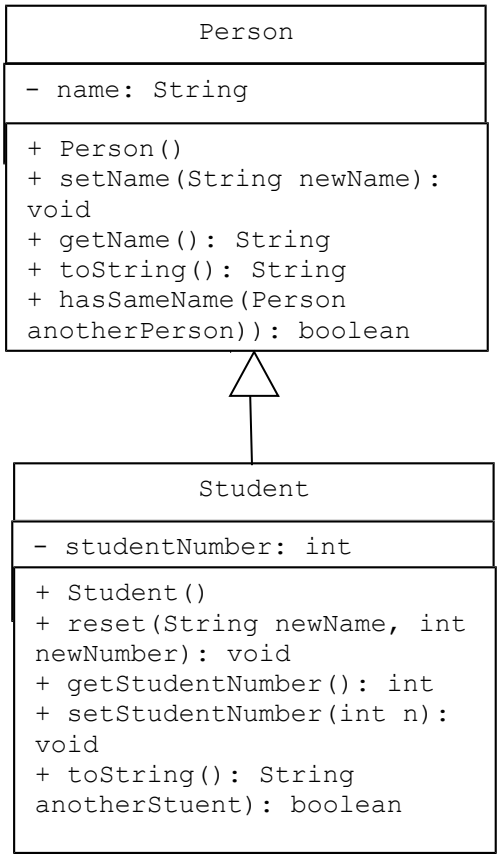
CSCI 1101 – Winter 2013
Laboratory No. 5

This lab focuses on developing object-oriented programs with inheritance.

Exercise 1: The following figure shows a UML diagram in which the class `Student` is inherited from the class `Person`



Implement the two classes. Write a demo program that tests all the methods in the `Student` class as well as the inherited methods.

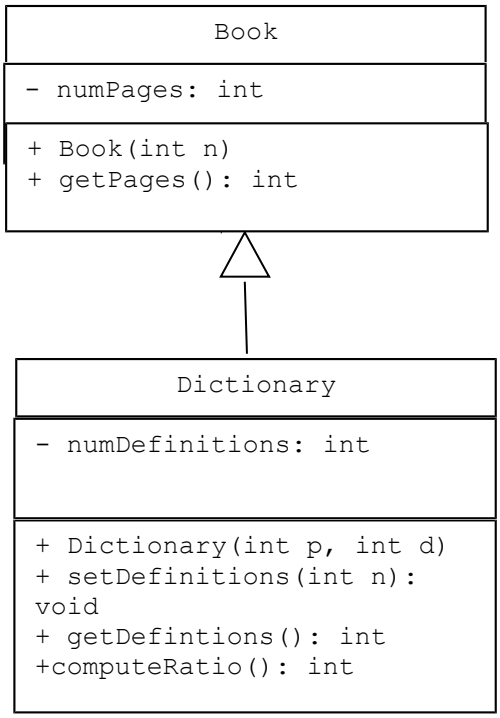
Note: In the `toString` method in the `Student` class, use the method `getName()` to get the name.

Can you modify the above `toString` method so that it uses the `toString` method from the `Person` class? Hint: keyword `super`.

Exercise 2: Create a class called `SchoolKid` that is the base class for children at a school. It should have attributes for the child's name and age, the name of the child's teacher, and a greeting (`String` type). It should have appropriate accessor and mutator methods for each of the attributes.

Derive a class `ExaggeratingKid` from `SchoolKid`, as described in the previous exercise. The new class should override the accessor method for the age, reporting actual age plus 2. It also should override the accessor method for the greeting, returning the child's greeting concatenated with the words "I am the best".

Exercise 3 (Revision from Test Practice): Implement the following UML diagram



Note: the constructor in the `Dictionary` class sets the number of pages and number of definitions. The method `computeRatio` returns the number of definitions per page.

Test the classes.

Exercise 4: The following is a class called `ShapeBasics` that can be used for drawing simple shapes on the screen using keyboard characters. This class will draw an asterisk on the screen as a test. It is not intended to create a "real" shape, but rather to be used as a *base* class for other classes of shapes.

```
public class ShapeBasics
{
              //offset indicates how far it is indented from the left
              //edge of the screen
              private int offset;
              // shape character
              public final static char SHAPE_CHAR = '*';

              public ShapeBasics()
              {
                     offset=0;
              }
              public ShapeBasics(int offset)
              {
                     this.offset = offset;
              }
              public void setOffset(int offset)
              {
                     this.offset = offset;
              }
              public int getOffset()
              {
```

```
        return this.offset;
    }
    //static method skipLines skips the given number of lines
    //down from the current one
    public static void skipLines(int numLines)
    {
        for(int i=0; i<numLines; i++)
            System.out.println();
    }
    //static method skipSpaces that skips the given number of spaces
    public static void skipSpaces(int numSpaces)
    {
        for(int i = 0; i<numSpaces; i++)
            System.out.print(' ');
    }
    //method draw draws a single star
    public static void draw()
    {
        System.out.print(SHAPE_CHAR);
    }
    //method draw(num) draws a given number of shapes
    public static void draw(int numShapes)
    {
        for(int i = 0; i < numShapes; i++)
            draw();
    }
    //method drawHere draws the shape beginning at the current line
    public void drawWithOffset()
    {
        skipSpaces(offset);
        draw();
    }
    //method drawHere draws the given # of shapes beginning at current line
    public void drawWithOffset(int numShapes)
    {
        skipSpaces(offset);
        draw(numShapes);
    }
}
```

a) Create a class `LineShape` that extends the `ShapeBasics` class. This class should have the following:
   - A no args constructor
   - A method to set the given offset
   - A method `drawVertical` that draws a vertical line of a given length starting at the offset
   - A method `drawHorizontal` that draws a horizontal line of a given length starting at the offset

   The class does not have any instance variables.

b) Create a class `RectangleShape` that extends the `LineShape` class. The class should have the following:
   - Instance variables height and width
   - Constructor that sets the offset, height and width
   - Method `drawHere` that draws the rectangle starting at the given offset and of dimensions width and height
   - Method `drawSides` that draws the two sides of the rectangle. This method is used by the `drawHere` method.

Test all the three classes.