CSCI 1130
Assignment 4
Wandering Space Object
2/20/19 (orig), last revision 3/7/19
See D2L Submission Folder for Due Date

Revision 3/5/19: Removed “Vector” interface. Will introduce “interfaces” at a later time. Renamed class MyVector to Vector.
Revision 3/7/19: Typo corrected for constructor spec for "Vector"
Revision 3/7/19: Added details for Circle's "drawOn" method

The references below apply to the "Course Notes". See link on our D2L homepage.

Refer to Part II:

- Chapter 4 -- Object Composition
- Chapter 5 -- Class Inheritance

In this assignment we'll build a wandering space object. The graphics framework has been provided. Your job will be to build the objects that it will use to render the graphics.

All instance variables should be private and all methods public.

This document contains descriptions of classes that you will build. Code each class in it's own JAVA file.

It is strongly suggested, but not required, to sketch out diagrams of your classes/objects before coding.

Note that for Java graphics the y axis is unorthodox oriented downward (e.g. top to bottom) while the x axis is “standard” (left to right).

Attached ZIP File

There is a ZIP file attached to this assignment named “Space Object.zip”. Open that up and copy the “Space Object” directory on your local computer. There are a number of JAVA completed files in that directory that you will need. Your files should go in the same directory (or package if you are using an IDE).

You will use these three provided classes as you code:

Frame
GraphicsApp

You will use the Frame and GraphicsApp classes as-is. There is an associated "README.TXT" file with the JAVA files.

The following pages describe the classes for you to code.

Class Name: SpaceObject

This is the main (top) class that clients of your system will use. The SpaceObject class is composed of three child objects as shown below. This is the baby that will be floating around in space. Code the class as described below

Instance Variables:

Name	Type
ss	SimpleShape
dir	GraphicsDirection
color	Color

Constructors:

Name	Parameters	Notes
SpaceObject	SimpleShape newSS and GraphicsDirection newDir	Set ivars
SpaceObject	SimpleShape newSS	Set ivar ss from method param and set ivar dir using GraphicsDirection's default (empty) constructor

Instance Methods :

Name	Parameters	Return Type	Description
getColor	None	Color	Return ivar
setColor	Color newColor	None	Set ivar
drawOn	Graphics g	None	Let our shape (ivar) draw using it's own “drawOn” method (use our color too).
moveBy	int scalarX and int scalarY	None	Construct a Vector using the two params. Hint: our “dir” child will help converting. Then let our shape move.
getBoundary	None	Frame	Return our frame.
reverseDirectionX	None	None	Reverse x direction
reverseDirectionY	None	None	Reverse y direction

Class Name: XYOrderedPair

Very simple class that really just "holds" and x and y value.
Both are ints.
Subclasses may be Point, Vector, ...

Instance Variables:

Name	Type
x	int
y	int

Constructors:

Name	Parameters	Notes
XYOrderedPair	int newX and int newY	Set ivars

Instance Methods :

Name	Parameters	Return Type	Description
getX	None	int	Return ivar
getY	None	int	Return ivar
setX	int newX	None	Set ivar
setY	int newY	None	Set ivar

Class Name: Point

A simple class that models an two dimensional point in the Cartesian system. It has an x and y value.

This class is a subclass of XYOrderedPair.

Instance Variables: None

Constructors:

Name	Parameters	Notes
Point	int newX and int newY - You will want to call the super's constructor as the first line in this constructor method like this: super(newX, newY);	Call superclasses constructor

Instance Methods :

Name	Parameters	Return Type	Description
moveBy	Vector v	None	Move this point *psuedocode*: set x = x + v.x set y = y + v.y (you'll have to use getters and setters getX, setX, getY, setY)

Class Name: Vector

Vector class:

Models a two dimensional (x, y) vector.
A vector has a mangnitude and direction.

Because a vector has a direction, it may be used to "move" objects (see Point).

This class is a subclass of XYOrderedPair.

