Programming Fundamentals 2

Repeat Assignment 2018 (100% of Module)

Please submit your repeat assignment on or before noon, August 31st, 2018. You need to attend an interview to assess your understanding of the submitted code. This will take place ton Monday afternoon, 2-5 pm September 3rd, 2018.. The interview marking will be used as a multiplier to determine your final mark.

The aim of this assignment is to develop a **Boat Club App** that holds details on Members and Crews. A member can be either a Crew Member or a Trainer Member (i.e. cox, coach).

Please read this section before starting the assignment:

UX approach:

- Your menu driven app should be user friendly, report progress to the user, robust, handle exceptions, intuitive, etc.
- The menu system should be well tested for many different user-input scenarios e.g. case sensitivity, input mismatches, invalid indexes, etc.

DX approach:

- Javadoc comments should be written for each method and class. Internal "programmer" comments should be used where required also.
- Your name should appear as the @author at the top of each java file submitted.
- Java naming standards must be followed e.g.:
    - Class names begin with capital letter – lowercase after that., e.g. **Member**
    - Field names / local variables begin with lowercase, use uppercase to separate words in name, e.g **crewName**
    - For instances of collections they should follow the conventions for field names and should be plural, e.g. **crewMembers.**
- Java coding principles / standards should be applied throughout the project.

Note on validations:

- When you wish to update fields though methods (though constructors or setters), and the input data/new value is not valid according to the validation rules, you deal with this differently in the two cases. In constructors, you should give the field a default value, (0 if this is allowed by the rules of validation, some other sensible valid value otherwise). In the setters, you should leave the field unchanged if the new value is invalid according to the validation rules.

Submission:

- Before you submit, save your project name as your name (i.e. first character of firstname followed by second name, e.g mmeagher for Mairead Meagher's assignment). Then zip it and submit it.

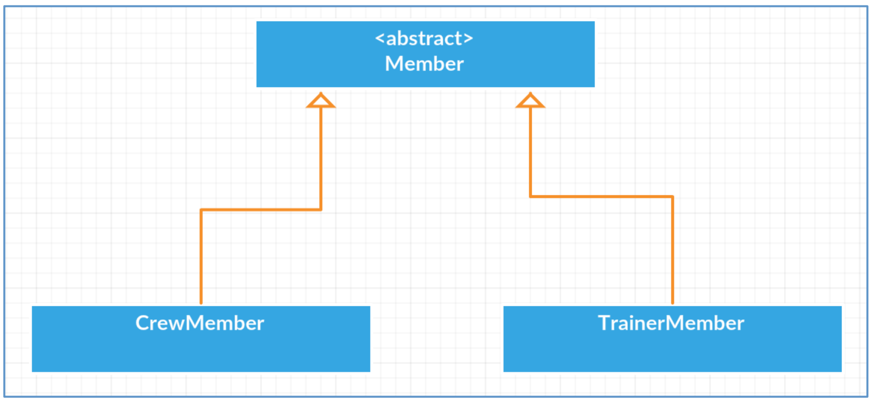
A note on the marking of the assignment:

- All methods should override and extend the overridden method, where possible.
- Serialization should use the XStream component we discussed in class.
- You are permitted to use any applicable code from your labs / previous assignments. However if you use anything from other sources, you must reference it. Failure to reference code taken from other sources is plagiarism and is a serious academic offence.
- An indicative marking scheme is as follows:

Structure and Principles adopted (i.e. DX approach, UX approach)	5%
Phase 1 (inheritance, abstraction and polymorphism)	25%
Phase 2 (Crew class)	10%
Phase 3 (BoatClub class)	40%
Phase 4 (MenuController class)	20%

Phase 1 – Inheritance and Polymorphism:

Create the following inheritance hierarchy in Java:



- Member (abstract). Stores id, name, phone number and email. The validation on these fields are as follows:

  ```
  //member id valid values 1000 to 9999 inclusive (default 1000)
  //member name, max 25 characters (default value if no name supplied
  //    is the empty string, "")
  //phone number, max 15 characters (default value if none supplied
  //    is "UNKNOWN")
  //email must contain a @ and a . in it.  (default value if none
  //    supplied is "NO VALID EMAIL")
  ```
  Member has an abstract method called isFullyTrained() with a boolean return type.

