ITSE 1430- Fall 2016
Program Set #4

See C# Grading /Program Guide Sheet for directions and grading/submission information.

Note: Create a console application program for each of the following problems. Do not create a windows application (form).

1. Design a `Ship` class that has the following members:
    - A field for the name of the ship (a string).
    - A field for the year that the ship was built (a string).
    - A constructor and appropriate properties.
    - A `ToString` method that displays the ship's name and the year it was built.

Design a `CruiseShip` class that is derived from the `Ship` class. The `CruiseShip` class should have the following members:

- A field for the maximum number of passengers (an integer).
- A constructor and appropriate properties.
- A `ToString` method that overrides the `ToString` method in the base class. The `CruiseShip` class' `ToString` method should display only the ship's name and the maximum number of passengers.

Design a `CargoShip` class that is derived from the `Ship` class. The `CargoShip` class should have the following members:

- A field for the cargo capacity in tonnage (an integer).
- A constructor and appropriate properties.
- A `ToString` method that overrides the `ToString` method in the base class. The `CargoShip` class' `ToString` method should display only the ship's name and the ship's cargo capacity.

Demonstrate the classes in a program that has a `Ship`. Assign various `Ship`, `CruiseShip`, and `CargoShip` objects to the array elements. The C# program should then step through the array, calling each objects `ToString` method. Place all classes into one file. Output should look similar to below.

Sample Run:

```
Name: Lolipop
Year built: 1960
----------------------------
Name: Disney Magic
Maximum passengers: 2400
----------------------------
Name: Black Pearl
Cargo capacity: 50000 tons
----------------------------
```

Name the program: `TestShipXX.cs`, where XX are your initials.

Note: Create a windows application (form) program for the following problems.

2. Write an OO C# windows application to play the card game Go Fish! -- human against computer.

https://en.wikipedia.org/wiki/Go_Fish

The game starts by shuffling the deck, and dealing 5 cards to each player. The players then alternate turns. On each player's turn, the game should ask him/her for the rank of a card to ask for. If the other player has any cards of that rank, they are transferred to the current player's hand. If the other player does not have any cards of that rank, the current player must "Go Fish" by drawing. If, at the end of the turn, a player has four cards of the same rank, they form a book, and are removed from the player's hand. The game ends when the deck is empty: the player with the greatest number of books wins.

The program should indicate when a player completes a book of four cards of the same rank: It's player 2's turn The game ends when there are no more cards in the deck. The program should determine which player has more books, and announce the winner of the game.

Implementation:

Some classes that will be needed are specified below and additional classes can be used. It is up to the student to figure out what the responsibilities (methods) of each class will be. Those classes are:

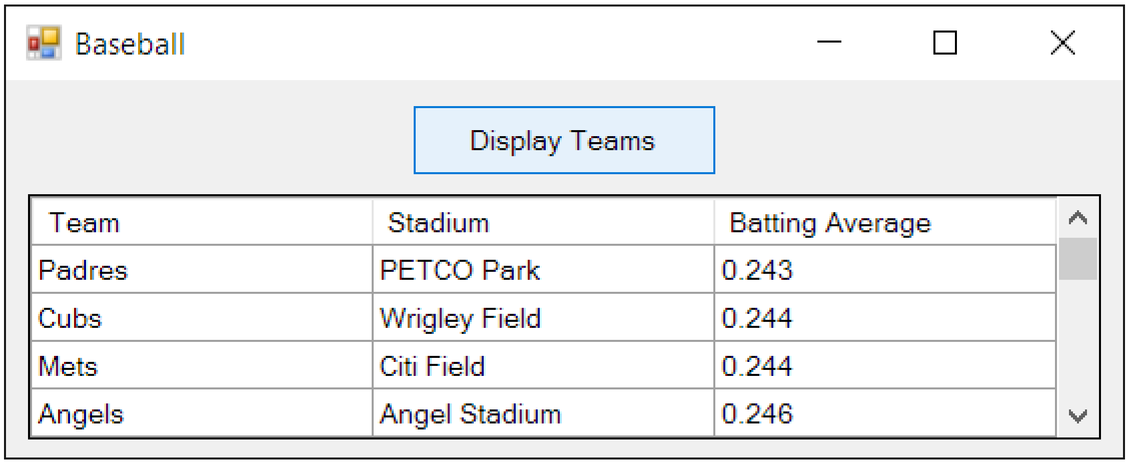
- `Fish`(object) - to represent the entire game. Creating a `Fish` object should start the game.
- `Card`-simulates one playing card (a suit and value)
- `Deck`-simulate a deck of 52 cards
- `Player` (object) - to represent a single player.

Also, no object should directly examine or alter the variables of any other object; all access should be done by talking to (calling methods of) the other object. The student should figure out what the various classes and objects should be and what their instance variables and their methods should be. There is no single "correct" design; any reasonable design will do. The better the design, the easier the programming will be. Use the objects defined to play a game of Go Fish! The computer player must play legally, but does not need to play well. Try to make the computer player "smart," so the human player does not always win. Use appropriate controls (labels, buttons, etc.). in your application. Card deck provided by your instructor in a file named `deck.zip`. Place all classes into one file. Output should be user friendly.

Name the application: `OOGoFishXX`, where XX are your initials.

3. Write a C# windows application to display in a `DataGridView` control the names of all the MLB teams, their home stadiums, and the teams' batting averages. Records should be sorted in ascending order by

batting averages. The form title should be named `Baseball`. The database will be in Access (`Baseball.accdb`) and provided by your professor. Microsoft Access 2013/2016 is needed for this problem. Output should look similar to below.



Team	Stadium	Batting Average
Padres	PETCO Park	0.243
Cubs	Wrigley Field	0.244
Mets	Citi Field	0.244
Angels	Angel Stadium	0.246

Name the application: `DBBattingAvgsXX`, where `XX` are your initials.

Note: Create a console application program for the following problem. Do not create a windows application (form).

4 (**). Write an OO C# program that simulates a telephone answering machine that records missed incoming calls. A missed call is any is any incoming call that the user did not answer in person. For each missed call, store the time of the call, telephone number of origin, and name of caller if the name is available. For unlisted numbers, set the name to "private caller". Provide the following features:

- Numbers are recalled in the order they arrive.
- Up to `10` numbers are recorded. When the eleventh call comes in, it is stored and the oldest call is deleted so that no more than `10` numbers are ever recorded.
- When the user presses the missed-calls button, the telephone displays the number of calls bumped (stored but then deleted because there were more than `10` missed calls). Next, the telephone displays the number of origin of each missed call that is still stored, one at a time.
- After each number display, the user can select:
    - To delete the call
    - To go on to the next missed call
    - To display the call details (number, caller name, and time)
- After each number-detail display, the user can select:
    - To delete the call
    - To go on to the next call
- If there are no missed calls, at the end of the display of missed calls, the telephone prints

```
       "End of Missed Calls"
```
Write a class to represent an incoming call with fields to hold the number, name of caller, and time of call. Write a test class that stores several numbers, simulates the user pressing the missed call button, and finally prints the entire collection of stored calls. Make sure that if more than one call comes from the same person or number of origin, all calls are recorded. Create your own numbers, names, and times for testing. Do not use a text file. Place all classes into one file. Output should be user friendly.

Name the program: `TestAnswerMachineXX.cs`, where XX are your initials.