

9.8 Wrap-Up

This chapter introduced inheritance—the ability to create classes by acquiring an existing class’s members (without copying and pasting the code) and having the ability to embellish them with new capabilities. You learned the notions of superclasses and subclasses and used keyword `extends` to create a subclass that inherits members from a superclass. We showed how to use the `@Override` annotation to prevent unintended overloading by indicating that a method overrides a superclass method. We introduced the access modifier `protected`; subclass methods can directly access `protected` superclass members. You learned how to use `super` to access overridden superclass members. You also saw how constructors are used in inheritance hierarchies. You learned about the methods of class `Object`, the direct or indirect superclass of all Java classes. Finally, we discussed designing classes with composition vs. inheritance.

In [Chapter 10, Object-Oriented Programming: Polymorphism and Interfaces](#), we build on our discussion of inheritance by introducing *polymorphism*—an object-oriented concept that enables us to write programs that conveniently handle, in a more general and convenient manner, objects of a wide variety of classes related by inheritance, by interfaces or both. After studying [Chapter 10](#), you’ll be familiar with classes, objects, encapsulation, inheritance and polymorphism—the key

technologies of object-oriented programming.