Assignment 3 – BCS 426 C# Programming

Due: 3/9/2020 @ 1:40pm

You will need to update the existing Visual Studio solutions from the previous assignment. When you submit the assignment in the Blackboard dropbox make sure you zip both solutions and submit them.

IMPORTANT – Make sure you ***properly comment*** AND ***properly indent*** your program. The commenting and indenting documents are on Blackboard in the "Handouts" folder. ***If you fail to properly comment or properly indent I will deduct points.***

VERY IMPORTANT – IF THE PROGRAM DOES NOT COMPILE THERE WILL BE MAJOR POINTS TAKEN OFF.

Overview

Create the Department class to store worker and shift data. Create a menu-driven console application that will be able to serialize and deserialize Department. The application will also have search functionality. Your application must be able to correctly use the JSON input file that you are given in the Homework folder on Blackboard.

Part 0 – Regions

All code that you write should be placed inside of appropriate regions similar to what you did in assignment 1.

Part 1 – Update the DLL

Add the following classes to the existing DLL solution from the previous assignment.

Class – Department

Member Variables (all private)

name (string), workers(List<Worker>), shifts(List<Shift>)

Member Method Signatures and Descriptions (all public)

Signature	**Description**
Department()	Default constructor. Sets the values of each member variable to a default value.
C# properties	Write properties for each member variable.
Worker FindWorker(int workerId)	This method takes a worker's id as a parameter. It should return the Worker with the given id. If it is not found then return null.
double CalculatePay(int workerId)	Calculates a worker's pay. The worker's pay is the total hours worked for all of their shifts times their pay rate.
String ToString()	This method should return a string that contains data for all member variables. This means that the department name as well as all data in each collection should be part of the string that is returned.

Serialization

Add the necessary code to make this class serializable.

Testing Project

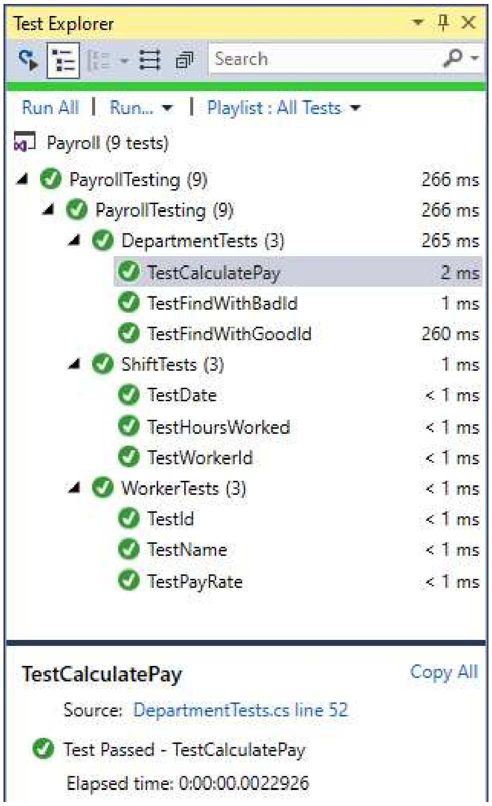
Add a new testing class to the existing Testing project in the DLL. Each of these testing methods should create a department instance with data inside of it. The best way to do this would be to just deserialize from a department JSON file. Be advised that you need to add DLL references to the testing project for System.Runtime.Serialization and System.Xml.

- Class - DepartmentTests

- TestFindWithGoodId – Should pass a VALID ID into Find and it should return a worker instance with the target id.
- TestFindWithBadId – Should pass an INVALID ID into Find and it should return null.
- TestCalculatePay – Should pass a VALID ID into CalculatePay and it should return the correct pay amount. You can figure out what the correct result should be by looking at the department input file.

You should use Assert methods within your testing methods to make sure the results are correct. Check at the end of the unit testing slides for an example.