Instance Variables: None

Constructors:

Name	Parameters	Notes
Vector	int newX and int newY See “Point” class for more info	Call superclasses constructor

Instance Methods :

Name	Parameters	Return Type	Description
plus	Vector v	None	Vector addition *psuedocode*: x = x + v.x y = y + v.y (use getters and setters getX, setX, getY, setY)
multiply	int k	None	Vector multiplication (by scalar) *psuedocode*: x = x * k y = y * k (use getters and setters)
reverse	None	None	Reverse direction of this vector (i.e. negate x and negate y)

Class Name: SimpleShape

This is an abstract class that represents a general shape. It is abstract so it does not know if it is a Circle, Line, Rectangle, etc.

Instance Methods :

Name	Parameters	Return Type	Description
drawOn	Graphics g and Color newColor	None	abstract method
moveBy	Vector v	None	abstract method
getBoundary	None	Frame	abstract method

Class Name: Circle

This class inherits from SimpleShape (i.e. is a sublcass of SimpleShape)

Instance Variables:

Name	Type	Description
c	Point	center point
int	r	radius

Constructors:

Name	Parameters	Notes
Circle	Point newC and int newR	Set ivars

Instance Methods :

Name	Parameters	Return Type	Description
drawOn	Graphics g and Color newColor	None	Send messages to "g": setColor, drawOval, and fillOval as described below.
Pertinent method headers from the Graphics class: • `void drawOval(int x, int y, int width, int height)` • `void fillOval(int x, int y, int width, int height)` • `void setColor(Color c)` **Where x is the left x of the shape** **Where y is the top y of the shape** **And where width and height are the dimensions of the shape.**			
moveBy	Vector v	None	Let center do the move (i.e. "delegate to child" / pass method to child object).
getBoundary	None	Frame	Return shape outer boundary using center and radius

Class Name: GraphicsDirection

This class provides the direction for the space object. It keeps track of if the space object is going forward or backward in the x and y direction.

Instance Variables:

Name	Type	Description
xForward	boolean	true means moving forward in x dir
yForward	boolean	true means moving forward in y dir (which is down in Java graphics)

Constructors:

Name	Parameters	Notes
GraphicsDirection	None	Set both ivars to true

Instance Methods :

Name	Parameters	Return Type	Description
reverseX	None	None	Reverses x direction. To invert a boolean, use: !myBoolean
reverseY	None	None	Reverses y direction.
convertScalarX	int scalar	int	Convert scalar to vector quantity, multiply the method param by this: (this.xForward ? 1 : -1)
convertScalarY	int scalar	int	Convert scalar to vector quantity, multiply the method param by this: (this.yForward ? 1 : -1)

Class Name: Rectangle

5% of the score will be on this problem.
This problem is to add another shape class. This shape should be a Rectangle. So for this problem you would code up Rectangle.java (your Circle.java will be a good guide).

Class Name: Line

5% of the score will be on this problem.
This problem is to add another shape class. This shape should be a Line. (your Circle.java will be a good guide).

The Space Graphics

When you are done with your objects, we're all set for the graphics. To render the graphics, comple and run “GraphicsApp” in the usual way.

Submit Instructions

- When submitting, submit the entire “Space Object” directory with all the provided files and your new files. Please submit only this one directory.
- You may **yes** have extra files in the directory like CLASS, BAT, SH, etc files, but please **not** extra JAVA files.
- You may be working in a different directory tree. Before submitting, simply copy the files from your directory tree to the “Space Object” directory.
- Copy the "Space Object" directory into your ZIP file
- An improper submit makes grading much more difficult and could receive up to -5 points
- Name your ZIP file per these instructions in the course notes here:
  "General Guidance -- Course Instructions -- ZIP File for Submit"
- Submit your ZIP file in the D2L proper assignment folder

Assignment Scoring

Possible Points

Submit Schedule	Points	Note
If Submitted By Due Date	100	Due Date is given in D2L
If Submitted By **Late** Due Date	70	Late Due Date is one week after due date
If Submitted **after** "Late Due Date"	0	