- CrewMember (concrete). Subclass of Member. Stores whether they are skullTrained or not. The validation is as follows:

  ```
  //skullTrained – true if the crew member is trained to be a crew
  //member of a double bladed (oar) boat.  The default value is false.
  ```

  The abstract method, isFullyTrained(), from Member is implemented here. If the skullTrained value is true, return true, otherwise return false.

- TrainerMember (concrete). Subclass of Member. Stores whether the member is garda vetted (boolean), their coachingLevel (int) and if they are cox trained (boolean). The validation is as follows:

  ```
          //gardaVetted:  true if they are garda vetted, false otherwise
  (default value false).
          //coachingLevel: valid levels are 1, 2, 3 (default value 0).  The
          //     coaching level can only go up i.e. if they are coaching level
          1, //     they can only go up to 2 or 3 and cannot be set back to
          zero.
          //cox: true if they are trained to be a cox, false otherwise
  (default value false)
  ```

The abstract method, isFullyTrained(), from Member is implemented here. If the coaching level is 3 and the trainer is both garda vetted and trained to be a cox, return true, otherwise return false.

The following rules should be applied to this hierarchy:

- The Member class should have a String toString() method that formats the printing of the object state and returns it. It's subclasses should override the superclass method String toString() so that they can report on the new fields defined in these subclasses.
- Each class in the hierarchy should define:
    - a constructor that initialises each instance variable based on user input data
    - getters and setters for each instance field.

Phase 2 – Crew Class:

The Crew class is a concrete class and has the following instance fields (and their associated validations):

`private TrainerMember coach;`	All crews have exactly one coach. If a crew does not have a coach, the value here will be null (note: default is null). Coach must be garda vetted and min level 1 training to be assigned a crew.
`private TrainerMember cox;`	Some boats are coxless. In this case, a value of null indicates that the crew doesn't have a cox. If the boat has a cox, the TrainerMember is stored in this field.
`private String boatType;`	The type of boat indicates how many crew members row in it, how many oars each crew member has and whether the boat can hold a cox. Assume for this assignment that the boats can hold one, two or four crew members. Therefore, valid values for a boat type are: SKULL (1 crew member, 2 oars, no cox). PAIR (2 crew members, 1 oar each, no cox). DOUBLE SKULL (2 crew members, 2 oars each, no cox). FOUR (4 crew members, 1 oar each, one cox). COXLESS FOUR (4 crew members, 1 oar each, no cox). Note: the number of oars per crew member will only come into use if you are attempting the extra credit section below.
`private String crewName`	Max 20 characters (default value is "UNKNOWN");
`private ArrayList <CrewMember> crewMembers`	The crew members rowing in the boat.

This class has at least:

- a standard constructor that initialises each instance variable based on user input data.
- getters and setters for each instance field.
- a String toString() method that formats the printing of the object state and returns it.
- a String listCrewMembers() method that returns a String containing all the crew members details.

Phase 3 – BoatClub class:

This is a concrete class that operates between:

- the inheritance hierarchy classes and crew class detailed above (phase 1 and 2) and
- the menu controller (phase 4).

This class stores the following instance fields:

- an ArrayList of Crew: stores all the Crews in the club
- an ArrayList of CrewMember: stores every crew member in the club, regardless of whether they are rowing on a crew or not.
- An ArrayList of TrainerMember: stores every trainerMember in the club, regardless of whether they are cox/coach on any crew.
- The boat club name (max 20 characters, default value is “UNKNOWN”)
- The boat club phone number (max 15 characters, default value is “UNKNOWN”)

This class contains standard getters and setters for each of the five instance fields listed above. It also contains a standard constructor that takes two parameters (one for boat club name and the second for the boat club phone number).

