

## ***Special Topics No. 3: Instantiable Main Classes***

---

We describe a simple technique to make any instantiable class the main class of a program. Allowing an instantiable class to be the main class of a program is a handy way to execute any class without defining a separate main class. Such a technique is useful, for example, in providing a demo usage of the instantiable classes you define. Since the use of this technique is very common among Java programmers, it is beneficial to know the technique even if you don't use it yourself.

## Introduction

In this document, we will show you a simple way to make any instantiable class also the main class of a program. Instead of defining a separate main class, as we have been doing so far, it is possible to define the main method to an instantiable class so the class becomes the main class of a program also. There are a number of advantages in doing this. First, you have one less class to manage if you don't have to define a separate main class. This advantage may not seem so substantial. Indeed, because a logical simplicity to beginners and the burden of writing a separate main class are not that huge, we opt to teach this approach in the textbook. However, when you write numerous classes (for example, writing solutions to the chapter exercises), writing a separate main class for all those classes so they become executable becomes tedious. What is acceptable to beginning programmers may not be so accommodating to advanced or professional programmers.

Second, when you develop instantiable and reusable classes for other programmers, you almost always want to include a simple example on how to use the classes. Instead of providing a separate sample main class, it is more convenient to make the instantiable classes also the main class. We will teach you how to do this in this document.

## 1 Using a Separate Main Class

---

Let's first write a trivial sample program in the textbook way of defining a separate main class. The sample program asks for the user's name and replies back with a personalized greeting. The program includes the two classes—SampleMain and Sample (which, by now, require no explanation for you to understand):

File: [SampleMain.java](#)

```
//Sample Main Program
class SampleMain
{
    public static void main( String[] arg)
    {
        Sample sample;
        sample = new Sample( );
        sample.start( );
    }
}
```

File: [Sample.java](#)

```
//A trivial sample instantiable class
class Sample
{
    private MainWindow mainWindow;
    private InputBox  inputBox;
    private MessageBox messageBox;

    public Sample( )
    {
        mainWindow = new MainWindow( );
        inputBox   = new InputBox( mainWindow );
        messageBox = new MessageBox( mainWindow );

        mainWindow.setVisible( true );
    }

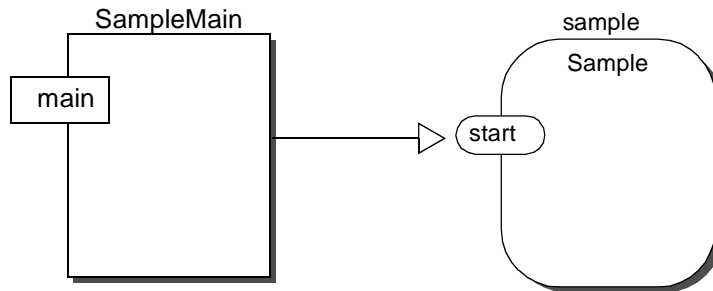
    public void start( )
    {
        //get the user's name
        String name = inputBox.getString( "What's your name?" );

        messageBox.show( "Hi, " + name + ". Nice to meet you." );
    }
}
```

Instantiable Main Class

Figure 1 is the program's object diagram.

**FIGURE 1** The object diagram for the **Sample** program.



## 2 Making Instantiable Classes the Main Class

Now let's rewrite the instantiable *Sample* class so that it also can be the main class. Remember from Chapter 2 that a Java program requires one of its classes to be designated as the main class, the class that includes the main method.

When we write a Java program, we have been defining a class that includes only the main method and we designate this class as the main class. Since the class includes only the main method, which is a static method, the main classes we have written so far are all noninstantiable classes. This, however, is not a requirement. Any class can be the main class; whether it is instantiable or noninstantiable is immaterial. All that is required for a class to be the main class is to include the main method.

Therefore, to make the instantiable Sample class also the main class of a program, all we have to do is simply add the main method to it. Here's how:

File: [Sample.java](#)

```
import javabook.*;
//A trivial sample instantiable class that is also the main class

class Sample
{
    private MainWindow mainWindow;
    private InputBox  inputBox;
    private MessageBox messageBox;

    public Sample( )
    {
        mainWindow = new MainWindow( );
        inputBox   = new InputBox( mainWindow );
        messageBox = new MessageBox( mainWindow );

        mainWindow.setVisible( true );
    }

    public void start( )
    {
        //get the user's name
        String name = inputBox.getString( "What is your name?" );

        messageBox.show( "Hi, " + name + ". Nice to meet you." );
    }

    public static void main( String[] arg)
    {
        Sample sample;
        sample = new Sample( );
        sample.start( );
    }
}
```

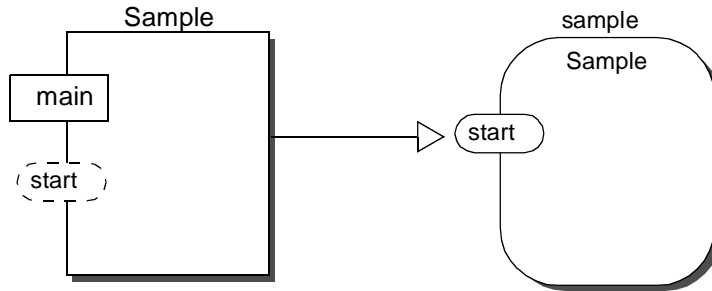
The class now  
includes the  
**main** method

Figure 2 is the object diagram for a program with only the Sample class. After the class is compiled correctly, you can execute it by entering the Java interpreter command as

```
java Sample
```

**FIGURE 2**

The object diagram for the new **Sample** program. Notice that there is only one class and one instance of that class.



If you are using any one of the Java IDEs, e.g., JBuilder and Visual Cafe, then you simply designate the **Sample** class as the main class of the program in whatever way the IDE requires you to do.

We stated earlier that for a program to be executable, one of its classes must be designated as the main class, and the class designated as the main class must include the main method. However, this does not mean that only one of the classes can include the main method. It does not cause any problem when more than one class has the main method, so long as the designated class has the main method. This means that the new **Sample** class, the one with the main method, and the **SampleMain** class do not conflict. You can still set the *SampleMain* as the main class and run the program. The implication is that you can include the main method in the instantiable classes you define so the programmers can run the classes with or without defining their main class. The programmers can choose whichever way is convenient to them.



***A Java program can include more than one class with the main method. The only restriction is that the class designated as the main class has the main method.***