DePaul University
College of Computing and Digital Media

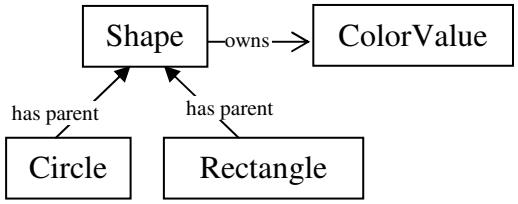
CSC 211 - Programming in Java I

Final Exam Assignment

Description:
In this final exam assignment, we're going do the following:

- We will create classes needed to represent the data and selected behaviors of Geometric Shapes. We will not be displaying these of course, just representing them in our objects.

- Object Schema:



- Remember the following:
  - Proper Java Syntax, naming, format, conventions, error checking, etc..
  - Inheritance & Polymorphism
    - What it is & how it works
    - Accepting constructor parameters need to call the parent constructor
    - Calling parent constructors & parent versions of methods
  - Aggregation & Containment (One object owning another object)
    - Owner uses reference data member to point to the object it owns
    - Owned object is allocated ("new") in the owner's constructor
    - Calling methods of owned objects via "get" methods

Instructions:
1) Create a new project. *You will be provided with a "Driver.java" on the COL site in the Documents section.*

2) Create a java package for your "domain" classes – the classes that represent the important elements in our application.

3) In the package you just made, create a class called "ColorValue". This class will contain the 3 RGB (Red, Green, Blue) values needed to display a color *(remember - we will not be displaying anything, we just need to store these values).*
   - This class should contain 3 data members (int's) that can hold the ColorValue's red, green & blue values. The related "set" methods should print an error message and exit the program if the value passed in is less than 0, or greater than 255.
   - The ColorValue class' constructor should accept 3 "int" parameters for the red, green & blue color values. Use these 3 parameters to set the corresponding data members.
   - Create a "toString" method that returns a String containing the ColorValue object's data in an easy to read format.
     Example:
     ```
     ColorValue RGB: 200, 100, 200
     ```

4) Create an *abstract* parent class called "Shape". This *abstract* class represents a generic geometric "shape". Subclasses (child classes) will be created later that contain data & behaviors for specific types of geometric shapes.
   - This class should contain a String data member for the shape's id. The id cannot be null or empty. The modifier for this data member should print an error message and exit the program if the id passed in is null or it is an empty String.
   - This class should also contain a "ColorValue" data member. The "set" method should print an error message and exit the program if the ColorValue object passed in is null.

- The Shape class' constructor should accept a String parameter for the shape's id, as well as 3 int's for the red, green & blue color values (needed by the ColorValue constructor). The constructor should do the following:
  1. Set the shape's id.
  2. Set the ColorValue to a *new ColorValue* object created using the red, green & blue color values passed in.
- We will also need 2 public *abstract* methods for this class that will be implemented in the child classes. We will implement these in "Shape" subclasses:
  1. A "double" method that will calculate the shape's area.
  2. A "double" method that will calculate the shape's perimeter
- Create a "toString()" method that returns a String containing the Shape object's data in an easy to read format. All double values should be displayed out to 1 decimal place.
  Example:

```
Shape Id:       Rectangle1
ColorValue RGB: 200, 100, 200
Area:           48.0
Perimeter:      32.0
```

5) Next, create a class called "Rectangle" that is a *child* class to the "Shape" class. Shape is the parent, Rectangle is the child. This class represents a Rectangle geometric shape.
   - This class should define 2 data member (doubles) one to hold the Rectangle's *length*, and one to hold its *width*. The related "set" methods should print an error message and exit the program if the values passed in are less than or equal to zero.
   - The Rectangle class' constructor should accept a String parameter for the shape's id, int's for the red, green & blue color values, and doubles for the length & width. The id, red, blue, green, and color value parameters are needed by the parent constructor (`super(…);`). The length & width data members should be set here using the related data parameters.
   - We will need implementations of the 2 methods that the parent class defined as abstract previously in the Shape parent class (up at Step 4, bullet #4):
     - A "double" method that will calculate the shape's area.
       - This method should return the Rectangle's area (Rectangle Area: *Length * Width*)
     - A "double" method that will calculate the shape's perimeter
       - This method returns the Rectangle's perimeter (Rectangle Perimeter: *(Length * 2) + (Width * 2)*)
   - Create a "toString()" method that returns a String containing all the Rectangle object's data in an easy to read format. All double values should be displayed out to 1 decimal place.
     Example:

```
Shape Id:        Rectangle1
ColorValue RGB:  100, 70, 111
Area:            32.0
Perimeter:       24.0
Length:          4.0
Width:           8.0
```

6) Now, create a class called "Circle" that is a child class to the "Shape" class. Shape is the parent, Circle is the child. This class represents a Circle geometric shape.
   - This class should contain a data member (double) to hold the Circle's *radius*. The related "set" method should print an error message and exit the program if the value passed in is less than or equal to zero.
   - The Circle class' constructor should accept a parameter for the shape's id, int's for the red, green & blue color values and a double for the radius. The id, red, blue, green, and color value parameters are needed by the parent constructor (`super(…);`). The radius data member should be set using the related parameter.
   - We will need implementations of the 2 methods that the parent class defined as abstract previously in the Shape parent class (up at Step 4, bullet #4):
     1) A "double" method that will calculate the shape's area.
        - This method should return the Circle's area (Circle Area: *Pi * Radius*^{2})
     2) A "double" method that will calculate the shape's perimeter
        - This method returns the Circle's perimeter (Circle Perimeter: *2 * Pi * Radius*)
   - *Note "Pi" is pre-defined in Java as a constant in the "Math" class. You can reference that constant in your code by saying: "`Math.PI`". (i.e., `x = x & Math.PI;`).*
   - Create a "toString()" method that returns a String containing all the Circle object's data in an easy to read format. All double values should be displayed out to 1 decimal place.