Here is a screenshot of executing all the tests:



```
Payroll (9 tests)
PayrollTesting (9)                266 ms
  PayrollTesting (9)              266 ms
    DepartmentTests (3)           265 ms
      TestCalculatePay              2 ms
      TestFindWithBadId             1 ms
      TestFindWithGoodId          260 ms
    ShiftTests (3)                  1 ms
      TestDate                    < 1 ms
      TestHoursWorked             < 1 ms
      TestWorkerId                < 1 ms
    WorkerTests (3)               < 1 ms
      TestId                      < 1 ms
      TestName                    < 1 ms
      TestPayRate                 < 1 ms

TestCalculatePay
  Source: DepartmentTests.cs line 52
  Test Passed - TestCalculatePay
  Elapsed time: 0:00:00.0022926
```

Part 2 – Main Project

Create a console application project in Visual Studio. It will contain a menu-driven program to manipulate a Department object. This project should import your DLL solution. DO NOT COPY THE DEFINITION OF ANY OF THE CLASSES INTO THE CONSOLE APPLICATION, IMPORT THE DLL SOLUTION!!!

Menu Description

This program will present a menu to the user and then perform an action depending on what the user chooses to do. You should create one instance of Department at the TOP of the main method. When the program runs it should display the menu to the user and give them a chance to input a choice. An action should be taken depending on what choice the user makes. The menu actions should manipulate and use the Department instance that you declared at the top of main.

You should not hardcode filename strings into the program. If the user chooses to read or write data to a file you should first ask the user to enter that filename then use that file for input/output as appropriate.

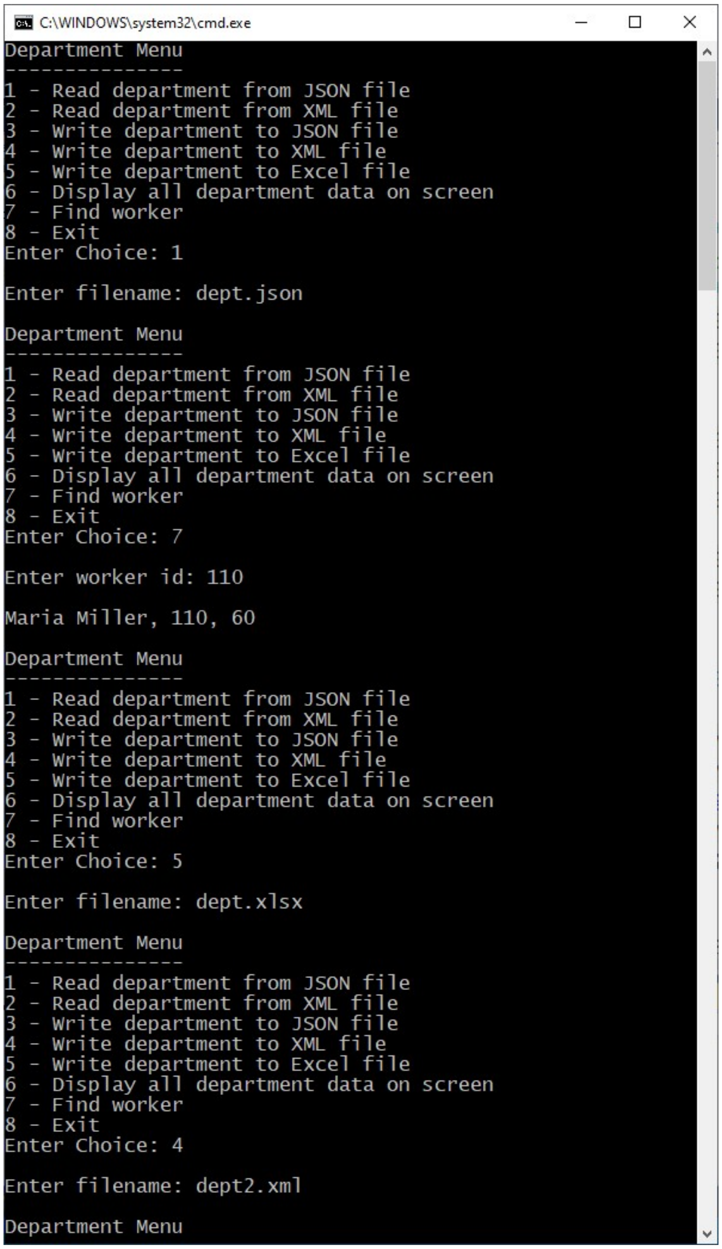
Option 6 should first ask the user to enter an assignment name to search for. Your program should search the submissions using the assignment name the user entered. Once you get the target submission you can then display the submission on screen.

