NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
ISLAMABAD CAMPUS
COMPUTER PROGRAMMING (CS103) - FALL 2018
ASSIGNMENT-5

Due Date: November 19, 2018 (05:00 pm)

Instructions:

1. *Make sure that you read and understand each and every instruction.*
2. *Plagiarism is strongly forbidden and will be very strongly punished. If we find that you have copied from someone else or someone else has copied from you (with or without your knowledge) both of you will be punished. You will be awarded straight zero in this assignment or all assignments.*
3. *Submit a single '.zip' file for your assignment and each problem solution must be provided in a folder and three separate CPP file. For instance, you must name the folder containing solution of first problem as 'q1' and second as 'q2' and so on.*
4. *You have to write each class in header and .cpp files Name the files as per your question. For instance, folder named as q1 will have three files. q1.h, q1.cpp and q1main.cpp.No code will be accepted without these files.*
5. *Note: You have to follow the submission instructions to the letter. Failing to do so can get a zero in assignment.*

Q1) Write a superclass called Shape (as shown in the diagram), which contains:

- Two instance variables color (String) and filled (boolean).
- Two constructors: a default constructor that initializes the color to "green" and filled to true, and a constructor that initializes the color and filled to the given values.
- Getter and setter for all the instance variables.
- A toString() method that returns "A Shape with color of xxx and filled/Not filled".

Write two subclasses of Shape called Circle and Rectangle, as shown in the diagram.
The Circle class contains:

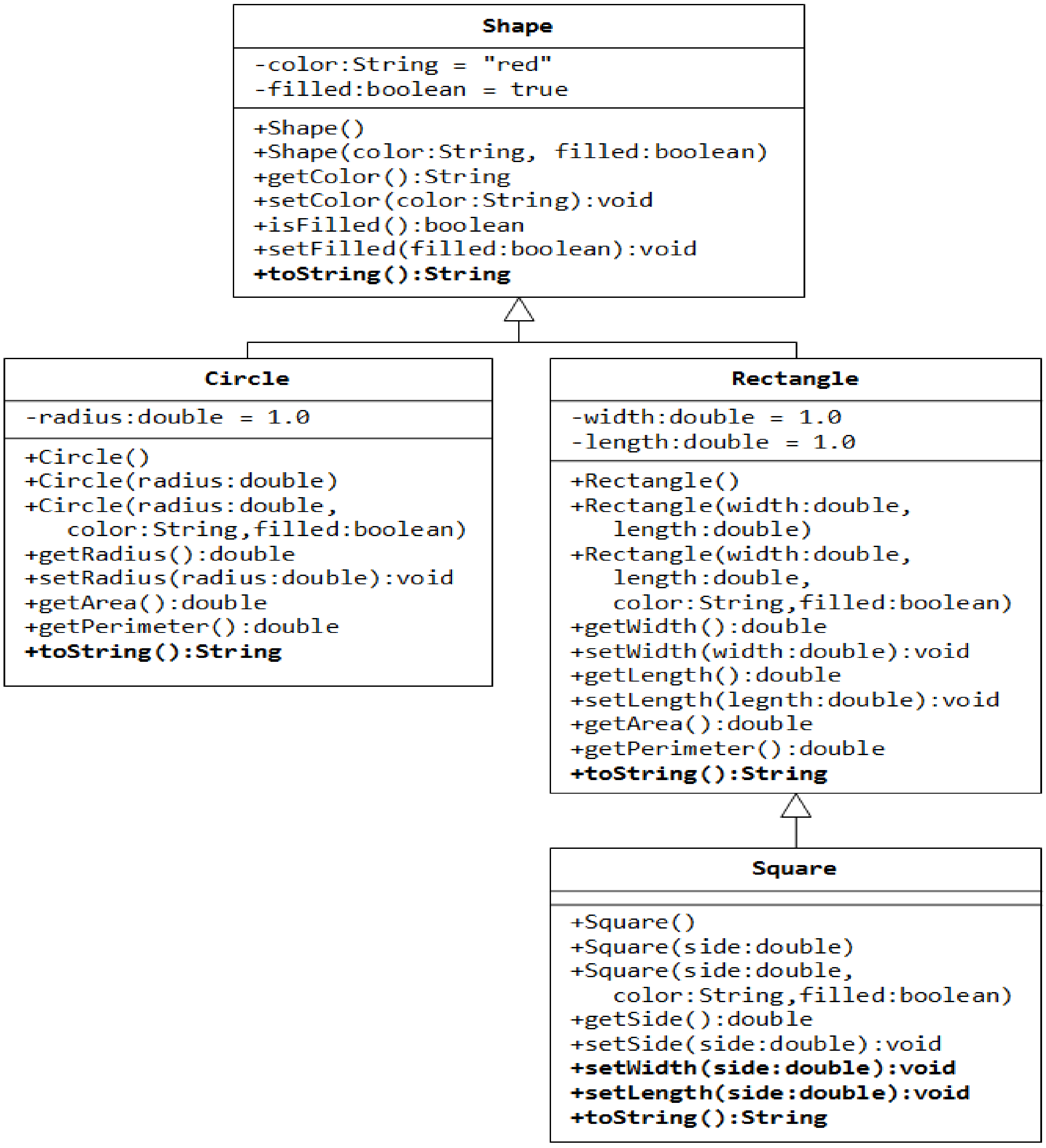
- An instance variable radius (double).
- Three constructors. The default constructor initializes the radius to 1.0.
- Getter and setter for the instance variable radius.
- Methods getArea() and getPerimeter().
- Redefine the toString() method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

