CSC 110: Fundamentals of Programming I
Assignment #7: Objects, searching, sorting

Due date

Sunday, December 6th, 2015 at 11:55 pm via submission to connex.

Specification document

How to hand in your work

Submit the file **`UvicOrganizer.java`** in the Assignment #7 link on connex.

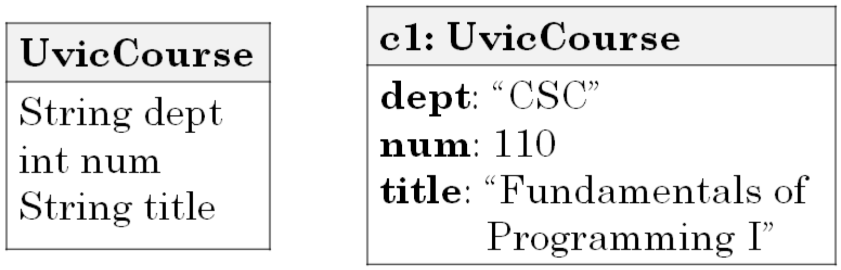
Learning outcomes

When you have completed this assignment, you will understand:

- How to create an *instance* of a *class* (object instantiation).
- How to invoke an *object's instance methods*.
- How to create and use an *array* of *objects* .
- How to *search* and *sort* through data in an array.

In this assignment you will be writing methods in a Java program UvicOrganizer.java. Your program will use UvicCourse objects. Download the **UvicCourse.java** file and save it in the same directory as the UvicOrganizer.java file.

UvicCourse objects have three instance variables, String dept, int num, and String title, as shown below. Suppose our course, CSC 110, was represented as an instance of a UvicCourse called c1, it would look like the following:



In this assignment, you will use a Scanner to read through a text file containing course data, and fill an array of UvicCourses with this data. You will then create methods to print, search and sort through the array.

Your program must:

1. Follow the specification document.
2. Ask the user to enter the name of the input file. Note that the marker can use a different data file than the examples provided in the Appendex. The data file will have exactly the same format (i.e., same columns, with each row having the same number of items) but it may contain more or fewer rows. ***Continue to ask for another input filename if the one entered is not valid.***

   Input file format (example: smallList.txt)
   - The first line contains the number of UvicCourse data elements are contained in the file (the total number of courses to put into the array)
   - Each remaining line contains the department name, followed by the course number. The remaining words on each line make up the course's title.

3. After linking a Scanner to an input file, write the method **makeArray**, that accepts the Scanner as a parameter, and then creates and fills a UvicCourse array with all of the UvicCourse data contained in the file the Scanner is linked to. The method then returns the array.

4. Using the array returned from the **makeArray** method, create and test all of the other methods according to the specification document. Examples of how to test these methods are shown in the Appendix.

Marking

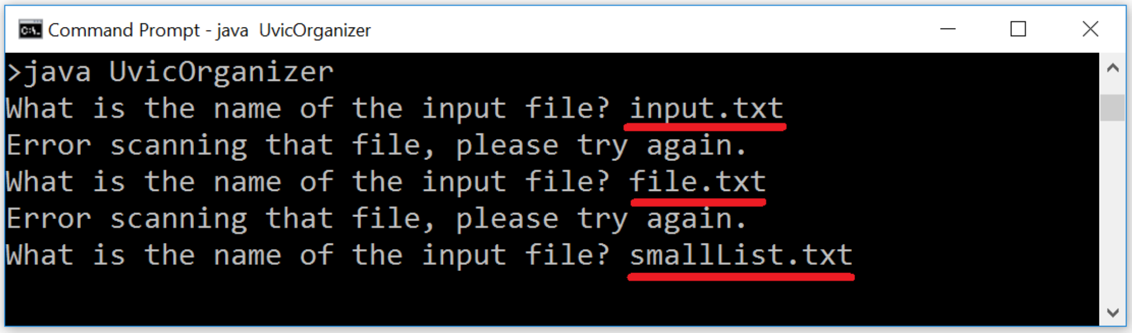
Your mark will be based on the following criteria:

- Your code *must compile and run*. Some examples of how to test your methods, along with expected output, are outlined in **Appendix A.**
- Your code must conform to all the requirements mentioned in the specification document.
- Test each of the required methods to ensure each one functions correctly.
- Your code must follow the guidelines outlined in Style_Guidelines.pdf, found through the Lectures & Stuff link in the Lab Resources folder on connex. You may notice that the specification document provides some very nice comments you are welcome to borrow.

