LAB1 – Java programming warm up

1. Create an application called Registrar that has the following classes:

   A ***Student*** class that minimally stores the following data fields for a student:
   - Name
   - Student id number
   - Number of credits
   - Total grade points earned

   The following methods should also be provided:
   - A constructor that initializes the name and id fields
   - A method that returns the student name field
   - A method that returns the student ID field
   - A method that determines if two student objects are equal if their student id are the same (override equals from the class Object)
   - Methods to set and retrieve the total number of credits (integer only)
   - Method to set and retrieve the total number of grade points earned (integer only)
   - A method that returns the GPA (grade points divided by credits)

   An ***Instructor*** class that minimally stores the following data fields for an instructor:
   - Name
   - Faculty id number
   - Department

   The following methods should also be provided:
   - A constructor that initializes the name and id fields
   - Methods to set and retrieve the instructor's department

   A ***Course*** class that minimally stores the following data for a course:
   - Name of the course
   - Course registration code
   - Instructor
   - Maximum number of students
   - Number of students
   - Array stores students registered in the course (use an regular array instead of ArrayList)
   - Maximum size of waitlist
   - Number of wait student (optional)

- Array stores students in the waitList (use an regular array instead of ArrayList)

The following methods should also be provided:

- A constructor that initializes the name, registration code, and maximum number of students (initial the registered students array with this maximum capacity) and Maximum size of waitlist (initial the waitList array with this maximum capacity).
- Methods to set and retrieve the instructor
- A method to search for a student in the course; the search should be based on an ID number.
- A method to add a student to the course.
    - If the course is full, check the waitlist, add to waitlist if possible. Otherwise, report action by saying the class is full.

  Be sure that the student is not already in the studentList or waitlist before you do the insertion. The records in waitList should be in order to indicate priority.
- A method to remove a student from the studentList.
    - If the student is not found, report action failure.
    - If the student is found, remove the student from the studentList. Then check if the waitlist is empty. If the waitlist is not empty, move the student waited longest time to the studentList.

You may notice that the Student and Instructor classes described above have some commonality. Create a ***Person*** class that captures this commonality and uses it as a base class for Student and Instructor. This class should be responsible for the name and id fields and also provide a *toString* method that returns a string of the form name, id. This will be the inherited *toString* method for the Student and Instructor class.

Implement the previous classes in Java. Write a main Program that provides a menu to allow the users to

- **Create a course, prompting the user for all of the course information.**
- **Add students to the course**
- **Check to see if a student is registered in the course**
- **Remove a student from the course**
- **Check how many student is on the waitList**