The Rectangle class contains:

- Two instance variables width (double) and length (double).
- Three constructors. The default constructor initializes the width and length to 1.0.
- Getter and setter for all the instance variables.
- Methods getArea() and getPerimeter().
- Redefine toString() method inherited, to return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

- Provide the appropriate constructors (as shown in the diagram).
- Redefine the toString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.
- Do you need to override the getArea() and getPerimeter()? Try them out.
- Redefine the setLength() and setWidth() to change both the width and length, so as to maintain the square geometry.


Shape
-color:String = "red"
-filled:boolean = true
+Shape()
+Shape(color:String, filled:boolean)
+getColor():String
+setColor(color:String):void
+isFilled():boolean
+setFilled(filled:boolean):void
+toString():String
Circle
-radius:double = 1.0
+Circle()
+Circle(radius:double)
+Circle(radius:double, color:String,filled:boolean)
+getRadius():double
+setRadius(radius:double):void
+getArea():double
+getPerimeter():double
+toString():String
Rectangle
-width:double = 1.0
-length:double = 1.0
+Rectangle()
+Rectangle(width:double, length:double)
+Rectangle(width:double, length:double, color:String,filled:boolean)
+getWidth():double
+setWidth(width:double):void
+getLength():double
+setLength(legnth:double):void
+getArea():double
+getPerimeter():double
+toString():String
Square
+Square()
+Square(side:double)
+Square(side:double, color:String,filled:boolean)
+getSide():double
+setSide(side:double):void
+setWidth(side:double):void
+setLength(side:double):void
+toString():String

Q2) Clash of the titans: The provided main program simulates a Fight between a dragon and a hydra. The hierarchy of classes that model the creatures of this game are missing and you are asked to provide them.
The class Creature A creature is characterized by:

- its name (a constant string);
- its level (an integer);
- its number of health points (health status; an integer);
- its force (an integer);
- its position (position , also an integer; for simplicity, our game takes place in 1D; Can be 2D as well an object of Point class).