It contains the following methods that operate on the ArrayList of **CrewMember**:

```
//Add an individual crew member
public void addCrewMember(CrewMember member)

// Return a String containing an indexed list of all crew members (note
// that they don’t need to be assigned to a crew)
public String listAllCrewMembers()

// Return a String containing a list of all fully trained CrewMembers
// (note that they don’t need to be assigned to a crew)
public String listAllFullyTrainedCrewMembers()

// Load all crew members from crewmembers.xml
public void loadAllCrewMembers() throws Exception

// Save all crew members to crewmembers.xml
public void saveAllCrewMembers() throws Exception
```

It contains the following methods that operate on the ArrayList of **TrainerMember**:

```
// Add an individual trainer member
public void addTrainerMember(TrainerMember member)

// Return a String containing an indexed list of all trainer members (note
// that they don’t need to be assigned to a crew)
public String listAllTrainerMembers()

// Return a String containing a list of all fully trained TrainerMembers
// (note that they don’t need to be assigned to a crew)
public String listAllFullyTrainedTrainerMembers()
```

```java
    // Load all trainer members from trainermembers.xml
    public void loadAllTrainerMembers() throws Exception

    // Save all trainer members to trainermembers.xml
    public void saveAllTrainerMembers() throws Exception{
```

It contains the following methods that operate on the ArrayList of **Crew**:

```java
    // Add a crew
    public void addCrew(Crew crew)

    // Return a String containing a list of all crews
    public String listAllCrews()

    //Return a String containing a list of crews of a specific size (1, 2 or 4)
    public String listCrewsBySize(int crewSize)

    // Return a String containing a list of all coxed crews (if true passed as
    // a parameter) OR coxless crews (if false passed as a parameter).
    public String listCrewsByCoxStatus(boolean coxed)

    // Load all crews from crews.xml
    public void loadAllCrews() throws Exception

    // Save all crews to crews.xml
    public void saveAllCrews() throws Exception
```

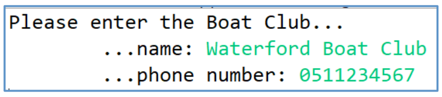
It also contains the following, more complex, methods that operate on the ArrayList of **Crew**:

```java
    // This method returns a Set containing all the crew members assigned to
    // crews (note, if a crew member is on more than one crew, they should only
    // appear once in the list…the Set doesn't store duplicates!).
    public Set<CrewMember> listAllCrewMembersOnCrews()

    // This method returns a Map containing all coaches allocated to crews and
    // the number of crews they are coaching).
    public Map<TrainerMember, Integer> listAllCoachesOnCrews()

    // This method returns a Map containing all coxes allocated to crews and
    // the number of crews they are coxing).
    public Map<TrainerMember, Integer> listAllCoxesOnCrews()

    // You can be a crew member, but you may not be assigned to a crew or you
    // may be assigned to multiple crews.
    // This method counts up the number of members rowing on all crews in the
    // club.  Note: if a crew member is assigned to two or more crews, they
    // should only be counted once (hint:  use Set to eliminate duplicates).
    public int numberCrewMembersOnCrews()

    // You can be a trainer member, but you may not be assigned to a crew or
    // you may be assigned to multiple crews.
    // This method counts the number of trainer members assigned to all
    // crews in the club, as either coach or cox.  Note: if a trainer member is
    // assigned to two crews or more crews, they should only be counted once
    // (hint:  use Set to eliminate duplicates).
    public int numberTrainerMembersOnCrews()
```

Phase 4 – Menu Controller:

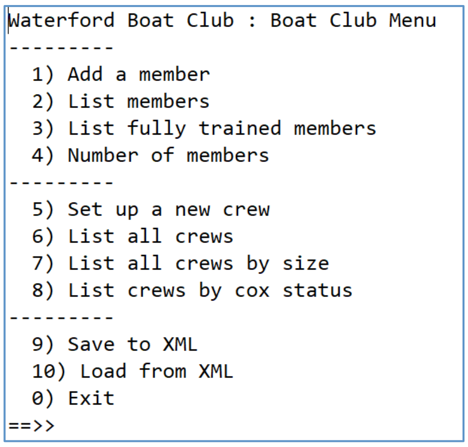
Create a driver class (MenuController) that uses the console I/O to interact with the user. This driver class should create an instance of the GymApi cl ass and allow the user to navigate the system through a series of menus.

