*Question, Programming* In BlueJ,

In this project you are going to complete a class called Rocket. Once completed this class will produce a simple graphical representation of a moving rocket in the Shapes window as shown in the animation.

Before you start, take a good look at the incomplete Rocket class provided.

Do not worry about the delay() method for now as it will be explained in part d.

Note that you will ***not*** need to change the Circle, Triangle or Square classes in this question, but you should remind yourself about them briefly before starting.

The square, triangle and circle that make up our representation of a rocket will be created in the Workspace, and then we will store references to them in a Rocket object.

a.The Rocket class has a constructor which creates a rocket object using references to the nose, body and jet, given as arguments to the constructor. However, at present only the nose is in the correct starting position. If you construct a Rocket at this point you will see the (correctly positioned) nose and the (incorrectly positioned) body and jet in the Graphical Display (Figure 2).

* To make the body and jet line up correctly with the nose, you need to complete the methods getBodyXPos(), getBodyYPos(), getJetXPos() and getPosJetYPos() in the Rocket class.

When you complete the helper methods in i. to iv. below, your methods must not use fixed numbers for the positions or dimensions of the rocket parts. Rather, you must use appropriate methods from the classes Circle, Triangle and Square.

* + Complete the method getBodyXPos() so that it returns an appropriate value, relative to the position of the nose.
  + Complete the method getBodyYPos() so that it returns an appropriate value, relative to the position of the nose.
  + Complete the method getJetXPos() so that it returns an appropriate value, relative to the position of the body.
  + Complete the method getJetYPos() so that it returns an appropriate value, relative to the position of the body.

Complete the method moveRocketBy(). The method should move the rocket **upwards** by the amount given by the method's argument. In other words, the method needs to change the y-positions of the parts of the rocket.

To do this, first set the nose position, and then set the body and jet positions relative to the nose and body, respectively. You must make use of the methods you wrote for part a., as well as other methods.

Test your method moveRocketBy() in the OUWorkspace, using the test code included in the README.TXT file for this project, to check that the rocket moves up and down correctly.

Complete the methods pulse1(), pulse2() and pulse3(), as described below.

* + i.The pulse1() method should do the following:
  + set the diameter of jet to 6
  + reset the position of the jet (because its size is now different, and we want it to stay centred underneath the body)
  + set the colour of jet to red.
  + ii.The pulse2() method should do the following:
  + set the diameter of jet to 12
  + reset the position of the jet (because its size is now different, and we want it to stay centred underneath the body)
  + set the colour of jet to orange.
  + iii.The pulse3() method should do the following:
  + set the diameter of jet to 24
  + reset the position of the jet (because its size is now different, and we want it to stay centred underneath the body)
  + set the colour of jet to red.

Complete the ignition() method. The method will simulate firing up the rocket's jet engines by repeatedly changing the size and colour of the jet using the methods pulse1(), pulse2() and pulse3()..

Because the Graphical Display will change very quickly as your method goes through each expansion / colour change step, we have provided a method called delay() that you can use to pause execution for a number of milliseconds after each change in the colour and size of the jet. You do not have to know how this method works, all you need to do is to include the message-send

this.delay(200);

after each pulse. If this does not slow down the display enough for you to see the changes, try increasing the argument ‘200’ to a bigger number until you can see the changes happening.

Use a loop to repeat the following five times:

* + send a pulse1() message to the receiver
  + delay execution for 200 milliseconds
  + send a pulse2() message to the receiver
  + delay execution for 200 milliseconds
  + send a pulse3() message to the receiver
  + delay execution for 200 milliseconds.

To test your completed method, execute the statements in part a. to create an instance of Rocket.

Then execute the following line of code, on its own, after the others.

* + r.ignition();

If your method is correct, you should see in the Graphical Display, the jets expanding and changing colour a total of five times.

You will now complete the method animateRocket(int speed).

This method animates the rocket, as follows.

As we have a rocket movement method (part b.) and three pulse methods (part c.), we can perform four different animations in a loop.

The first time the loop executes:

* + send a pulse1() message to the receiver
  + delay execution for 200 milliseconds.

The second time the loop executes:

* + send a pulse2() message to the receiver
  + delay execution for 200 milliseconds.

The third time the loop executes:

* + send a pulse3() message to the receiver
  + delay execution for 200 milliseconds.

The fourth time the loop executes:

* + move the rocket by the specified number of units (speed) upwards in one step.

Write a loop that repeats this sequence 100 times.

Complete the animateRocket() method.

In the final method launch() a dialogue box is displayed to prompt the user to enter a positive integer representing the speed (number of units) with which they want the rocket to move upwards.

You may assume that the user enters characters that represent valid positive integers in the dialogue box.

* + You will need to convert the string returned by the dialogue box into an integer.
  + If the user has entered a number that would move the tip of the nose past the top of the Graphical Display (i.e. animating the rocket would cause the yPos of the nose to become less than zero at some stage), a dialogue box should alert the user that the rocket will not launch as the number entered is too large.
  + Otherwise the method should send an ignition() message to the receiver, followed by an animateRocket() message with the user-specified speed as the message's argument.