DUE DATE: OCTOBER 2, 2019 AT 11:59PM

Instructions:

- You must complete the "**Blanket Honesty Declaration**" checklist on the course website before you can submit any assignment.
- Only submit the **java files**. Do ***not*** submit any other files, unless otherwise instructed.
- To submit the assignment, upload the specified files to the **Assignment 1** folder on the course website.
- Assignments must follow the **programming standards** document published on UMLearn.
- After the due date and time, assignments may be submitted but will be subject to a late penalty. Please see the ROASS document published on UMLearn for the course policy for late submissions.
- If you make multiple submissions, only the **most recent version** will be marked.
- These assignments are your chance to learn the material for the exams. Code your assignments independently. We use software to compare all submitted assignments to each other, and **pursue academic dishonesty vigorously**.
- Your Java programs must compile and run upon download, without requiring any modifications.
- ArrayLists are not allowed in this assignment. You must use (partially filled) arrays.

Assignment overview

In this assignment, you will implement a set of related classes of objects:

- **`Student`**: A university student
- **`StudentList`**: A list of students
- **`Course`**: A course with a title, short title, class list, etc.
- **`CourseList`**: A class list

Keep all of your methods short and simple. In a properly-written object-oriented program, the work is distributed among many small methods, which call each other. There are no methods in this entire assignment that should require more than 12 lines of code, not counting comments, blank lines, or {} lines (and many of them need no more than 3-4).

Unless specified otherwise, all instance variables must be **`private`**, and all methods should be **`public`**. Also, unless specified otherwise, there should be ***no*** **println** statements in your classes. Objects usually do not print anything, they only return **String** values from **toString** methods. The only exceptions in this assignment are **`register`** in **`Course`** (Phase 2) and **`withdraw`** in **`Course`** (Phase 3)**.**

Testing Your Coding

We have provided sample test files for each phase to help you see if your code is working according to the assignment specifications. These files are *starting points* for testing your code. Part of developing your skills as a programmer is to think through additional important test cases and to write your own code to test these cases. *For marking, we will use longer and more comprehensive tests*.

Phase 1: Student and StudentList

First, implement two simple classes: a **`Student`** class and a **`StudentList`** class

The **`Student`** class should have:

- Three instance variables: the student's first name (a **`String`**), last name (a **`String`**) and student number (an int)
- A constructor that has three parameters, in the above order, which are used to initialize the three instance variables.
- A standard **`toString`** method, which returns a **`String`** containing the student's last and first names separated by a comma, and student number in between parentheses (as shown in the output below).

- An **equals** method that checks if two Students are equal. Two students are equal if they have the same first name, last name and student number.

The **`StudentList`** class should have:

- Three instance variables: an array of Students, the maximum size of the array (an **`int`**) and the current number of students in the array (an **`int`**)
- A constructor that takes one parameter: the maximum size of the array. The other instance variables must be initialized in the constructor (assume that the StudentList will be empty when constructed).
- A method **contains(Student)** that checks if the student is already in the list and returns a boolean (true if in the list or false otherwise).
- A method **addStudent** that has a parameter of type Student, adds the Student to the list only if there is enough space. This method returns a boolean (true if the student was added and false otherwise).
- A **`toString`** method which returns a **`String`** containing all the Students in the list, one Student per line (as shown below).

You can test your classes using the supplied test program **`TestPhase1.java`**. You should get the output shown below.

```
Frost, Miranda (123001)

The list already contains: Frost, Miranda (123001)

Frost, Miranda (123001)
Bullion, Mister (123002)

Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)

Cannot add student: Grishenko, Boris (123004)
Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)
```

Phase 2: Course and CourseList

Next, implement the **`Course`** class, and a **`CourseList`**.

The **`Course`** class should have:

- A class constant: the maximum size of the wait list (you can set it to 100).
- Five instance variables: the title of the course (a **`String`**), the short title of the course (a **`String`** - e.g. COMP1020), the maximum class size (an **`int`**), a class list (a **`StudentList`**) and a wait list (a **`StudentList`**).
- A constructor that accepts three parameters, the title, short title and maximum class size. The two StudentLists should also be instantiated by the constructors.
- A method **`register`**(**Student**) that registers the Student to the appropriate list, if possible, using the rules described here:
    - if the student is already in the class list, nothing is to be done, and the method should print "The student [student description] is already registered to [course short title]!"
    - if the student is not in the class list, and space is available, the method adds the student to the class list and prints: "The student [student description] has been registered successfully to [course short title]." This method also adds the course to the course list of the student (see **Changes to Student class** below).
    - if the class list is full, then the method tries to add the student to the wait list
    - if the student is already on the wait list, nothing is to be done and the method should print "The student [student description] is already on the wait list for [course short title]!"
    - if the student is not on the wait list and space is available, the method adds the student to the wait list and prints: "The student [student description] has been placed on the wait list for [course short title]."