On app startup, ask the user to enter the BoatClub name and phone number.



```
Please enter the Boat Club...
        ...name: Waterford Boat Club
        ...phone number: 0511234567
```

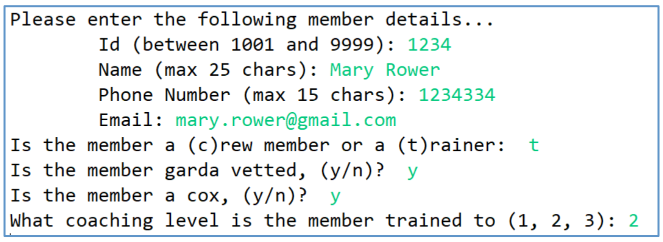
Then display the following menu to the user:



```
Waterford Boat Club : Boat Club Menu
---------
  1) Add a member
  2) List members
  3) List fully trained members
  4) Number of members
---------
  5) Set up a new crew
  6) List all crews
  7) List all crews by size
  8) List crews by cox status
---------
  9) Save to XML
  10) Load from XML
  0) Exit
==>>
```

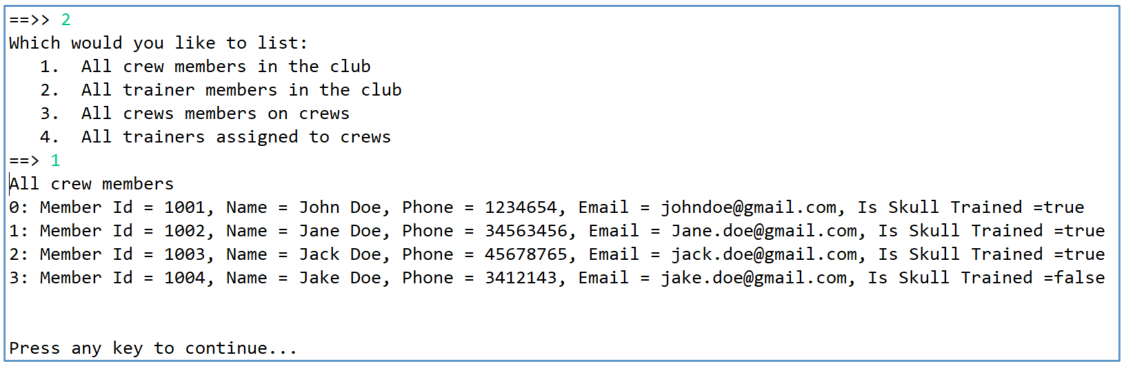
Below are some sample screen shots for the first 8 menu options. Please note that these are prototype screen shots and you are free to design your output in whatever way you choose. Remember also that UX (robustness, usability) is very important here.

Sample output, option 1:



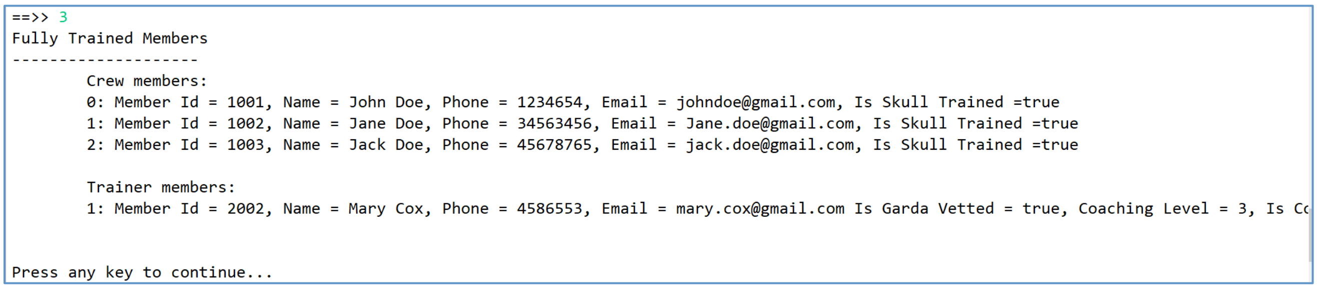
```
Please enter the following member details...
        Id (between 1001 and 9999): 1234
        Name (max 25 chars): Mary Rower
        Phone Number (max 15 chars): 1234334
        Email: mary.rower@gmail.com
Is the member a (c)rew member or a (t)rainer:  t
Is the member garda vetted, (y/n)?  y
Is the member a cox, (y/n)?  y
What coaching level is the member trained to (1, 2, 3): 2
```

