Program #2 (25 points)

Important Dates

1. TeamMember class with **testbed main** due Wednesday, Feb. 17 @10pm, or **-2 points**.
2. Team class **JUnit test** due Friday, Feb. 19 @10pm, or **-2 points.**
3. All Java files due on the grader Monday, Feb. 22 @ 10pm.
4. Grace date: Wednesday, **Feb. 24 @10pm.**

** Item #1 and #2 due on the **K drive** in **1Dropbox** folder

Program Description

In this program, you will implement a growable container class Team, and use it to hold a group of TeamMembers. The program will allow the user to create a project team without the restriction of how many team members you can have in a team. You should provide the following operations: **add**, **remove**, and **print** team members. Your program should read and process commands from the standard input, and display the results to the standard output. When the program runs, display "`Let's start a new team!`", and your program MUST be able to process the following commands. All commands are case-sensitive, which means the commands with lowercase letters are invalid!

- **`A`** `command`—**add** a team member to the team. For example, `A Lily 2/3/2011,` which adds Lily as a team member who joined the company since 2/3/2011. You must check duplicate. If Lily is already in the team (given the same name and same date exist in the team), display "`Lily 2/3/2011 is already in the team.`". If Lily was successfully added, display "`Lily 2/3/2011 has joined the team.`". You MUST check if a date is valid.
- **`R`** command—**remove** a team member from the team. For example, `R Lily 2/3/2011`, which removes Lily from the team. If the remove was not successful, display `"Lily 2/3/2011 is not a team member."`, otherwise display `"Lily 2/3/2011 has left the team."`. You MSUT check if a date is valid.
- **`P`** command—**output** all members in the team. If the team is empty, display `"We have 0 team members!"`, otherwise display `"We have the following team members: "` followed by the list of team members, and `"-- end of the list --"`.
- **`Q`** `command`—output all members in the team, display "`The team is ready to go!`", and terminate the program.

Program Requirement

1. You MUST follow the software development ground rules.
2. You must submit the program with 0 differences to pass this course.
3. Program submitted with differences or after **Wednesday, Feb. 24 @10pm** is worth **0 points**!
4. This is an **individual assignment**. Sharing your solution could result in being accused of **plagiarism**. I will ALWAYS run the plagiarism check.
5. You MUST use **SE tools** to keep an up-to-date log of the time spent working on this program. You must use the automatic **punch in** and **punch out** when possible. Select the activities and provide specific comments about what you worked on during that period. Up to **-5 points** if this is not done or you do not log all your time or you do not provide specific comments as to what you were working on.
6. You MUST have the testbed main in the TeamMember class, and submit the Word document for the test cases. You must create and submit the JUnit test file TeamTest.java. You will lose up to **5 points** if you failed to submit these files on the K drive.
7. Each class must go in a separate file. Month.java defines all constants you need for this program. For example, use **`Month.JAN`** to access the value of 1. You must submit all .java files to the grader.
8. If you need help from me, place your project in **K:\Courses\CSSE\changl\cs2430\your_login_id**. So I can access it from my office when you stop by.

9. Your program MUST handle bad commands! The commands are case-sensitive. You must use **switch case** statement and the **`charAt(0)`** method of the String class to check the commands. **-2 points** if you didn't.
10. Download all Java files in the CSSE course file folder Prog2. You MUST finish the following classes, and you MUST finish all the methods in the Java files. **-2 points** for each method not finished or not used.

- **Date class.** This class defines the properties of a date. You CANNOT add other data members to this class, you CANNOT change the signature of the methods, and you CANNOT do I/O in this class. **-2 points** if you did. You MUST check if a date is valid. For example, for the month of January, March, May, July, August, October and December, there are 31 days; for April, June, September and November, there are 30 days; February has 28 days in a normal year, and 29 days in a leap year. To determine whether a year is a leap year, follow these steps:

  Step 1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
  Step 2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
  Step 3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
  Step 4. The year is a leap year.
  Step 5. The year is not a leap year.

