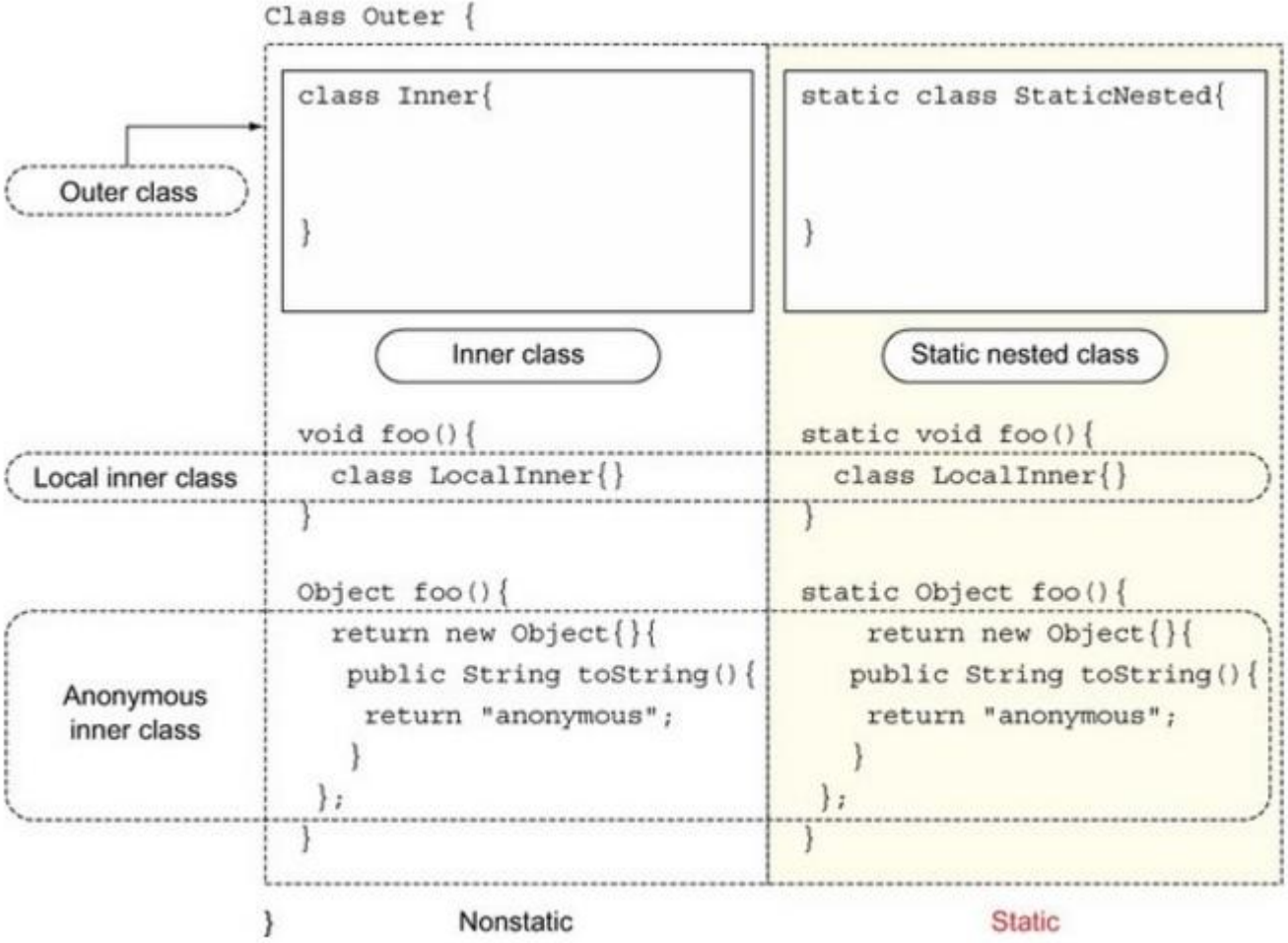
Types of inner classes

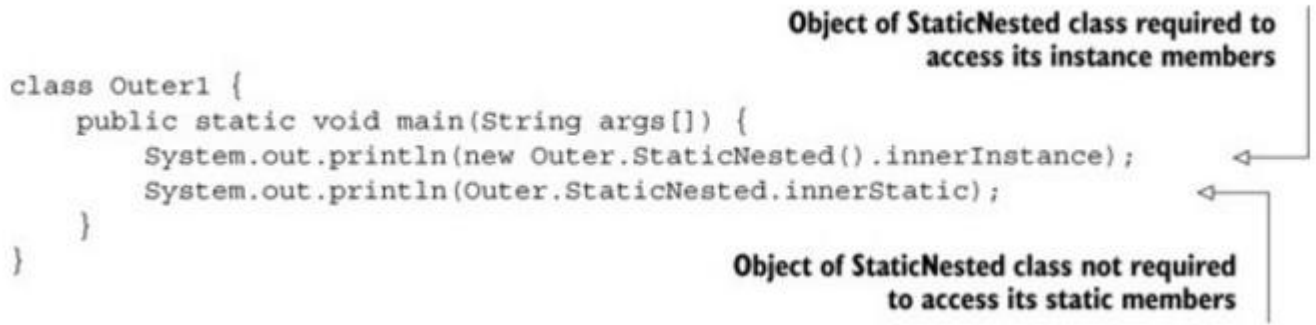

Class Outer {
class Inner{
}
static class StaticNested{
}
Outer class
Inner class
Static nested class
void foo(){
class LocalInner{}
}
static void foo(){
class LocalInner{}
}
Local inner class
Object foo(){
return new Object{}{
public String toString(){
return "anonymous";
}
};
}
static Object foo(){
return new Object{}{
public String toString(){
return "anonymous";
}
};
}
Anonymous inner class
}
Nonstatic
Static

instantiation

```
StaticNested one = new StaticNested();                    ✓
Outer.StaticNested two = new Outer.StaticNested();        ✓
```

```
StaticNested three = new Outer.new StaticNested();        ✗
StaticNested four = new Outer().new StaticNested();       ✗
