compile time.

Reference Variable Casting (OCA Objectives 7.3 and 7.4)

- There are two types of reference variable casting: downcasting and upcasting.
  - **Downcasting** If you have a reference variable that refers to a subtype object, you can assign it to a reference variable of the subtype. You must make an explicit cast to do this, and the result is that you can access the subtype's members with this new reference variable.
  - **Upcasting** You can assign a reference variable to a supertype reference variable explicitly or implicitly. This is an inherently safe operation because the assignment restricts the access capabilities of the new variable.

Implementing an Interface (OCA Objective 7.6)

- When you implement an interface, you are fulfilling its contract.
- You implement an interface by properly and concretely implementing all of the methods defined by the interface.
- A single class can implement many interfaces.

Return Types (OCA Objectives 6.1 and 6.3)

- Overloaded methods can change return types; overridden methods cannot, except in the case of covariant returns.
- Object reference return types can accept `null` as a return value.
- An array is a legal return type, both to declare and return as a value.
- For methods with primitive return types, any value that can be implicitly converted to the return type can be returned.
- Nothing can be returned from a `void`, but you can return nothing. You're allowed to simply say `return` in any method with a `void` return type to bust out of a method early. But you can't return nothing from a method with a non-`void` return type.
- Methods with an object reference return type can return a subtype.
- Methods with an interface return type can return any implementer.