A superclass’s constructor is not inherited; they are invoked explicitly or implicitly.

It is invoked explicitly by using the *super* keyword.

A class is made into a subclass when it extends the superclass.

public class subClass extends superClass {}

If the keyword *super* is not explicitly used, the *superclass’s* default constructor is automatically invoked.

The *super’s* constructor can only be invoked from the subclasses’ constructors.

It must be the first statement in the subclasses constructor.

The keyword *super* can be used to call a superclass’s constructor or method.

A subclass inherits from a superclass.

You can:

* Add new properties
* Add new methods
* Override the methods of the superclass.

A super’s method is overriding when the subclass creates a method with the same name.

public class superclass

{

public String toString()

{}

}

public class superclass

{

@override

public String toString ()

{}

}

Only public methods can be overridden.

A Static method cannot be overridden.

Override vs overloading

* If the variable in both methods are the same, the method is being overridden.
* If they are different variables, the methods are being overloaded.

Casting objects:

Convert an object of one class type to another.

//implicit casting

Object o = new Student ();

This statement is implicit casting because Student is an instance of object.

//Explicit casting

Student b = (Student) Object o;

This statement is explicit because object is not an instance of Student.

We need to tell the compiler that o is a Student object, using an explicit casting.

A subclass cannot weaken the accessibility.

When overriding a method it can be made more visible but not less.

Chapter 13:

Abstract class and methods names are italicized.

An abstract method must be in an abstract class.

A subclass may be abstract even if the superclass is abstract.

A method from the superclass can be overridden in a subclass as abstract.

Interface:

An interface is a class like construct that contains only constants and abstract methods.

It is similar to an abstract class but with the intent to specify common behavior for objects.