

Chapter 9 Assignment

The purpose of this assignment is to give you practice creating classes and objects as well as additional experience working with arrays. You have one program to create, the specifications for which are described below. This assignment is worth 100 points.

Specifications

In this assignment, you will create an application that keeps track of a store's inventory of products as well as allow product purchases. This application will be written using the object-oriented coding paradigm, so we will build a class as well as a program, each of which will reside in its own file. Therefore, your solution this week will consist of two files, both of which you'll need to submit together in a single zip file. So let's get started.

The Product Class

Create a file named `Product.rb` that contains a `Product` class. The class should have the following attributes:

<code>ProductNumber</code>	A string used to hold a product number. Each product in the inventory should have a unique product number.
<code>Description</code>	A string that holds a brief description of the item.
<code>Cost</code>	A decimal value that holds the amount the retail store paid for the item.
<code>Retail</code>	A decimal value that holds the retail price for the item.
<code>QuantityOnHand</code>	An integer value that holds the number of units in stock for the product. This value cannot be less than zero.

Your `Product` class also needs a method to display all of the aforementioned attributes. Create a method named `display()` that prints out the values in an attractive way. One approach you could take is to display the data in two columns – the first to display the name of the attribute, and the second to display the corresponding value.

Make sure you have a comment block at the top of the `Product.rb` file that describes the class and provides a description of each attribute and method.

The Program

Create a file named `Ch9Asg.rb` and put it in the same folder. Put a comment block at the top that lists the appropriate information. Since we will be creating `Product` objects in the program, you need to make this file aware of the `Product.rb` file. To do that, place the following line of code directly underneath the comment block:

```
require_relative 'Product'
```

Your program should consist of a central loop that displays a menu, gets the user's selection, carries out the selection, and returns back to the menu. The menu should list the following options:

1. Create a new product
2. Look up a product
3. Process a product purchase
4. Exit

Your program needs an array to hold the `Product` objects the user will create with menu option 1 in order to access those objects in menu options 2 and 3. Here's an overview of how each menu item should work:

Create a New Product

The user should be prompted for all values that make up a product. All appropriate values should be validated as they are entered. Make sure the product number the user supplies does not already exist in the list of existing products. Once all of the data are supplied and are valid, create a new `Product` object, passing the data to it when created. Then, add the `Product` object to the array that holds them.

Look Up a Product

The user should be prompted for a product number. The program will then attempt to locate the product in the array. If it finds a `Product` with a matching number, the product will be described. Call the `display()` method of the appropriate `Product` object to do so. If a product with the number the user entered cannot be found, inform the user and return back to the menu.

Process a Product Purchase

Display the product number, description, retail price, and number of units on hand for all products whose `QuantityOnHand` is greater than zero. The user should then be prompted for a product number from that list. If the user enters a valid product number, the user should be asked how many to purchase. The quantity to be purchased cannot exceed the number of units on hand. If the quantity is valid, the application should display the subtotal, sales tax amount (6%), and the grand total for the sale. Don't forget to update the product's `QuantityOnHand` by subtracting the number purchased.