C# OOP Assignment – Computer Shop

Deadline: Tuesday, September 8th 2020, 09:00AM

GitHub Classroom Repository

Introduction

This practice is designed to help you familiarize yourself with class and object creation.

Requirements **Names in "Quotes" are not concepts, but rather arbitrary names for the purpose of the task.**

- Create a "CPU" class with the following:
  - A "Brand" property
  - A "Speed" property (double).
- Create an abstract "Device" class with the following:
  - A "CPU" property (reference to "CPU" object)
  - A "MemoryBank" property (a list of "Memory" objects)
  - A "Connectors" property as a read-only dictionary.
    - The keys will come from the Connector enumeration (enum = enumeration).
    - The values will be ints to represent the number of connections available.
  - A private "Peripherals" property (a polymorphic list of Peripheral objects)
  - A "ConnectPeripheral()" method that accepts a polymorphic peripheral argument and:
    - Throws an exception if there are no available connectors of the connection-type of the peripheral.
      - Unless the type is "Integrated".
    - Otherwise, adds the peripheral to the list.
  - A "DisconnectPeripheral()" method that accepts a type of peripheral, and will disconnect all connected peripherals of that type.
  - An abstract "StartUp()" method that will be overridden in each derived class, with some functionality of your choice. Write something to the console, make the computer beep, whatever you'd like.
  - A "Brand" property (string).
- Create a "Memory" class with the following:
  - A "Brand" property (string).
  - A "Size" property (int).
- Create a "Laptop" class derived from "Device" with the following:
  - A "Screen" property that must have a "Screen" object with the "Integrated" connector type assigned.
  - A "Keyboard" property that must have a "Keyboard" object with the "Integrated" connector type assigned.
  - A default and greedy constructor.
- Create a "Desktop" class derived from "Device" with the following:
  - A default and greedy constructor.
- Create a "CellPhone" class derived from "Device" with the following:
  - A "Screen" property that must have a "Screen" object with the "Integrated" connector type assigned.
  - A default and greedy constructor.
- Create a public "Connector" enumeration of:
  - "USBTypeA"

    - “USBTypeB”
    - “USBTypeC”
    - “MiniUSB”
    - “MicroUSB”
    - “DisplayPort”
    - “Integrated”
- Create an abstract “Peripheral” class with the following:
    - A “Brand” property (string).
- Create a “Keyboard” class derived from “Peripheral” with the following:
    - A “Type” enumeration of: “Mechanical” or “Membrane”.
    - A “ConnectorType” property with a value of the “Connector” enumeration.
    - A default and greedy constructor.
- Create a “Mouse” class derived from “Peripheral” with the following:
    - A “ButtonCount” property (int).
    - A “ConnectorType” with a value of the “Connector” enumeration.
    - A default and greedy constructor.
- Create a “Screen” class derived from “Peripheral” with the following:
    - A “Width” property, in whole pixels (int).
    - A “Height” property, in whole pixels (int).
    - A “ConnectorType” with a value of the “Connector” enumeration.
    - A default and greedy constructor.

Challenges

- Implement File IO that will allow a list of devices to be stored in a file and read back in.
- Implement ToString() overrides into all classes to make a descriptive plain-English output string.
- Add more types of peripherals and connectors to build a full virtual computer (speakers, headphones, webcam, etc).

Hints

- Create your dictionary in the constructor as a normal dictionary, then feed it into the constructor of ReadOnlyDictionary.
- For the “ConnectPeripheral()” method, the Aug31Practice branch has an example of adding to a polymorphic list. It will require some validation as well.
- For the “RemovePeripheral()” method, you could try using a Where() with a clause checking for type not equal to the type to remove (see Sept03Practice commit “SpaceCovered and LinesDrawn;”, Drawing.cs).

Reminders

Ensure you are tracking your time and that your timesheet is in the appropriate folder for viewing and marking.

Ensure you are committing frequently. It is advised to commit once per successful feature implementation at a minimum.

Ensure you are pushing your repository to the GitHub Classroom repository, and not a personal, private repository.

Ensure your repository has a readme with at a minimum your name, the name of the project, the project’s purpose and a link to your Trello board.

Ensure you are using Trello appropriately to keep track of outstanding features, and that it is linked in the README.md file.

Ensure you are using the #class channel to request clarifications on assignment specifications.

Ensure you are using the #homework-help channel in slack to request for assistance from instructional staff if needed, and that you include (at a minimum) a specific description of the problem and a list of what you have tried.

Feel free to reach out on #peer-support if the #homework-help queue is lengthy. If someone helps you out, give them a shoutout using our handy-dandy form!

Citation Guide for Borrowed Code

Whenever you borrow code, the following information must be included:

- Comments to indicate both where the borrowed code begins and ends.
- A source linking to where you found the code (URL, book, example, etc.).
- Your reason for adding the code to your assignment or project instead of writing it out yourself.
- Explain to us how the code is supposed to work, include links to documentation and articles you read to help you understand.
- A small demonstration to prove you understand how the code works.

```
const inputArr = [5,1,3,4,2];

/*Borrowed code for bubbleSort starts*/
let bubbleSort = (inputArr) => {
    let len = inputArr.length;
    for (let i = 0; i < len; i++) {
        for (let j = 0; j < len; j++) {
            if (inputArr[j] > inputArr[j + 1]) {
                let tmp = inputArr[j];
                inputArr[j] = inputArr[j + 1];
                inputArr[j + 1] = tmp;
            }
        }
    }
    return inputArr;
};

/*Borrowed code from bubbleSort ends*/

//Source: bubbleSort function obtained from https://medium.com/javascript-algorithms/javascript-algorithms-bubble-sort-3d27f285c3b2
//Reason to add: implementing bubble sort can be tedious and bug prone, it would be better to use a proven version than to write my own
//How it works: I read the following article to understand how bubble sorts work (http://www.pkirs.utep.edu/CIS3355/Tutorials/chapter9/tutorial9A/bubblesort.htm)
//Demonstration of understanding:
//Example array: [3,1,2]
//Step 1: Compare 3 and 1. Since 1 is smaller, swap places.
//Array: [1,3,2]
//Step 2: Compare 3 and 2. Since 2 is smaller, swap places.
//Array: [1,2,3]
//Step 3: Compare 1 and 2. No need to swap.
//Array: [1,2,3]
//Step 4: Compare 2 and 3. No need to swap.
//Array: [1,2,3]
//Function complete.
console.log(bubbleSort(inputArr));
```