Here is the menu:

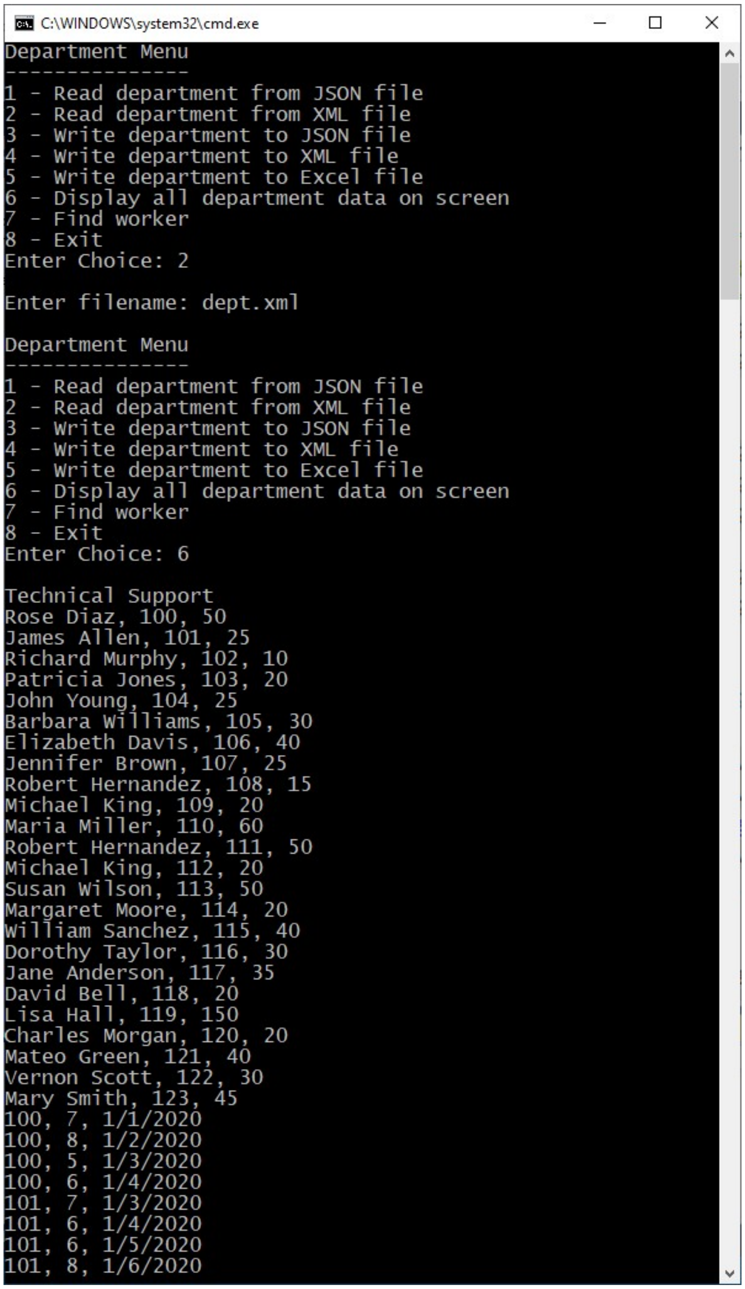
```
Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 – Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice:
```

THE PROGRAM SHOULD KEEP SHOWING THE MENU AND PERFORMING AN ACTION UNTIL THE USER CHOOSES TO EXIT. THERE IS A SCREEN SHOT BELOW.

