WORKED EXAMPLE 3.1 Making a Simple Menu

Problem Statement Your task is to design a class `Menu`. An object of this class can display a menu such as

```
1) Open new account
2) Log into existing account
3) Help
4) Quit
```

The numbers should be supplied automatically when options are added to the menu.

© Mark Evans/iStockphoto.

Step 1 Find out which methods you are asked to supply.

The problem description lists two tasks:

Display the menu.
Add an option to the menu.

Step 2 Specify the public interface.

Here we turn the list in Step 1 into a set of methods, with specific types for the parameter variables and the return values. As recommended in How To 3.1, we start by writing out sample code:

```
mainMenu.addOption("Open new account");
mainMenu.addOption("Log into existing account");
mainMenu.display();
```

Now we have a specific list of methods:

```
public void addOption(String option)
public void display()
```

To complete the public interface, we need to specify the constructors. We have two choices:

- Supply a constructor `Menu(String firstOption)` that makes a menu with one option.
- Supply a constructor `Menu()` that makes a menu with no options.

Either choice will work fine. If we decide in favor of the second choice, the user of the class needs to call `addOption` to add the first option—after all, there is no sense in having a menu with no options. At first glance, that seems like a burden for the programmer using the class. But actually, it is usually conceptually simpler if an API has no special cases (such as having to supply the first option in the constructor). Therefore, we decide that "simplest is best" and specify the constructor

```
public Menu()
```

Step 3 Document the public interface.

Here is the documentation, with comments, that describes the class and its methods:

```
/**
   A menu that is displayed on a console.
*/
public class Menu
{
   /**
      Constructs a menu with no options.
   */
   public Menu()
   {
   }
```

```
   /**
      Adds an option to the end of this menu.
      @param option the option to add
   */
   public void addOption(String option)
   {
   }

   /**
      Displays the menu on the console.
   */
   public void display()
   {
   }
}
```

Step 4 Determine instance variables.

What data does a `Menu` option need to keep in order to fulfill its responsibilities? Of course, in order to display the menu, it needs to store the menu text. Now consider the `addOption` method. That method adds a number and the option to the menu text. Where does the number come from? The menu object needs to store it too, so that it can increment whenever `addOption` is called.

Therefore, our instance variables are

```
public class Menu
{
   private String menuText;
   private int optionCount;
   . . .
}
```

Step 5 Implement constructors and methods.

We now implement the constructors and methods in the class, one at a time, in the order that is most convenient. The constructor seems pretty easy:

```
public Menu()
{
   menuText = "";
   optionCount = 0;
}
```

The `display` method is easy as well:

```
public void display()
{
   System.out.println(menuText);
}
```

The `addOption` method requires a bit more thought. Here is the pseudocode:

```
Increment the option count.
Add the following to the menu text:
   The option count
   A ) symbol
   The option to be added
   A "newline" character that causes the next option to appear on a new line
```

How do you add something to a string? If you look at the API of the `String` class, you will find a method `concat`. For example, the call

```
menuText.concat(option)
```

creates a string consisting of the strings `menuText` and `option`. You can then store that string back into the `menuText` variable:

```
menuText = menuText.concat(option);
```

As you will learn in Chapter 4, you can achieve the same effect with the `+` operator:

```
menuText = menuText + option;
```

We use the `+` operator in our solution because it is so convenient. Our method then becomes

```
public void addOption(String option)
{
   optionCount = optionCount + 1;
   menuText = menuText + optionCount + ") " + option + "\n";
}
```

Step 6 Test your class.

Here is a short program that demonstrates all methods in the public interface of the `Menu` class:

```
public class MenuDemo
{
   public static void main(String[] args)
   {
      Menu mainMenu = new Menu();
      mainMenu.addOption("Open new account");
      mainMenu.addOption("Log into existing account");
      mainMenu.addOption("Help");
      mainMenu.addOption("Quit");
      mainMenu.display();
   }
}
```

Program Run

```
1) Open new account
2) Log into existing account
3) Help
4) Quit
```