CSI-155 Object-Oriented Programming

Rules:

* **Turn in 2 things: the entire zipped project + word document (not Form1.cs)**
* **Copy the content of Form1.cs to the Word document. However you need to organize the word document such that all the code pertaining to a question be together. Every answer is labeled and in ordered sequence.**

**(I am not asking for to turn in Form1.cs, I am asking for a word document)**

* **Follow the stated requests above to avoid being penalized by up to 10 points off your final exam score**

**Final Exam : Create a new windows project (your name should be part of the project name)**. questions 1 and 2 should be answered as comments

1. Define an abstract class, using your own words and provide a complete statement (for full credit)
2. What is the difference between an abstract class and an interface. Use your own words and provide complete statements (for full credit)
3. Add an interface IFillShape. This interface should define a method Fill() that takes no parameters and returns a bitmap. (if you define a method that does not fit the requested requirements, it will be considered wrong). Any child class must create a bitmap according to the dimensions of the shape, fill the bitmap with the child shape and color
4. Define an abstract class \_2DShape that inherits the IFillShape interface and defines 4 fields \_x, \_y (which represents the upper left corner of the shape), \_side (which defines the measurement of all the sides), and \_color (which define the color to use to fill the shape.
5. Define a class Square that inherits \_2DShape. As stated above this class is to create a bitmap according to its measurements, fill its shape with the given color and return the bitmap from the Fill() method. Likewise define a class Circle that inherits \_2DShape and have its Fill() method return a bitmap that contains a Circle filled with the given color.
6. In Form1 create a List where you can save both Square and Circle objects. Populate the List with 3 Squares of 3 Circles with different colors, locations and dimensions, then call a Display method that displays all the shapes to a Panel using the Graphics class method DrawImage
7. Write a method ‘private void ReverseQueue( Queue<int> queue)’ that takes a Queue<int> and reverse it. This method returns a void. Also this method is not to use any temporary Queue, Stack, List, Array or any other collection.
8. Write a method ‘ private void ReverseStack(Stack<int> stack) ‘ this method takes stack and returns a void. This method is to reverse the stack (you may use one temporary collection)
9. Write a method ‘ private void DisplayOdds(int num)’ that takes a value (could be even or odd) the method is to display the first 10 odd values larger than num. Now using similar logic, rewrite the method as a recursive method ‘ RecursiveDisplayOdds..’
10. Add a class Student as shown below and in Form1 define the list as shown below. Create a Dictionary to hold students object and using the ID as the key, then define a method that sequences through the list of students and populate the dictionary with all the students in the list. Call this method from Form1()
11. Add a button to read a student ID from user, then get the corresponding student from the dictionary.

public class Student

{

public string First { get; set; }

public string Last { get; set; }

public int ID { get; set; }

public List<int> Scores;

}

static List<Student> students = new List<Student>

{

new Student {First="Svetlana", Last="Omelchenko", ID=111, Scores=

newList<int> {97, 92, 81, 60}},

new Student {First="Claire", Last="O'Donnell", ID=112, Scores= new List<int> {75, 84, 91, 39}},

new Student {First="Sven", Last="Mortensen", ID=113, Scores= new List<int> {88, 94, 65, 91}},

new Student {First="Cesar", Last="Garcia", ID=114, Scores= new List<int> {97, 89, 85, 82}},

new Student {First="Debra", Last="Garcia", ID=115, Scores= new List<int> {35, 72, 91, 70}},

new Student {First="Fadi", Last="Fakhouri", ID=116, Scores= new List<int> {99, 86, 90, 94}},

new Student {First="Hanying", Last="Feng", ID=117, Scores= new List<int> {93, 92, 80, 87}},

new Student {First="Hugo", Last="Garcia", ID=118, Scores= new List<int> {92, 90, 83, 78}},

new Student {First="Lance", Last="Tucker", ID=119, Scores= new List<int> {68, 79, 88, 92}},

new Student {First="Terry", Last="Adams", ID=120, Scores= new List<int> {99, 82, 81, 79}},

new Student {First="Eugene", Last="Zabokritski", ID=121, Scores= new List<int> {96, 85, 91, 60}},

new Student {First="Michael", Last="Tucker", ID=122, Scores= new List<int> {94, 92, 91, 91} }

};