Here are sample program executions:

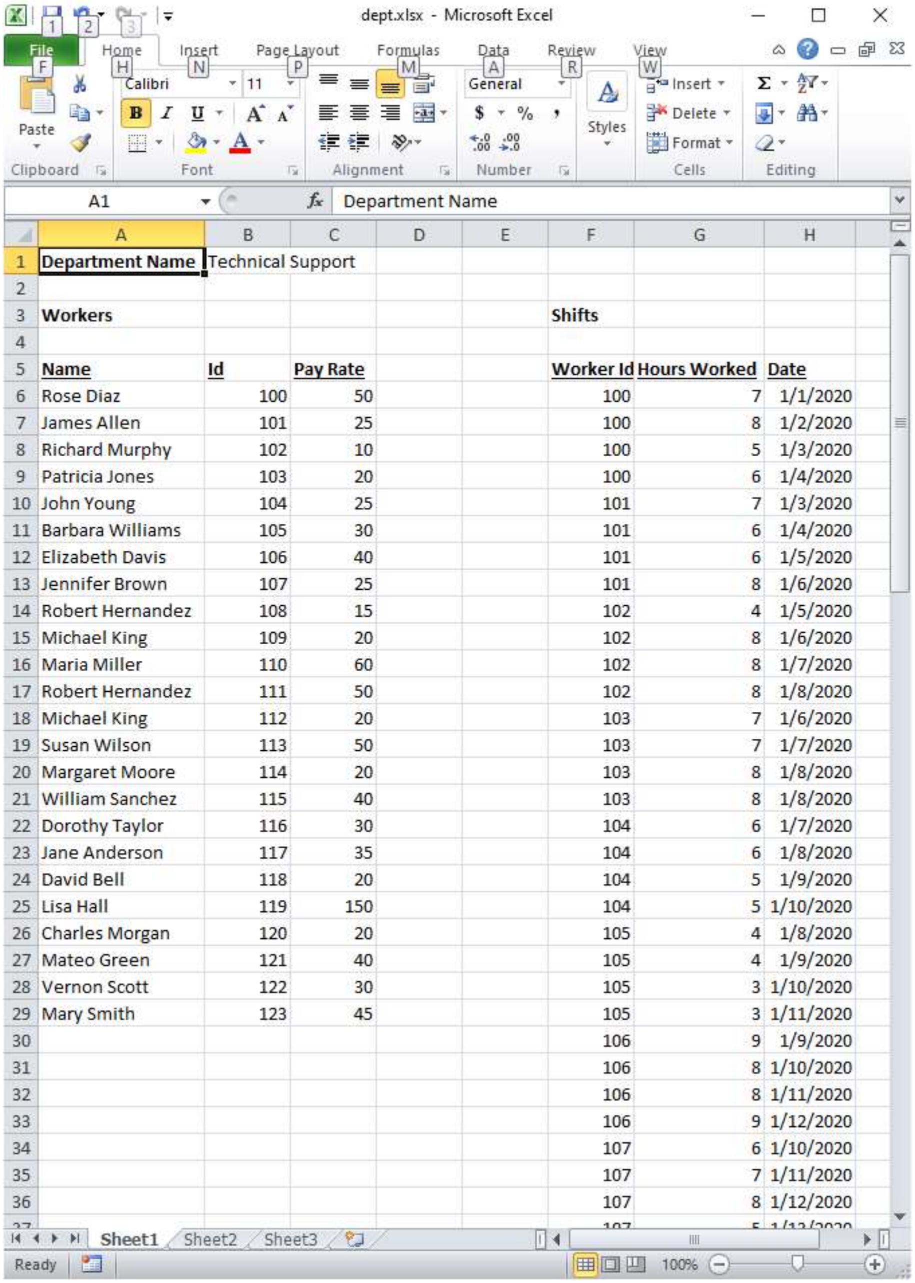


```
Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 - Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice: 1

Enter filename: dept.json

Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 - Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice: 7

Enter worker id: 110

Maria Miller, 110, 60

Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 - Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice: 5

Enter filename: dept.xlsx

Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 - Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice: 4

Enter filename: dept2.xml

Department Menu
```



```
Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 - Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice: 2

Enter filename: dept.xml

Department Menu
---------------
1 - Read department from JSON file
2 - Read department from XML file
3 - Write department to JSON file
4 - Write department to XML file
5 - Write department to Excel file
6 - Display all department data on screen
7 - Find worker
8 - Exit
Enter Choice: 6

Technical Support
Rose Diaz, 100, 50
James Allen, 101, 25
Richard Murphy, 102, 10
Patricia Jones, 103, 20
John Young, 104, 25
Barbara Williams, 105, 30
Elizabeth Davis, 106, 40
Jennifer Brown, 107, 25
Robert Hernandez, 108, 15
Michael King, 109, 20
Maria Miller, 110, 60
Robert Hernandez, 111, 50
Michael King, 112, 20
Susan Wilson, 113, 50
Margaret Moore, 114, 20
William Sanchez, 115, 40
Dorothy Taylor, 116, 30
Jane Anderson, 117, 35
David Bell, 118, 20
Lisa Hall, 119, 150
Charles Morgan, 120, 20
Mateo Green, 121, 40
Vernon Scott, 122, 30
Mary Smith, 123, 45
100, 7, 1/1/2020
100, 8, 1/2/2020
100, 5, 1/3/2020
100, 6, 1/4/2020
101, 7, 1/3/2020
101, 6, 1/4/2020
101, 6, 1/5/2020
101, 8, 1/6/2020
```