StaticNested five = Outer.new StaticNested();             ✗
```

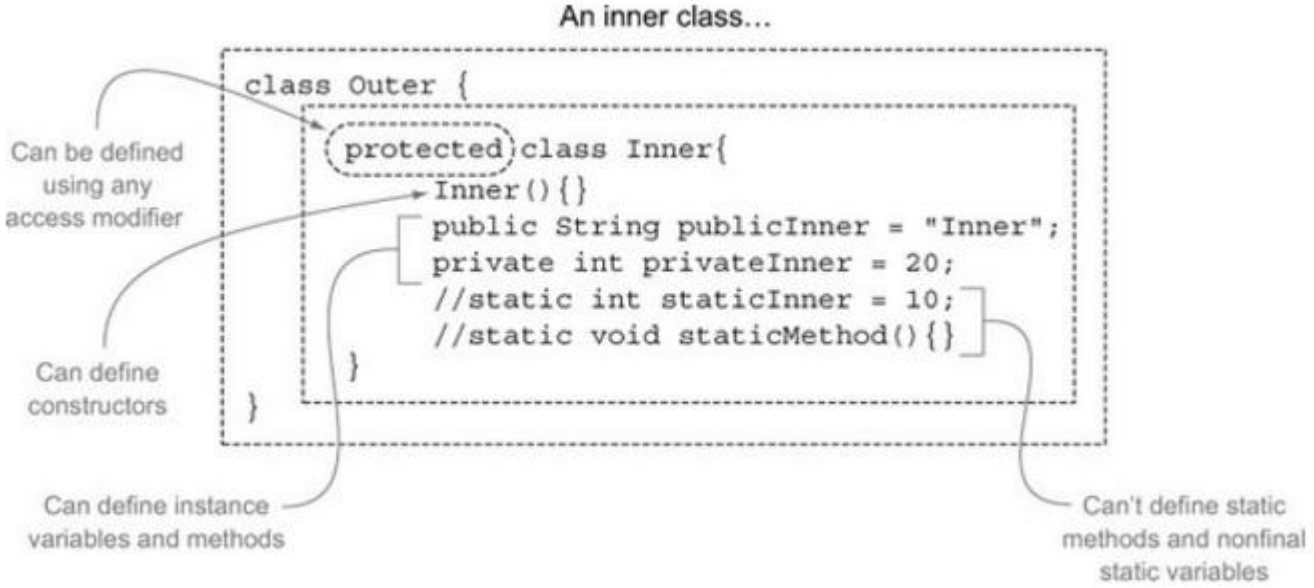
Accession members of static nested class



Rules to remember about static nested classes

- To create an object of a static nested class, you need to prefix its name with the name of its outer class (necessary only if you're outside the outer class).
- A static nested class can define both static and nonstatic members.
- You need not create an object of a static nested class to access its static members. They can be accessed the way static members of a regular class are accessed.
- You should create an object of a static nested class to access its nonstatic members, by using the operator `new`.
- A static nested class can be defined using any access modifier.
- A static nested class can define constructor(s).