Appendix A – Testing your code

Getting started setting up the `main` method:

Your main method will have two Scanners. The first Scanner is used to read in user input from the console (user entries underlined in red):



```
>java UvicOrganizer
What is the name of the input file? input.txt
Error scanning that file, please try again.
What is the name of the input file? file.txt
Error scanning that file, please try again.
What is the name of the input file? smallList.txt
```

The second Scanner scans the contents of a file. If the file cannot be found, the program prompts the user to enter another file name, until the second Scanner is successfully linked to a file (as shown above).

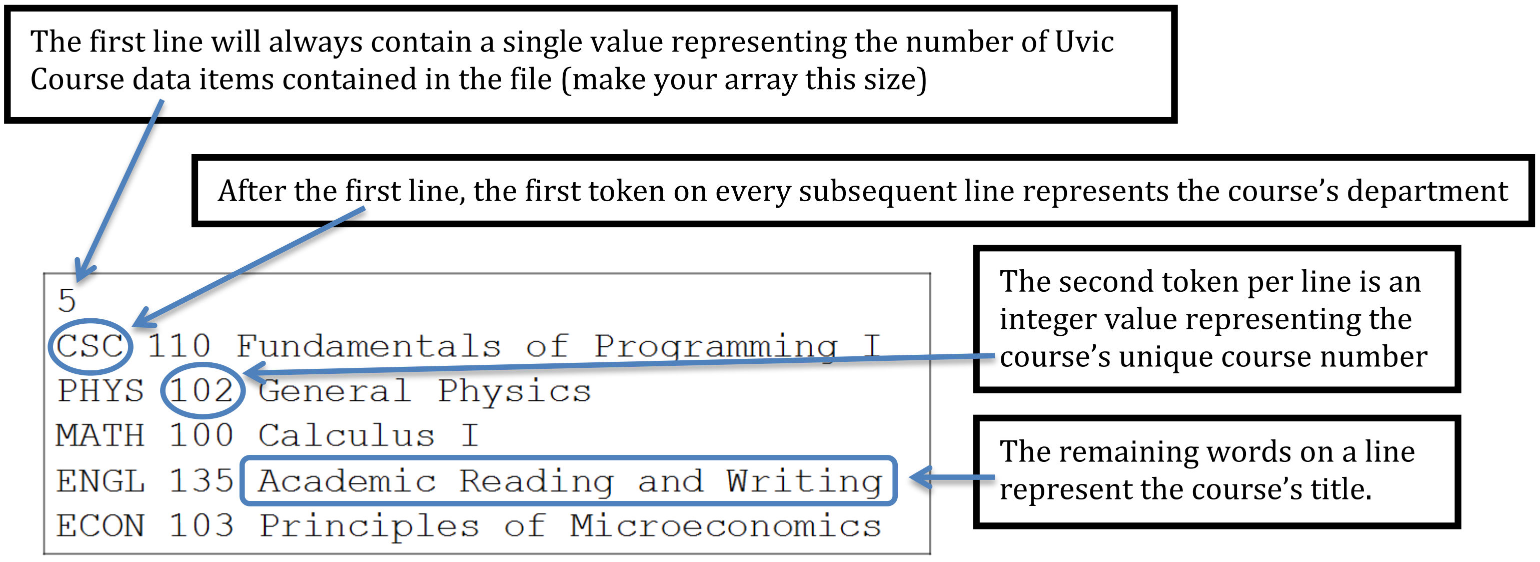
Testing the *`makeArray`* method:

Now that you have a Scanner linked to a text file, the next step is to copy all of the contents of the file into an array of UvicCourses.

Input files will all have the following format:

- The first line contains the number of UvicCourse data elements that are contained in the file (the total number of courses to put into the array)
- Each remaining line contains the department name, followed by the course number. The remaining words on each line make up the course's title.

Assuming we have linked a Scanner to the file **smallList.txt**:



The first line will always contain a single value representing the number of Uvic Course data items contained in the file (make your array this size)

After the first line, the first token on every subsequent line represents the course's department

The second token per line is an integer value representing the course's unique course number

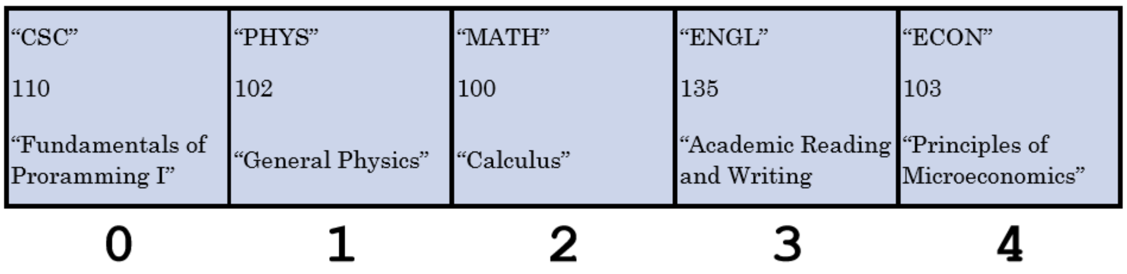
The remaining words on a line represent the course's title.

```
5
CSC 110 Fundamentals of Programming I
PHYS 102 General Physics
MATH 100 Calculus I
ENGL 135 Academic Reading and Writing
ECON 103 Principles of Microeconomics
```

