Ch 11 Inheritance

Inheritance allows a new class to extend an existing class. The new class inherits the members of the class it extends.

Definitions: subclass, superclass, inheritance in UML, chain of inheritance, the `Object` class, polymorphism, dynamic binding

Java syntax:

- `extends` - establish inheritance relationship when defining subclass
- `super()` - to call the superclass' ctor
- `final` - method modifier to prevent override by a subclass
- `public`, `private`, `protected` - modifiers to control member access
- `instanceof` - determines whether an object is an instance of a class
- `abstract` - a class or method that must be overridden in a subclass
- `interface` - specifies the behavior of a class, but none of its implementation
- `implements` - specifies an interface to be implemented in a class

Inheritance	**Composition**
"is-a" relationship	"has-a" relationship
Java: `extends` keyword	Java: class member variable

Example: a ladybug is-a insect:
 	
`public class LadyBug`
`  extends Insect {`
`…`
`}`
	Example: a ladybug has-a wing
	
	`public class LadyBug {`
	`  Wing leftWing;`
	`  Wing rightWing;`
	`}`

Superclass constructor (ctor) issues (p. 673):

- Superclass ctor always executes before the subclass
- Can only call superclass ctor, `super()`, from subclass ctor
- If calling superclass ctor, it must be the first statement of subclass ctor
- If no explicit call to superclass ctor, `super()` is called by default
- If superclass has no default ctor, then you must explicitly call a ctor with args from the subclass ctor

Overriding superclass methods:

- Subclass may override superclass methods, just use same method signature
- Superclass may prohibit override with `final` modifier on method
- Method override is difference than overload

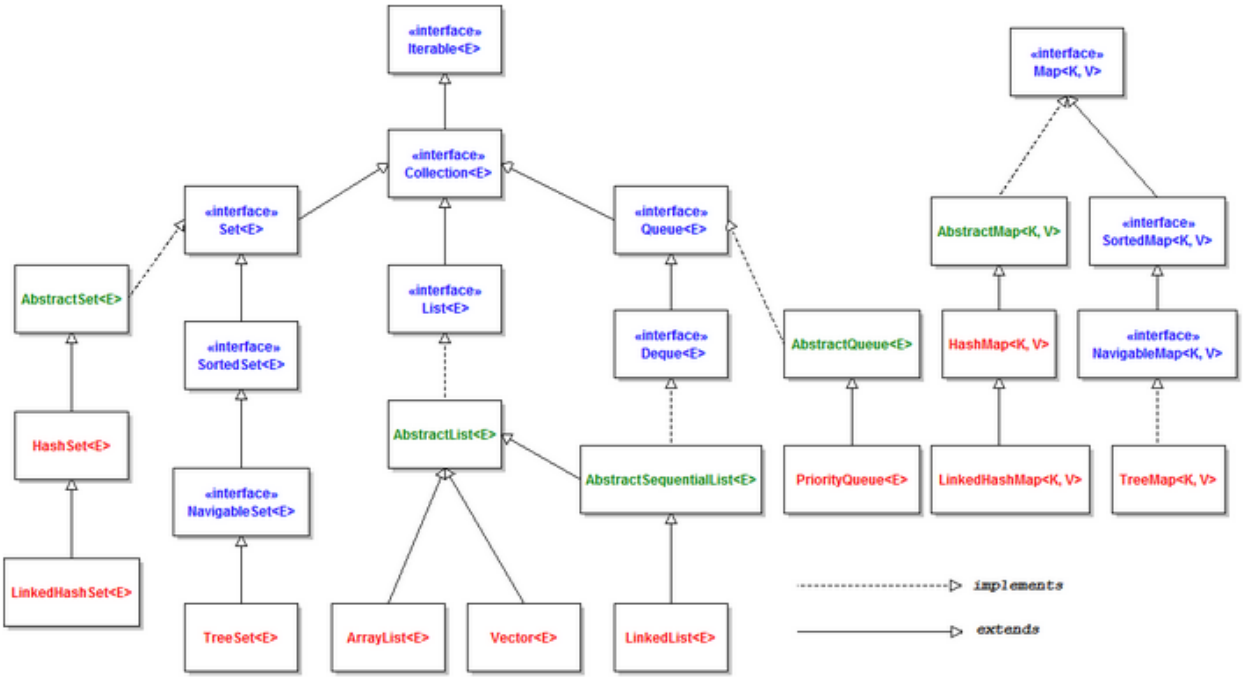
Superclass protection of members:

- ➢ `public` - anyone may access
- ➢ `private` - no one (outside the superclass) may access
- ➢ `protected` - only subclass may access

Interface differences:

- A class can only extend one superclass. A class may implement many interfaces.
- Polymorphism works the same for interfaces as a regular superclass
- You cannot create an instance of an interface

UML - open arrow from subclass to superclass; dashed line means interface
Here's a famous UML: The Java Collection Framework (JCF)



Class diagram of Java Collections framework