  - if the wait list is full, nothing is to be done and the method prints “The wait list is full for [course short title]!”

- A **getTitles** method, that returns a **String** containing the course short title followed by a dash and the course title.
- A **toString** method that returns a **String** containing the course short title followed by a dash and the course title on the first line. On the following lines, the class list and wait list are printed (see example below).

Then, implement a **CourseList** class. This class should have:

- A class constant: the maximum number of courses a student can take (set it to 100).
- Two instance variables: an array of Courses, and the current number of courses in the list (an **int**)
- A constructor that has no parameters, which initializes the instance variables (the CourseList should be empty when created).
- A method **contains**(**Course**) that returns true if the course is already in the list, false otherwise.
- An **addCourse(Course)** method (void) that adds the Course to the list.
- a **getAllTitles** method that calls the **getTitles** method for each Course of the list, and builds a **String** representation of all the courses on the list, one Course per line.
- A **toString** method that returns a **String** containing the full String representation (titles, class list and wait list) of all Courses in the list (as shown in the example below).

Changes to Student class:

- Add an additional instance variable: a CourseList, containing the list of all courses the student is registered to (fully registered, not on the wait list).
- The constructor should instantiate the above instance variable (empty CourseList).
- A method **addCourse(Course)**, that adds the Course to the CourseList. You may assume that it will never get full.
- A method **getCourseListString**, that returns a **String** containing “Student [student description] is registered to:” followed on the next lines by the list of course titles, using the **getAllTitles** method of the CourseList class.

You can test your class with **TestPhase2.java**. You should get the output shown below.

```
The student Frost, Miranda (123001) has been registered successfully to COMP1010.
The student Bullion, Mister (123002) has been registered successfully to COMP1010.
The student Simonova, Natalya (123003) has been registered successfully to COMP1010.
The student Grishenko, Boris (123004) has been registered successfully to COMP1010.
The student Davidov, Sacha (123005) has been registered successfully to COMP1010.
The student Carver, Paris (123006) has been registered successfully to COMP1010.
The student Gupta, Henry (123007) has been placed on the wait list for COMP1010.

COMP1010 - Introductory Computer Science 1
Class list:
Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Wait list:
Gupta, Henry (123007)

The student Simonova, Natalya (123003) has been registered successfully to COMP1020.
The student Onatopp, Xenia (123008) has been registered successfully to COMP1020.
The student Drax, Hugo (123009) has been registered successfully to COMP1020.
The student Zukovsky, Valentin (123010) has been registered successfully to COMP1020.
The student Ourumov, Arkady Grigorovich (123011) has been registered successfully to
COMP1020.
The student Mishkin, Dmitri (123012) has been placed on the wait list for COMP1020.
The student Lynd, Vesper (123013) has been placed on the wait list for COMP1020.
The student Trevelyan, Alec (123014) has been placed on the wait list for COMP1020.
The student Scaramanga, Francisco (123015) has been placed on the wait list for
COMP1020.
The student Jones, Christmas (123016) has been placed on the wait list for COMP1020.
The student Lin, Wai (123017) has been placed on the wait list for COMP1020.
The student King, Elektra (123018) has been placed on the wait list for COMP1020.

COMP1020 - Introductory Computer Science 2
Class list:
Simonova, Natalya (123003)
Onatopp, Xenia (123008)
Drax, Hugo (123009)
Zukovsky, Valentin (123010)
Ourumov, Arkady Grigorovich (123011)
Wait list:
Mishkin, Dmitri (123012)
Lynd, Vesper (123013)
Trevelyan, Alec (123014)
Scaramanga, Francisco (123015)
Jones, Christmas (123016)
Lin, Wai (123017)
King, Elektra (123018)

The student Gupta, Henry (123007) is already on the wait list for COMP1010!

COMP1010 - Introductory Computer Science 1
Class list:
Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Wait list:
Gupta, Henry (123007)

The student Drax, Hugo (123009) is already registered to COMP1020!

COMP1020 - Introductory Computer Science 2
Class list:
Simonova, Natalya (123003)
Onatopp, Xenia (123008)
Drax, Hugo (123009)
Zukovsky, Valentin (123010)
Ourumov, Arkady Grigorovich (123011)
Wait list:
Mishkin, Dmitri (123012)
Lynd, Vesper (123013)
Trevelyan, Alec (123014)
Scaramanga, Francisco (123015)
Jones, Christmas (123016)
Lin, Wai (123017)
King, Elektra (123018)
```

```
The student Simonova, Natalya (123003) has been registered successfully to COMP2140.
The student Simonova, Natalya (123003) has been registered successfully to COMP2150.
The student Simonova, Natalya (123003) has been registered successfully to COMP3350.

Student Simonova, Natalya (123003) is registered to:
COMP1010 - Introductory Computer Science 1
COMP1020 - Introductory Computer Science 2
COMP2140 - Data Structures and Algorithms
COMP2150 - Object Orientation
COMP3350 - Software Engineering 1
```