Each line of data can be used to create a single instance of a UvicCourse. For example, the first line would create the following (in this case named c1):

c1: UvicCourse
dept: "CSC"
num: 110
title: "Fundamentals of Programming I"

Then each UvicCourse is inserted into an array, creating the following array of UvicCourse objects:

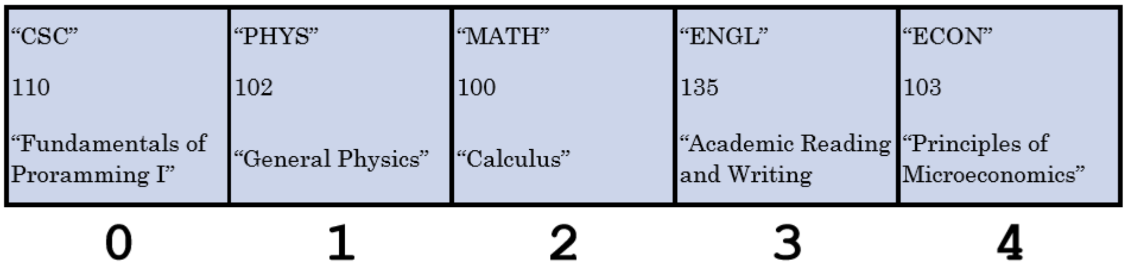


0	1	2	3	4
"CSC"	"PHYS"	"MATH"	"ENGL"	"ECON"
110	102	100	135	103
"Fundamentals of Proramming I"	"General Physics"	"Calculus"	"Academic Reading and Writing	"Principles of Microeconomics"

Testing the `printArray` **method:**

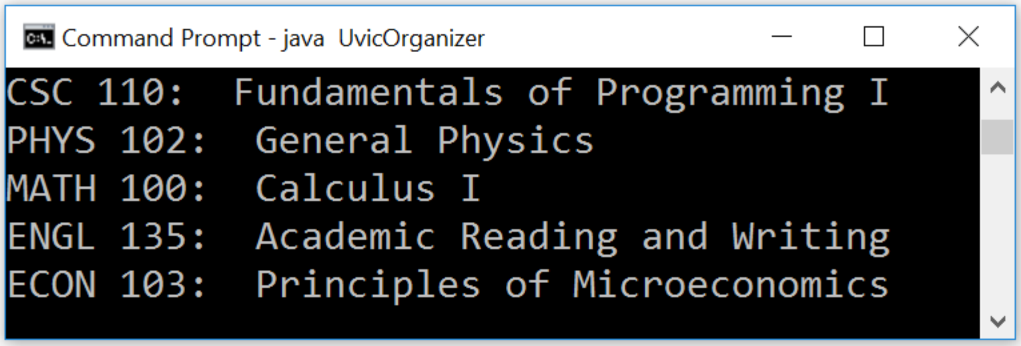
Given an array of UvicCourse objects, the method prints out information about each element in the array.

Given the following array (read in from **smallList.txt**):



0	1	2	3	4
“CSC”	“PHYS”	“MATH”	“ENGL”	“ECON”
110	102	100	135	103
“Fundamentals of Proramming I”	“General Physics”	“Calculus”	“Academic Reading and Writing	“Principles of Microeconomics”

Calling this method and passing in the array above produces the following output:



```
CSC 110:  Fundamentals of Programming I
PHYS 102:  General Physics
MATH 100:  Calculus I
ENGL 135:  Academic Reading and Writing
ECON 103:  Principles of Microeconomics
```

Testing the `listCoursesInDept` method:

Given an array of UvicCourses and a department name, this method prints out all UvicCourses in the array that match the department.