These attributes must be accessible to classes deriving from Creature.
The methods defined for this class are:

- a constructor allowing the initialization of the name, level, health points, force and position of the creature using the values passed as parameters, in this order; the constructor accepts zero as default value for the position;
- a method bool alive() returning true if the creature is alive (number of health points greater than zero) or false otherwise;
- a method AttackPoints returning the number of attack points that can be inicted by the creature to others; the value is computed as the level multiplied by the force if the creature is alive, or zero otherwise;
- a method Move(int), which does not return anything and adds the integer passed as parameter to the position of the creature;
- a method GoodBye() which does not return anything and displays the message (English: <name> is no more!): using strictly this format. <name> is the name of the creature;
- a method Weak, which does not return anything and substracts the number of points passed as parameter from the number of health points of the creature, if it is alive; if the creature dies, its number of health points is set to zero and the method GoodBye is called;
- a method Display(), which does not return anything and displays informations about the creature using strictly the following format:
  <name>, level: <level>, health_status: <points>, force: <force>,
  Attacking Points: <attack>, position: <position>
  <name> is the name of the creature, <level> is its level, <points> is its number of health points, <force> is its force, <attack> is its number of attack points and <position> is its position.

The class Dragon A Dragon is a Creature. It has as specific characteristic the range of its flame (flamerange an integer). Its specific methods are:

- a constructor which initializes its name, level, number of health points, the force, the range of the flame and the position of the dragon using the values passed as parameters, in this order; the constructor accepts zero as default value for the position;
- a method Fly(int pos) which does not return anything and allows the dragon to move to the given position pos;
- a method BlowFlame(Creature&) which does not return anything and simulates what happens when the dragon blows its flame towards another Creature:
    1. if the dragon and the creature are both alive and if the creature is in range of its flame, the dragon inflicts its attack points as damage to the creature; the creature weakens by the number of attack points; The dragon also weakens; it loses of ' health points, with of

' being the distance between the dragon and the creature (the further the dragon has to blow, the more it weakens);

2. if after this epic fight the dragon is still alive and the creature dies, the dragon increases in level by one unit;

The creature is in the range of the flame of the dragon if the distance between them is smaller or equal to the range of the flame (you should use the function distance we provide).

The class Hydra A Hydra is a Creature. It has specific characteristics the length of its neck (necklength, an integer) and the dose of poison it can inject in an attack (poisondose, an integer). Its specific methods are:

- a constructor which initializes its name, level, number of health points, force, the length of its neck, the poison dose and the position using the values passed as parameters, in this order; the constructor accepts zero as default value for the position;
- a method InjectPoison(Creature&) which does not return anything and simulates what happens when the hydra poisons another Creature:
    1. if the hydra and the creature are alive and the creature is in range of the head of the hydra, then the hydra inflicts damage to the creature; the creature weakens by the number of attack points of the hydra plus its dose of poison;
    2. if at the end of the fight the creature is no longer alive, the hydra increases in level by one unit;

The creature is “in range of the head of the hydra" if the distance the creature and the hydra is smaller or equal to the length of the neck of the hydra.

The function Fight (fight) takes as parameters a dragon and a hydra. It allows:

- the hydra to poison the dragon;
- and the dragon to blow on the hydra.

Execution Examples:

The example of output below corresponds to the provided program.

```
Dragon red, level: 2, health_status: 10, force: 3, points of attack: 6, position: 0 is preparing for fight with:
Hydra evil, level: 2, health_status: 10, force: 1, points of attack: 2, position: 42

1st Fight :
the creatures are not within range, so can not Attack.
After the Fight :
Dragon red, level: 2, health_status: 10, force: 3, points of attack: 6, position: 0
Hydra evil, level: 2, health_status: 10, force: 1, points of attack: 2, position: 42

Dragon has flown close to Hydra :
Dragon red, level: 2, health_status: 10, force: 3, points of attack: 6, position: 41

Hydra moves :
Hydra evil, level: 2, health_status: 10, force: 1, points of attack: 2, position: 43
```

```
2nd Fight :
  + Hydra inflicts a 3-point attack on dragon
      [ level (2) * force (1) + poison (1) = 3 ] ;
  + Dragon inflicts a 6-point attack on Hydra
      [ level (2) * force (3) = 6 ] ;
  + during his attack, dragon loses two additional points
       [ corresponding to the distance between dragon and hydra : 43 - 41 = 2 ].
After the Fight :
Dragon red, level: 2, health_status: 5, force: 3, points of attack: 6, position: 41
Hydra evil, level: 2, health_status: 4, force: 1, points of attack: 2, position: 43
Dragon moves by one step :
Dragon red, level: 2, health_status: 5, force: 3, points of attack: 6, position: 42

3rd Fight :
  + Hydra inflicts a 3-point attack on dragon
      [ level (2) * force (1) + poison (1) = 3 ] ;
  + Dragon inflicts a 6-point attack on Hydra
      [ level (2) * force (3) = 6 ] ;
  + during his attack, dragon lost 1 additional life point.
       [ corresponding to the distance between dragon and hydra : 43 - 42 = 1 ] ;
  + Hydra is defeated and the dragon rises to level 3
Hydra evil is no more!
After the Fight :
Dragon red, level: 3, health_status: 1, force: 3, points of attack: 9, position: 42
Hydra evil, level: 2, health_status: 0, force: 1, points of attack: 0, position: 43

4th Fight:
   when one creatures is defeated, nothing happpens.
After the Fight :
Dragon red, level: 3, health_status: 1, force: 3, points
of attack: 9, position: 42
Hydra evil, level: 2, health_status: 0, force: 1, points of attack: 0, position: 43
```