Sample output, option 2:



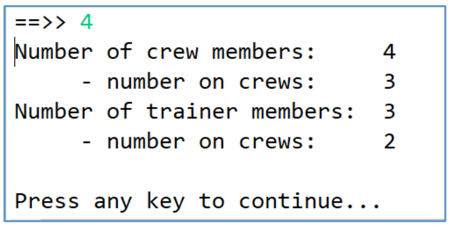
```
==>> 2
Which would you like to list:
   1.  All crew members in the club
   2.  All trainer members in the club
   3.  All crews members on crews
   4.  All trainers assigned to crews
==> 1
All crew members
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true
2: Member Id = 1003, Name = Jack Doe, Phone = 45678765, Email = jack.doe@gmail.com, Is Skull Trained =true
3: Member Id = 1004, Name = Jake Doe, Phone = 3412143, Email = jake.doe@gmail.com, Is Skull Trained =false


Press any key to continue...
```

Sample output, option 3:



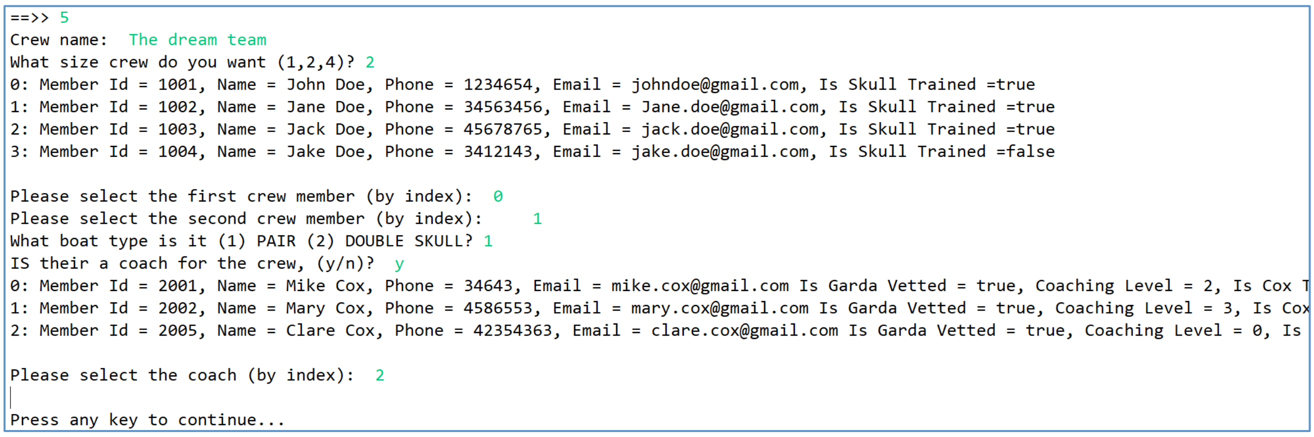
```
==>> 3
Fully Trained Members
--------------------
        Crew members:
        0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
        1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true
        2: Member Id = 1003, Name = Jack Doe, Phone = 45678765, Email = jack.doe@gmail.com, Is Skull Trained =true

        Trainer members:
        1: Member Id = 2002, Name = Mary Cox, Phone = 4586553, Email = mary.cox@gmail.com Is Garda Vetted = true, Coaching Level = 3, Is Co


Press any key to continue...
```

Sample output, option 4:



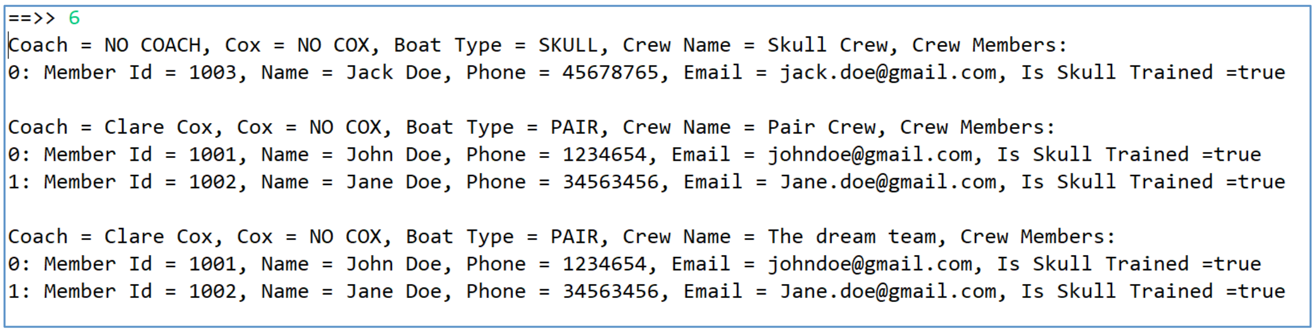
```
==>> 4
Number of crew members:     4
      - number on crews:    3
Number of trainer members:  3
      - number on crews:    2

Press any key to continue...
```