- **TeamMember class**. This class defines the properties and methods of a team member. You CANNOT add other data members to this class, you CANNOT change the signature of the methods, and you CANNOT do I/O in this class. **-2 points** if you did. You must write a **test bed main** for this class. This is a small main that you put at the bottom of your class that is used to test your class. You MUST include simple tests for ALL constructors and methods. **-1 points** for each method not tested. You MUST print out the results to ensure the correctness of the results. All test cases must be documented with a Word document. For example:

Test Case	Description	Sample Input	Expected result / output
1	Test the constructor and use toString() method to display the content of the object.	"Lily", "11/27/2011"	Lily 11/27/2011
2	Test the .equals() method	Case 1: "Lily", "11/27/2011" "Lily", "11/28/2011" Case 2: "Lily", "11/27/2011" "Lily", "11/27/2011"	Case 1: not equal Case 2: equal
...	…	…	…

You must follow the instructions in the "Test Specification" section of the Software Development Ground Rules. Name your Word document with your login name followed by an underscore followed by **p2test.docx**. For example**, johnsona_p2test.docx.** When you are done, zip the **Word document, TeamMember.java** and **Date.java** into a folder called your_login_name_prog2.zip (for example, **johnsona_prog2_testbed.zip**,) and copy the zip file to the dropbox folder: **K:\Courses\CSSE\changl\cs2430\1Dropbox**. I WILL NOT grade your program if it is not properly named.

- **Team class**. This is the **container class** that defines a team and its operations. You CANNOT add other data members to this class, you CANNOT change the signature of the methods, and you CANNOT do I/O in this class, except the **print()** method. **-2 points** if you did. The add() method appends item to the end of the array, NO NEED to check duplicate here. The remove() method calls find() method and finds the index of the team member to be removed. Move the last item in the array to fill the slot and set the reference of the last item to null.

  You MUST create a **JUnit test** for this class. Follow the steps.

  a) Close Team class. (Right click on the class and select Close.)

b) Right click on Team class, and select Tool, and then select Create/Update Tests.
c) In the Create Tests Window, make sure the location is "Test Packages" and click OK. If prompted, select JUnit 4.x. This should create a new class in Test Packages called TeamTest.
d) Open TeamTest.java if it is not opened already. (It is in the default package of the Test Packages folder.)
e) Ignore TeamTest(), setUpClass(), tearDownClass(), setUp(), and tearDown(). There is a method in TeamTest to test each public method in Team class. JUnit has tried to set up one test for each method. In some cases, the automatically-generated test is OK. In other cases, it is NOT OK and need to be modified. In all cases, one test is not enough to thoroughly test a particular method. You are going to write more tests for each method.
f) Each test method in TeamTest declares a variable instance and creates an object of Team class and assigns it to instance. It is a standard name that JUnit uses. You can call any instance methods such as:

```
instance.add(new TeamMember("Lily", new Date("11/27/2011")));
or
if ( instance.remove(new TeamMember("Lily", new Date("11/27/2011"))) );
...
```

g) Each test method in TeamTest also uses two other variables expResult and result to assist you to write JUnit tests. You don't have to use them and could remove them if you do not use them.

JUnit also has three methods that allow you to check if the values returned are what they are supposed to be: `assertTrue(), assertFalse()` and `assertEquals().` Here are some examples:

```
// this test expects that the container is empty.
    assertTrue( instance.isEmpty() );

// this test expects that removing the new team member will fail,
// in other words, the new team member is not in the team.
    assertFalse( instance.remove(new TeamMember("Lily", new Date("11/30/2011"))));
```

If the tests for a method all pass, then when you run the test (right click the class file and select Run File), they show up as green, if a test fails, it shows up as red.

h) Run the tests: select Run/test Project (Prog2), in the output window, you will see the JUnit test results. In particular, you should see that all tests have FAILED and are shown in red. Under testRemove(), **remove the fail() line** and change the code as follows.

Change

```
TeamMember = null;
```

to

```
TeamMember m = new TeamMember("Lily", new Date("11/27/2011"));
```

Change

```
boolean expResult = false;
```

to

```
boolean expResult = true;
```

add

`instance.add(m);` above the statement: `boolean result = instance.remove(m);`

Run the JUnit tests again. Run / Test Project (Prog2) and this time testRemove() should be marked as PASSED. You can try to remove someone that is not in the team, and expect a false in this case.

i) For testAdd(), you must add enough instances of TeamMember to make the list "grow".
j) For every other method, remove the fail() line, modify the test created by JUnit, if necessary, and put in some additional good tests. The tests should attempt to THOROUGHLY test the method. The test should set up the list, then assert something. For testPrint(), just put elements in the list and call the print method WITHOUT asserting anything.

When you are done, zip **TeamTest.java** and **Team.java** into a zip file called your_login_name_prog2_JUnit.zip (for example, **johnsona_prog2_JUnit.zip** and copy the file to the dropbox folder: **K:\Courses\CSSE\changl\cs2430\1Dropbox**.