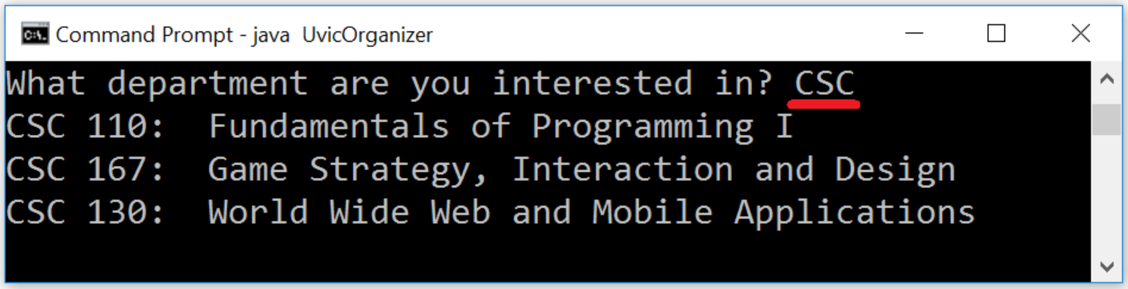
Example:

courseArray (read in from `firstYearList.txt`**):**

“CSC”	“MATH”	“PHYS”	“MATH”	"CSC"	"ENGL"	“ECON”	"CSC"	“MATH”	"ENGL"
110	151	102	100	167	135	103	130	122	147
“Fund..”	"Fin.."	“Gen..”	“Calc..”	Game.."	"Acad.."	“Prin..”	"Worl.."	"Logic.."	"Great.."
0	1	2	3	4	5	6	7	8	9

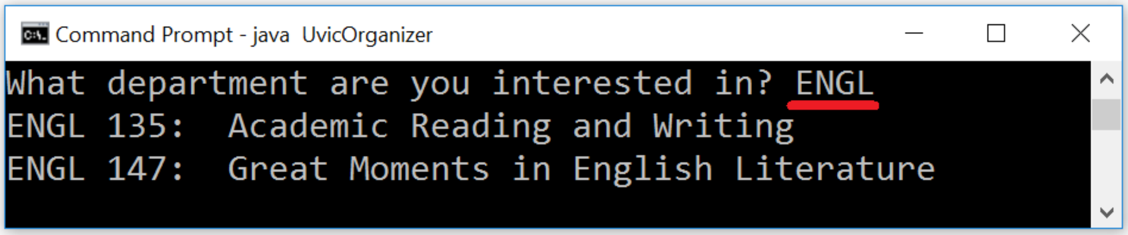
```
System.out.print("\nWhat department are you interested in? ");
String dept = console.next();
listCoursesInDept(dept, courseArray);
```

Given the above courseArray, and 3 lines of code, the following output is produced (input entered by the user into the console is underlined in red):



Command Prompt - java UvicOrganizer

```
What department are you interested in? CSC
CSC 110:   Fundamentals of Programming I
CSC 167:   Game Strategy, Interaction and Design
CSC 130:   World Wide Web and Mobile Applications
```



Command Prompt - java UvicOrganizer

```
What department are you interested in? ENGL
ENGL 135:   Academic Reading and Writing
ENGL 147:   Great Moments in English Literature
```

Testing the `listCoursesByDeptAndYear` **method:**

Given an array of UvicCourses and a department name, this method prints out all UvicCourses in the array that match the department.

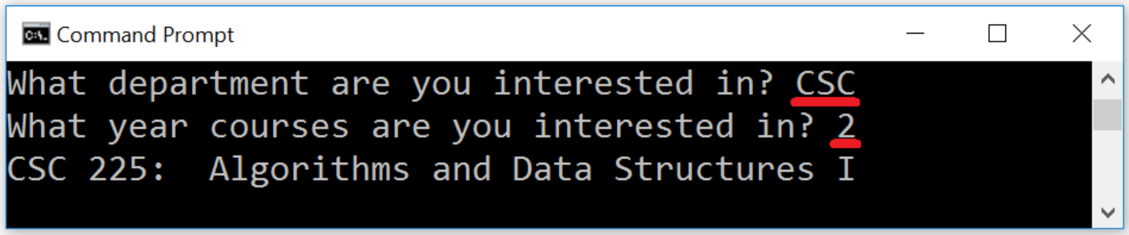
Example:

`courseArray (read in from file medList.txt):`

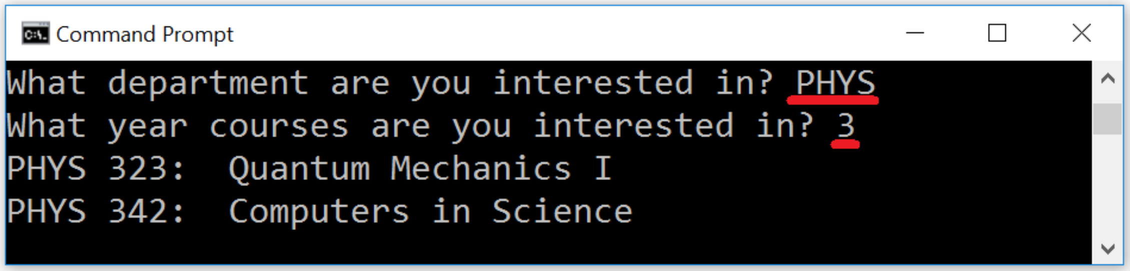
"PHYS"	"CSC"	"ENGL"	"PHYS"	"CSC"	"PHYS"	"MATH"	"ENGL"	"CSC"	"MATH"
323	498	341	342	110	102	100	135	225	211
"Quan.."	"Bio.."	"Old E.."	"Comp.."	"Fund.."	"Gen.."	"Calc.."	"Acad.."	"Algo.."	"Matr.."
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

```
System.out.print("\nWhat department are you interested in? ");
String dept = console.next();
System.out.print("What year courses are you interested in? ");
int year = console.nextInt();
listCoursesByDeptAndYear(dept, year, courseArray);
```

Given the above courseArray, and 5 lines of code, the following output is produced (input entered by the user into the console is underlined in red):



```
What department are you interested in? CSC
What year courses are you interested in? 2
CSC 225:  Algorithms and Data Structures I
```



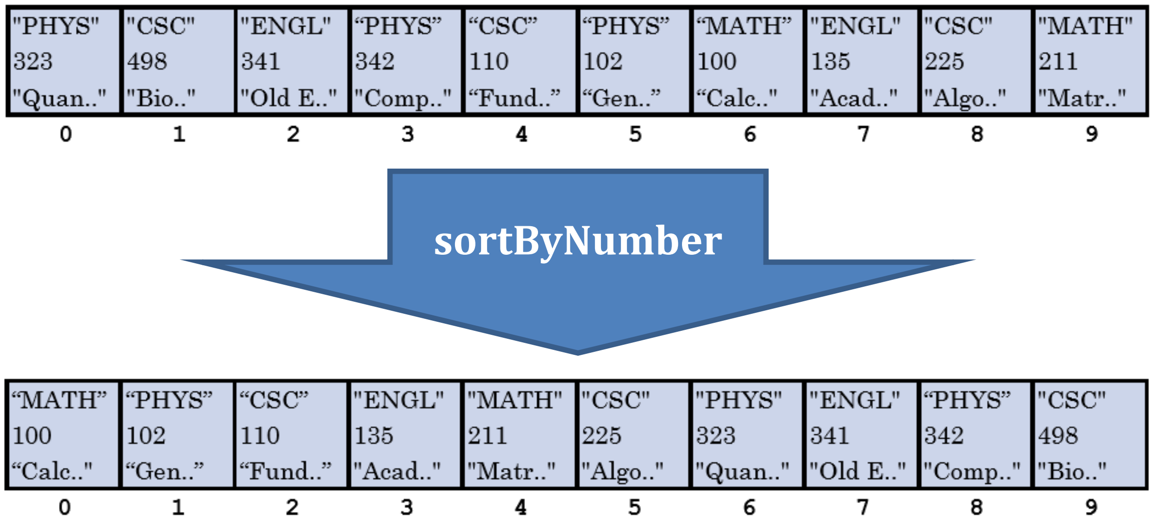
```
What department are you interested in? PHYS
What year courses are you interested in? 3
PHYS 323:  Quantum Mechanics I
PHYS 342:  Computers in Science
```

Testing the *`sortByNumber`* **method:**

Given an array of UvicCourses, this method sorts the array by the **`num`** instance variable.

Example:

`courseArray (read in from file medList.txt):`



"PHYS"	"CSC"	"ENGL"	“PHYS”	“CSC”	“PHYS”	“MATH”	"ENGL"	"CSC"	"MATH"
323	498	341	342	110	102	100	135	225	211
"Quan.."	"Bio.."	"Old E.."	"Comp.."	“Fund..”	“Gen..”	“Calc.."	"Acad.."	"Algo.."	"Matr.."
0	1	2	3	4	5	6	7	8	9

“MATH”	“PHYS”	“CSC”	"ENGL"	"MATH"	"CSC"	"PHYS"	"ENGL"	“PHYS”	"CSC"
100	102	110	135	211	225	323	341	342	498
“Calc.."	“Gen..”	“Fund..”	"Acad.."	"Matr.."	"Algo.."	"Quan.."	"Old E.."	"Comp.."	"Bio.."
0	1	2	3	4	5	6	7	8	9

Call your completed **`printArray`** method before and after sorting to ensure the array has been properly sorted.