Phase 3: Withdrawing from a Course

Next implement a **withdraw(Student)** method in the **`Course`** class, which will withdraw the Student from the class list, if the Student is on the class list, or withdraw the Student from the wait list, if the Student is on the wait list, or do nothing if the Student is not in any of the lists.

- The method must print:
    - "The student [student description] has been withdrawn from [course short title]." if the student was on the class list.
    - "The student [student description] has been withdrawn from the wait list of [course short title]." if the student was on the wait list.
    - "The student [student description] is not on any list of [course short title]." otherwise.
- If the Student was on the class list, the method must also remove the Course from the Student's CourseList.
- If the Student was on the class list, and there is at least one Student on the wait list, the first Student on the wait list must be registered to the class list (and removed from the wait list).
- When removing a Student from a class/wait list, or removing a course from a CourseList, you must fill in the gap by shifting up all the following Students/Courses. Also, don't forget to update the counter of the current number of students or current number of courses.
- **Important: to fully complete Phase 3, following the instructions above, you will need to add other methods to other classes (while maintaining encapsulation – every object is responsible for modifying its own instance variables).**

You can test your class with TestPhase3.java. You should get the output shown below:

```
The student Frost, Miranda (123001) has been registered successfully to COMP1010.
The student Frost, Miranda (123001) has been registered successfully to COMP2150.
The student Bullion, Mister (123002) has been registered successfully to COMP1010.
The student Simonova, Natalya (123003) has been registered successfully to COMP1010.
The student Grishenko, Boris (123004) has been registered successfully to COMP1010.
The student Davidov, Sacha (123005) has been registered successfully to COMP1010.
The student Carver, Paris (123006) has been registered successfully to COMP1010.
The student Gupta, Henry (123007) has been placed on the wait list for COMP1010.
The student Onatopp, Xenia (123008) has been placed on the wait list for COMP1010.

COMP1010 - Introductory Computer Science 1
Class list:
Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Wait list:
Gupta, Henry (123007)
Onatopp, Xenia (123008)
```

```
Student Frost, Miranda (123001) is registered to:
COMP1010 - Introductory Computer Science 1
COMP2150 - Object Orientation

The student Zukovsky, Valentin (123010) is not on any list of COMP1010.

COMP1010 - Introductory Computer Science 1
Class list:
Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Wait list:
Gupta, Henry (123007)
Onatopp, Xenia (123008)

The student Gupta, Henry (123007) has been withdrawn from the wait list of COMP1010.

COMP1010 - Introductory Computer Science 1
Class list:
Frost, Miranda (123001)
Bullion, Mister (123002)
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Wait list:
Onatopp, Xenia (123008)

The student Frost, Miranda (123001) has been withdrawn from COMP1010.
The student Onatopp, Xenia (123008) has been registered successfully to COMP1010.

COMP1010 - Introductory Computer Science 1
Class list:
Bullion, Mister (123002)
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Onatopp, Xenia (123008)
Wait list:

The student Bullion, Mister (123002) has been withdrawn from COMP1010.

COMP1010 - Introductory Computer Science 1
Class list:
Simonova, Natalya (123003)
Grishenko, Boris (123004)
Davidov, Sacha (123005)
Carver, Paris (123006)
Onatopp, Xenia (123008)
Wait list:

Student Frost, Miranda (123001) is registered to:
COMP2150 - Object Orientation
```

Hand in

Submit your four Java files (`Student.java`, `StudentList.java`, `Course.java, CourseList.java`). ***Do not submit .class or .java~ files!*** You do ***not*** need to submit the `TestPhaseN.java` files that were given to you. If you did not complete all phases of the assignment, **use the Comments field when you hand in the assignment to tell the marker which phases were completed**, so that only the appropriate tests can be run. For example, if you say that you completed Phases 1-2, then the marker will compile your files, and compile and run the tests for phases 1 and 2. If it fails to compile and run, you will lose ***all*** of the marks for the test runs. The marker will ***not*** try to run anything else, and will ***not*** edit your files in any way. ***Make sure none of your files specify a package at the top!***