Sample output, option 5:



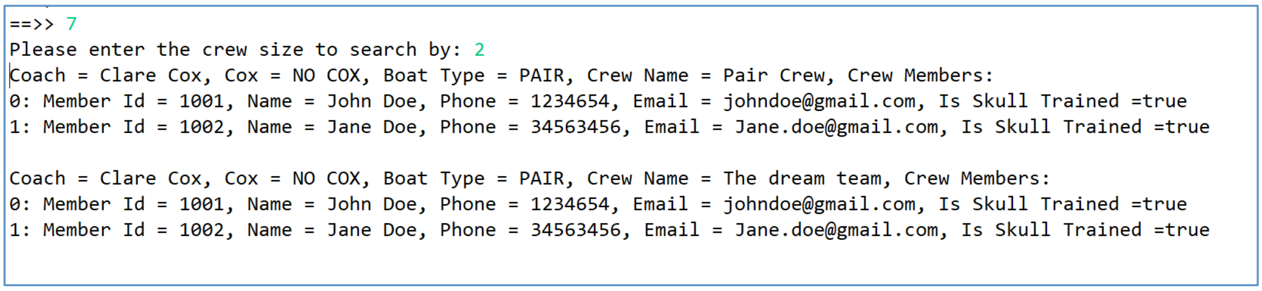
```
==>> 5
Crew name:  The dream team
What size crew do you want (1,2,4)? 2
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true
2: Member Id = 1003, Name = Jack Doe, Phone = 45678765, Email = jack.doe@gmail.com, Is Skull Trained =true
3: Member Id = 1004, Name = Jake Doe, Phone = 3412143, Email = jake.doe@gmail.com, Is Skull Trained =false

Please select the first crew member (by index):  0
Please select the second crew member (by index):     1
What boat type is it (1) PAIR (2) DOUBLE SKULL? 1
IS their a coach for the crew, (y/n)?  y
0: Member Id = 2001, Name = Mike Cox, Phone = 34643, Email = mike.cox@gmail.com Is Garda Vetted = true, Coaching Level = 2, Is Cox T
1: Member Id = 2002, Name = Mary Cox, Phone = 4586553, Email = mary.cox@gmail.com Is Garda Vetted = true, Coaching Level = 3, Is Cox
2: Member Id = 2005, Name = Clare Cox, Phone = 42354363, Email = clare.cox@gmail.com Is Garda Vetted = true, Coaching Level = 0, Is

Please select the coach (by index):  2

Press any key to continue...
```

Sample output, option 6:



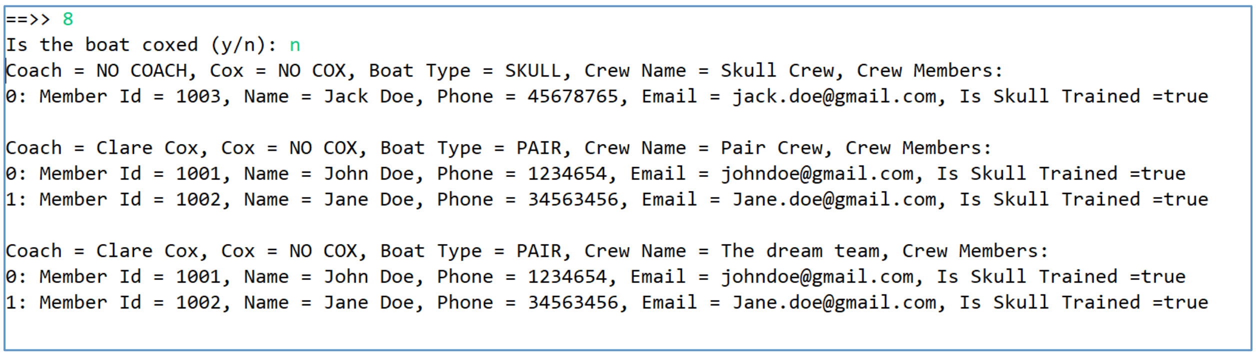
```
==>> 6
Coach = NO COACH, Cox = NO COX, Boat Type = SKULL, Crew Name = Skull Crew, Crew Members:
0: Member Id = 1003, Name = Jack Doe, Phone = 45678765, Email = jack.doe@gmail.com, Is Skull Trained =true

Coach = Clare Cox, Cox = NO COX, Boat Type = PAIR, Crew Name = Pair Crew, Crew Members:
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true

Coach = Clare Cox, Cox = NO COX, Boat Type = PAIR, Crew Name = The dream team, Crew Members:
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true
```

Sample output, option 7:



```
==>> 7
Please enter the crew size to search by: 2
Coach = Clare Cox, Cox = NO COX, Boat Type = PAIR, Crew Name = Pair Crew, Crew Members:
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true

Coach = Clare Cox, Cox = NO COX, Boat Type = PAIR, Crew Name = The dream team, Crew Members:
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true
```

Sample output, option 8:



```
==>> 8
Is the boat coxed (y/n): n
Coach = NO COACH, Cox = NO COX, Boat Type = SKULL, Crew Name = Skull Crew, Crew Members:
0: Member Id = 1003, Name = Jack Doe, Phone = 45678765, Email = jack.doe@gmail.com, Is Skull Trained =true

Coach = Clare Cox, Cox = NO COX, Boat Type = PAIR, Crew Name = Pair Crew, Crew Members:
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true

Coach = Clare Cox, Cox = NO COX, Boat Type = PAIR, Crew Name = The dream team, Crew Members:
0: Member Id = 1001, Name = John Doe, Phone = 1234654, Email = johndoe@gmail.com, Is Skull Trained =true
1: Member Id = 1002, Name = Jane Doe, Phone = 34563456, Email = Jane.doe@gmail.com, Is Skull Trained =true
```

Extra Credit Ideas

As usual with all your programming assignments to date, an extra credit of max 10% will apply. This means that the above phases are marked out of 100%. However, if you lose any marks in that 100%, you can recoup it through the extra credit route (up to a max grade of 100%). For example:

- Example 1: if you scored 94% in the above phases and achieved 8% in the extra credit ideas below, you would achieve an overall grade of 100% (94 + 8 = 102 ➔ max grade being 100).

Example 2: if you scored 56% in the above phases and achieved 10% in the extra credit ideas below, you would achieve an overall grade of 66%.**Idea 1:**

Having reviewed the JUnit test classes from Assignment 1, provide test classes for:

- TrainerMember
- CrewMember
- Crew

Test methods in these classes should include:

- test method for getters and setters.
- test method for constructor/s.

Ensure that you test for both valid and invalid data entry. Also the super class methods in Member can be tested via its subclasses.

Idea 2:

We have included a field in the CrewMember class called skullTrained. However, we haven't really used it in the above specification. If a member is skullTrained, this means they can row in boats that have two oars. Recall that the boat types we have included in this assignment are:

```
SKULL (1 crew member, 2 oars, no cox).

PAIR (2 crew members, 1 oar each, no cox).

DOUBLE SKULL (2 crew members, 2 oars each, no cox).

FOUR (4 crew members, 1 oar each, one cox).

COXLESS FOUR (4 crew members, 1 oar each, no cox).
```

Note that the SKULL and DOUBLE SKULL both have 2 blades (oars) for each crew member. Consider adding in functionality for this field such as when building a crew, only allow those crewMembers that are skullTrained to be added when the boatType is SKULL or DOUBLE SKULL. This would require an additional method in BoatClub.java to return a list of skullTrained members.

Other Ideas:

If you come up any other idea/s, clearly document it/them and how and where it is implemented in either a readme file or in the program comments.