- **ProjectManager class.** This class is the user interface class that handles **input** commands and **output** messages. Finish the **run() method** and all private methods. The main method in Prog2.java will call the run() method here. You MUST check if the team "contains" the member you're adding by calling the .**contains()** method. When adding or removing a member, you must check if a given date is valid.

Sample Input

```
R Lily 11/27/2011
r Lily 11/27/2011
p
P
A Lily 2/29/2011
A Lily 4/31/2011
A Lily 6/31/2011
A Lily 9/31/2011
A Lily 11/31/2011
A Lily 2/28/2011
A Chang 4/30/2008
A Scanlan 2/29/2000
A Clifton 6/32/1995
A Clifton 6/0/1995
A Clifton 6/1/1995
P
A Langraf 1/31/1997
A Langraf 1/31/1997
A Shi 3/31/2011
R Shi 3/30/2011
P
R Shi 3/31/2000
R Shi 3/31/2011
P
A Yang 13/1/2016
A Yang 0/1/2016
A Yang 12/30/2012
A Gavin 9/30/1999
A Tian 10/31/2013
A Shi 2/29/2012
R Langraf 1/31/1997
P
A Chang 5/1/2012
A Chang 7/31/2012
a Chang 7/31/2012
R Tian 11/30/2012
P
R Lily 2/28/2011
R Chang 4/30/2008
R Shi 2/29/2012
R Scanlan 2/29/2000
q
Q
```

Sample Output

```
Let's start a new team!
Lily 11/27/2011 is not a team member.
Command 'r' not supported!
Command 'p' not supported!
We have 0 team members!
2/29/2011 is not a valid date!
4/31/2011 is not a valid date!
6/31/2011 is not a valid date!
9/31/2011 is not a valid date!
11/31/2011 is not a valid date!
Lily 2/28/2011 has joined the team.
Chang 4/30/2008 has joined the team.
Scanlan 2/29/2000 has joined the team.
6/32/1995 is not a valid date!
6/0/1995 is not a valid date!
Clifton 6/1/1995 has joined the team.
We have the following team members:
Lily 2/28/2011
Chang 4/30/2008
Scanlan 2/29/2000
Clifton 6/1/1995
-- end of the list --
Langraf 1/31/1997 has joined the team.
Langraf 1/31/1997 is already in the team.
Shi 3/31/2011 has joined the team.
Shi 3/30/2011 is not a team member.
We have the following team members:
Lily 2/28/2011
Chang 4/30/2008
Scanlan 2/29/2000
Clifton 6/1/1995
Langraf 1/31/1997
Shi 3/31/2011
-- end of the list --
Shi 3/31/2000 is not a team member.
Shi 3/31/2011 has left the team.
We have the following team members:
Lily 2/28/2011
Chang 4/30/2008
Scanlan 2/29/2000
Clifton 6/1/1995
Langraf 1/31/1997
-- end of the list --
13/1/2016 is not a valid date!
0/1/2016 is not a valid date!
Yang 12/30/2012 has joined the team.
Gavin 9/30/1999 has joined the team.
Tian 10/31/2013 has joined the team.
Shi 2/29/2012 has joined the team.
Langraf 1/31/1997 has left the team.
We have the following team members:
Lily 2/28/2011
Chang 4/30/2008
```

```
Scanlan 2/29/2000
Clifton 6/1/1995
Shi 2/29/2012
Yang 12/30/2012
Gavin 9/30/1999
Tian 10/31/2013
-- end of the list --
Chang 5/1/2012 has joined the team.
Chang 7/31/2012 has joined the team.
Command 'a' not supported!
Tian 11/30/2012 is not a team member.
We have the following team members:
Lily 2/28/2011
Chang 4/30/2008
Scanlan 2/29/2000
Clifton 6/1/1995
Shi 2/29/2012
Yang 12/30/2012
Gavin 9/30/1999
Tian 10/31/2013
Chang 5/1/2012
Chang 7/31/2012
-- end of the list --
Lily 2/28/2011 has left the team.
Chang 4/30/2008 has left the team.
Shi 2/29/2012 has left the team.
Scanlan 2/29/2000 has left the team.
Command 'q' not supported!
We have the following team members:
Chang 7/31/2012
Chang 5/1/2012
Gavin 9/30/1999
Clifton 6/1/1995
Tian 10/31/2013
Yang 12/30/2012
-- end of the list --
The team is ready to go!
```