Characteristics inner classes



Instantiation inner class

Now, let's try to instantiate class `Inner` within a static method of class `Outer` (additions in bold):

```
class Outer {
    class Inner {}
    static void staticMethod() {
        Inner in = new Inner();
    }
}
```

❶ Won't compile

The code at ❶ doesn't compile because in method `staticMethod()` there's no outer class instance to tie the inner class instance to, which is required for creation of its *inner class* `Inner`. But it's possible to instantiate class `Outer` in the method `staticMethod()`. When you have an `Outer` instance, you can instantiate `Inner`:

```
class Outer {
    class Inner {}
    static void staticMethod () {
        Outer outObj = new Outer();
        Inner innerObj = outObj.new Inner();
    }
}
```

❶ Instance of Outer can be created in method staticMethod()

❷ Instance of Inner is created by calling operator new on Outer instance

If another class wants to create an instance of class `Inner`, it needs an instance of class `Outer`. If it doesn't have one, it should first create one, just like with method `staticMethod()` in the previous code snippet:

```
class Foo {
    Inner inner;
    Foo () {
        Outer outer = new Outer();
        inner = outer.new Inner();
    }
}
```

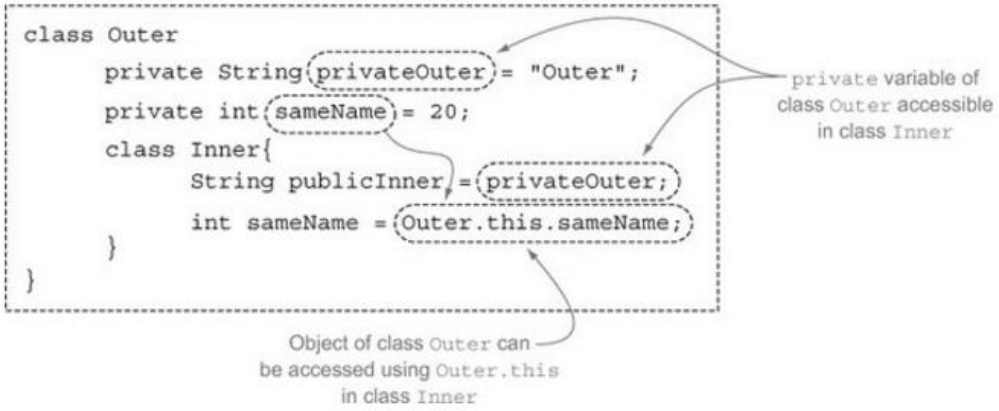
Won't compile

What makes the preceding code fail compilation? `Inner` isn't a top-level class, so its variable type should include the name of its outer class, `Outer`, so class `Foo` can find this class. The following code compiles successfully and creates an `Inner` instance:

```
class Foo {
    Outer.Inner inner;
    Foo () {
        Outer outer = new Outer();
        inner = outer.new Inner();
    }
```

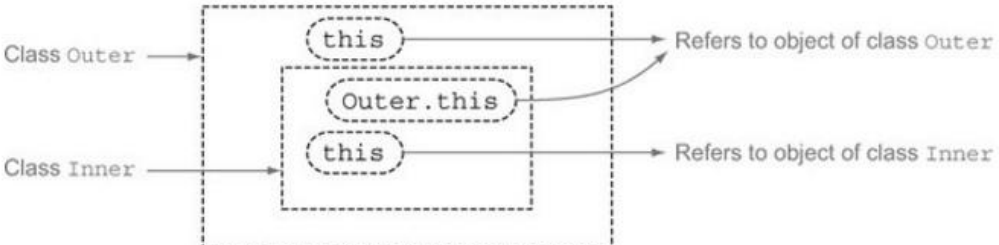
Outside its outer class, the type of inner class should include the name of its outer class.

Figure 2.12. An inner class can access all the members of its outer class, including its private members. Outer class members with the same name as inner class members can be accessed using `Outer.this`, where `Outer` is the name of the outer class.



An object uses the reference `this` to refer to its own object. An inner class can use the reference `this` to refer to its own object, and the name of its outer class followed by `.this` to refer to the object of its outer class, as shown in figure 2.13.

Figure 2.13. An inner class uses this to refer to its own object and `<name_of_its_outer_class>.this` to refer to its outer class's object.



Rules to remember about inner classes

- You can create an object of the inner class within an outer class or outside an outer class.
- When an inner class is created outside its outer class, its type name should include the name of its outer class, followed by a dot (.) and then the name of the inner class.
- To create an inner class with a static method of an outer class, or outside an outer class, call the operator `new` on the object of the outer class to instantiate the inner class.
- An inner class can't define static methods. It can define final static variables but nonfinal static variables aren't allowed.
- Members of the inner class can refer to all variables and methods of the outer class.
- An inner class can be defined with all access modifiers.
- An inner class can define constructors.
- An inner class can define variables and methods with any access level.