# 3. Objects and Classes II :: Nested Classes :: Anonymous Classes

An **anonymous class** is another special type of inner class and is similar to a local class except:

1. An anonymous class does not have a name.
2. The anonymous class and an object of that class are created at the same time.
3. An anonymous class is not actually a class declaration, but rather, is an **expression** which means that it appears inside another expression or where an expression is expected.

Let's take an example.

```
// Super will be the superclass of the anonymous class object created in
// AnonExample.someMethod().
public class Super {
  public Super() { ... }
  public Super(int pInt) { ... }
  public void overriddenMethod() { ... }
}
```

# 3. Objects and Classes II :: Nested Classes :: Anonymous Classes (continued)

```
public class SomeClass {
  // SomeClass instance variable declarations ...
  // SomeClass instance method declarations ...

  // The parameter pObj can be a Super object or an object of any subclass of
  // Super.
  public void doSomething(Super pObj) {
    // Will call Super.overriddenMethod() when pObj is a Super. Will call
    // <anonymous-class>.overriddenMethod() when pObj is an <anonymous-class>.
    pObj.overriddenMethod();
  }
}
```

# 3. Objects and Classes II :: Nested Classes :: Anonymous Classes (continued)

```
public class AnonExample {
  // AnonExample instance variable declarations ...
  // AnonExample instance method declarations ...

  public void someMethod() {
    SomeClass someObject = new SomeClass();

    // We are creating an object of an anonymous inner class which inherits from
    // Super. We call the Super ctor that takes an int argument passing 10.
    someObject.doSomething(new Super() {
      // Anonymous class instance variable declarations ...
      @Override public void overriddenMethod() { ... }
    });

    // More statements of someMethod() ...
  }
}
```

# 3. Objects and Classes II :: Nested Classes :: Anonymous Classes (continued)

Rules for anonymous classes:

1. An anonymous class must extend a superclass or implement an interface.
2. An anonymous class cannot declare a constructor.
3. If an anonymous class extends a superclass, the superclass constructor will be called with the arguments in the parentheses following the superclass name.
4. Like inner classes and local classes, an anonymous class has access to the instance variables and methods of the outer class.
5. Like a local class, an anonymous class has access to the local constants of the enclosing method but not to the local variables.
6. Like inner classes and local classes, an anonymous class may not declare class variables or methods, although it may declare class constants.
7. An anonymous class may contain a local class declaration.