Here is a screenshot of the Microsoft Excel file:



A1 | *fx* Department Name

	A	B	C	D	E	F	G	H
1	**Department Name**	Technical Support						
2								
3	**Workers**					**Shifts**		
4								
5	**Name**	**Id**	**Pay Rate**			**Worker Id**	**Hours Worked**	**Date**
6	Rose Diaz	100	50			100	7	1/1/2020
7	James Allen	101	25			100	8	1/2/2020
8	Richard Murphy	102	10			100	5	1/3/2020
9	Patricia Jones	103	20			100	6	1/4/2020
10	John Young	104	25			101	7	1/3/2020
11	Barbara Williams	105	30			101	6	1/4/2020
12	Elizabeth Davis	106	40			101	6	1/5/2020
13	Jennifer Brown	107	25			101	8	1/6/2020
14	Robert Hernandez	108	15			102	4	1/5/2020
15	Michael King	109	20			102	8	1/6/2020
16	Maria Miller	110	60			102	8	1/7/2020
17	Robert Hernandez	111	50			102	8	1/8/2020
18	Michael King	112	20			103	7	1/6/2020
19	Susan Wilson	113	50			103	7	1/7/2020
20	Margaret Moore	114	20			103	8	1/8/2020
21	William Sanchez	115	40			103	8	1/8/2020
22	Dorothy Taylor	116	30			104	6	1/7/2020
23	Jane Anderson	117	35			104	6	1/8/2020
24	David Bell	118	20			104	5	1/9/2020
25	Lisa Hall	119	150			104	5	1/10/2020
26	Charles Morgan	120	20			105	4	1/8/2020
27	Mateo Green	121	40			105	4	1/9/2020
28	Vernon Scott	122	30			105	3	1/10/2020
29	Mary Smith	123	45			105	3	1/11/2020
30						106	9	1/9/2020
31						106	8	1/10/2020
32						106	8	1/11/2020
33						106	9	1/12/2020
34						107	6	1/10/2020
35						107	7	1/11/2020
36						107	8	1/12/2020
37						107	5	1/13/2020