OOP – Multimedia Shop

The goal of this lab is to practice **Object-oriented programming** by building a Multimedia Shop System for managing different items – movies, books and games. The items can be **sold** or **rented**.

Problem 5. Shop Engine

Now that we have our items, rents and sales, it's time to write our **ShopEngine**. The engine will receive commands from the console and execute them. The possible commands are:

First, let's define what our engine will do:

- Supplying the store with a given quantity of items
- Selling an item in supply
- Renting an item in supply
- Reporting sales/rents

The following commands should be supported:

- **supply [type] [quantity] [params]** – adds **[quantity]** items of **[type]** to the supplies. **[params]** is a string in the format **key1=value1&key2=value2&key3=value3**, where key-value pairs are separated by **&**.
- **sell [id] [saleDate]** – sells an item with the specified **[Id]** on **[saleDate]**.
- **rent [id] [rentDate] [deadline]** – rents an item with the specified **[id]**, **[rentDate]** and **[deadline]**.
- **report sales [startDate]** – prints the sum of all sales going back to **[startDate]**.
- **report rents** – prints all **overdue rents**, ordered by their **rent fine** in ascending order (then by **title** as secondary criteria).

Step 1 – Supplying Books

Let's implement supplying books.

We need to keep the current item supplies somewhere – in some data structure. Define a **Dictionary<key, value>** where the **key** is the item in stock (**accessed through its interface**), and **value** is the quantity. Will this dictionary be used outside ShopEngine? What access level should it have?

We also need to create some sort of loop that **reads input on each iteration** and executes the read **command**. Create a method **Run()** in our **ShopEngine** – when that method is called, an infinite loop should start reading input commands.

```
public void Run()
{
    while (true)
    {
        string command = Console.ReadLine();
        // TODO: Execute command
    }
}
```

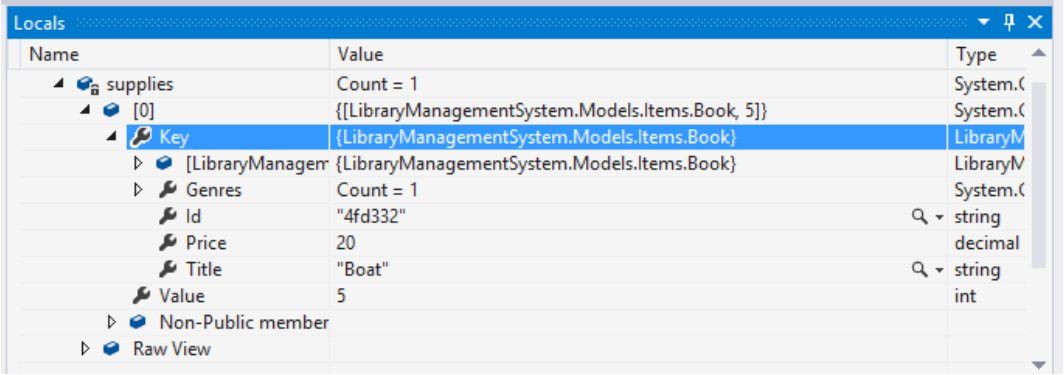
Suppose we have the following input line:

- supply book 5 id=4fd332&title=Boat&price=20&author=Sellinger&genre=comedy

Our engine should parse this line and add to the supplies a **Book item with Id "4fd332", title "Boat", price 20.00, author "sellinger" and genre "comedy"** with quantity **5**.

Note: Parsing the line and adding the item to the supply should be done in **separate methods**!

After executing this command, our **supplies** should look as follows:



Step 2 – Supplying Games and Videos

Implement the supply command for **games** and **videos** as well.

- supply game 4 id=sfd33&title=Assassin's_Creed&price=19.00&genre=fiction&ageRestriction=Teen
- supply video 40 id=sfd332&title=The_Godfather&price=79.00&genre=crime&legth=187

Separate different tasks into methods. Make sure you do not expose any unnecessary members outside the **ShopEngine** class.

Hint:

```
// Export to method!!!
Dictionary<string, string> keyValuePairs = new Dictionary<string, string>();
string[] pairs = paramsString.Split('&');

foreach (var pair in pairs)
{
        string[] keyValuePair = pair.Split('=');
        keyValuePairs[keyValuePair[0]] = keyValuePair[1];
}
```