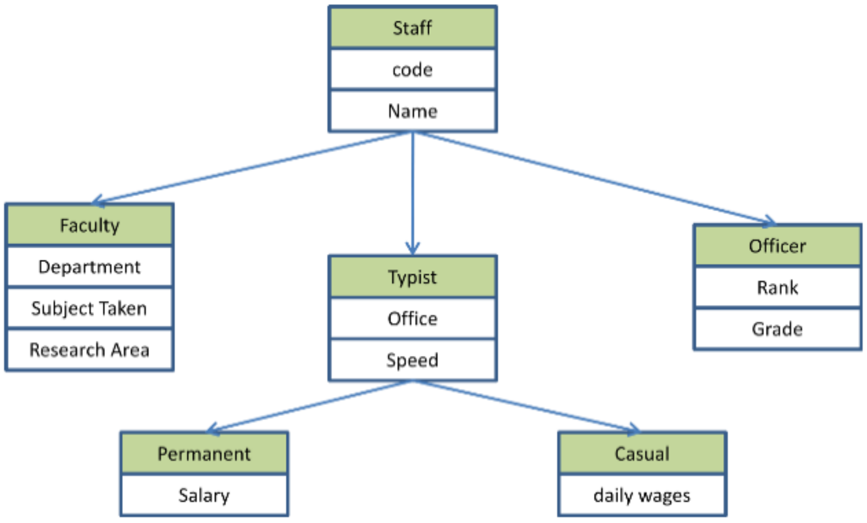
Q3) Imagine a publishing company that markets both book and audiocassette versions of its works.

- **Task 1:** Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the client, and a putdata() function to display its data.
- **Task 2:** Add a class sales that holds an array of three floats so that it can record the dollar sales of a particular publication for the last three months. Include a getdata() function to

get three sales amounts from the client, and a putdata() function to display the sales figures. Alter the book and tape classes so they are derived from both publication and sales. An object of class book or tape should input and output sales data along with its other data.

- **Task 3:** Assume that the publisher decides to add a third way to distribute books: on computer disk, for those who like to do their reading on their laptop. Add a disk class that, like book and tape, is derived from publication. The disk class should incorporate the same member functions as the other classes. The data item unique to this class is the disk type: either CD or DVD.
- **Task 4:** Suppose you want to add the date of publication for both books and tapes. From the publication class, derive a new class called publication2 that includes this member data. Then change book and tape so they are derived from publication2 instead of publication. Make all the necessary changes in member functions so the user can input and output dates along with the other data. For the dates, you have to write the date class which stores a date as three ints, for month, day, and year.

Q4) An eduational institute wishes to maintain data of its employees. The database is divided into a number of classes where hierarchical relationships are shown in figure below.



The figure also shows the minimum information required for each class. Specify all the classes and functions and retrieve individual information as and when required.

The above hierarchy does not include the education information of the staff. It has been decided to add information to Faculty and Officer which will help the management in decision making with regard to training, promotion etc. Add another class education that holds two pieces of educational information, namely, highest qualification in general education and highest professional qualification. This class should be inherited by the classes teacher and officer.