Example:

```
Shape Id:         Circle1
ColorValue RGB:   77, 200, 128
Area:             95.0
Perimeter:        34.6
Radius:           5.5
```

7) Done!
   Expected Outputs from "main" in Driver.java:

```
Creating Shapes...
Shape Id:         Circle1
ColorValue RGB:   50, 75, 225
Area:             153.9
Perimeter:        44.0
Radius:           7.0

Shape Id:         Rectangle1
ColorValue RGB:   200, 200, 100
Area:             48.0
Perimeter:        32.0
Length:           12.0
Width:            4.0

Shape Id:         Circle2
ColorValue RGB:   42, 255, 255
Area:             314.2
Perimeter:        62.8
Radius:           10.0

Shape Id:         Rectangle2
ColorValue RGB:   44, 22, 50
Area:             16.0
Perimeter:        20.0
Length:           2.0
Width:            8.0

Shape Id:         Circle3
ColorValue RGB:   20, 30, 40
Area:             78.5
Perimeter:        31.4
Radius:           5.0

Shape Id:         Rectangle3
ColorValue RGB:   100, 111, 70
Area:             32.0
Perimeter:        24.0
Length:           4.0
Width:            8.0

Shape Id:         Circle4
ColorValue RGB:   77, 128, 200
Area:             95.0
Perimeter:        34.6
Radius:           5.5

Shape Id:         Rectangle4
ColorValue RGB:   200, 200, 255
Area:             27.1
Perimeter:        22.5
Length:           3.5
Width:            7.8

Shape Id:         Circle5
ColorValue RGB:   0, 70, 80
Area:             46,759.5
Perimeter:        766.5
Radius:           122.0

Shape Id:         Rectangle5
ColorValue RGB:   44, 22, 78
Area:             4,000.0
Perimeter:        280.0
Length:           40.0
Width:            100.0

Done Creating Shapes
```

You will be provided with a "Driver.java" on the COL site in the Documents section.

Submission

- This assignment is due by 9:00 pm on Monday, June 7th. *NO Late submissions will be accepted.*
  - Submission is done via the COL site, just like you have done with your assignment submissions.

Extra Credit

The Final Exam extra credit opportunities and their related point values are described below. *These are optional – you are not required to implement these in order to get full credit on your final exam.*

NOTE: These Extra Credit opportunities below are *cumulative*. You MUST do the "Level 1" Extra Credit *before* you do the "Level 2" Extra Credit. You MUST do the "Level 2" Extra Credit *before* you do the "Level 3" Extra Credit.

Since these Extra Credit opportunities are cumulative, Extra Credit features must be *complete and fully functional* to be counted for credit. Do not move to the next Extra Credit level unless the previous level is complete and fully functional.

Level 1:

Square (3 pts)

- Create an additional "Shape" subclass called "Square". Square ***MUST be a sub-class of the "Rectangle" class.***
- The constructor for "Square" MUST accept these – and only these - parameters: `(String idIn, int redIn, int greenIn, int blueIn, double widthIn)`
- Update the provided Driver.java "main" by adding the creation of 2 Square objects
- Expected "toString" output:

```
Shape Id:       Square1
ColorValue RGB: 234, 12, 198
Area:           144.0
Perimeter:      48.0
Length:         12.0
Width:          12.0
```

Level 2:

Cube (5 pts)

- Create an additional "Shape" subclass called "Cube" (representing a 3-dimensional cube). Cube ***MUST be a sub-class of the "Rectangle" class.***
- The constructor for "Cube" MUST accept these – and only these - parameters: `(String idIn, int redIn, int greenIn, int blueIn, double lengthIn, double widthIn, double heightIn)`
- Add 2 additional double methods to the Cube class – "calculateVolume()" and "calculateSurfaceArea()". These will return the volume and surface area of the Cube. The results of these calculations should be present in the "toString" output. (The existing area and perimeter methods should still work, using the length & width to perform these calculations).
- Update the provided Driver.java "main" by adding the creation of 2 Cube objects
- Expected "toString" output:

```
Shape Id:       Cube1
ColorValue RGB: 123, 45, 211
Area:           72.0
Perimeter:      36.0
Length:         12.0
Width:          6.0
Height:         8.0
Volume:         576.0
Surface Area:   432.0
```

Level 3:

Cube (5 pts)

- Create an additional "Shape" subclass called "Cylinder" (representing a 3-dimensional cylinder). Cylinder ***MUST be a sub-class of the "Circle" class.***
- The constructor for "Cylinder" MUST accept these – and only these - parameters: `(String idIn, int redIn, int greenIn, int blueIn, double radiusIn, double heightIn)`
- Add 2 additional double methods to the Cylinder class – "calculateVolume()" and "calculateSurfaceArea()". These will return the volume and surface area of the Cylinder. The results of these calculations should be present in the "toString" output. (The existing area and perimeter methods should still work, using the length & width to perform these calculations).
- Update the provided Driver.java "main" by adding the creation of 2 Cylinder objects
- Expected "toString" output:

```
Shape Id:       Cylinder1
ColorValue RGB: 211, 145, 111
Area:           254.5
Perimeter:      56.5
Radius:         9.0
Height:         11.0
Volume:         2799.2
Surface Area:   1130.9
```