**Incomplete Rocket class**

**import ou.\*;**

**/\*\***

**\* Class Rocket allows a representation of a rocket to be created in the Shapes**

**\* Window, and for it to move up the screen.**

**\***

**\* This class is incomplete.**

**\***

**\* @author (M250 Course Team)**

**\* @version (1.0)**

**\*/**

**public class Rocket**

**{**

**private Triangle nose; // represents the rocket's nose cone**

**private Square body; // represents the rocket's body**

**private Circle jet; // represents the blast from the rocket's engine**

**/\*\***

**\* Constructor for objects of class Rocket**

**\*/**

**public Rocket(Triangle t, Square s, Circle c)**

**{**

**//first, we store references to the workspace shape objects**

**this.nose = t;**

**this.body = s;**

**this.jet = c;**

**//sets the initial positions of the nose.**

**//The other parts need to be set relative to these positions.**

**this.nose.setXPos(50);**

**this.nose.setYPos(300);**

**//sets the body relative to the nose, using the helper methods**

**this.body.setXPos(getBodyXPos());**

**this.body.setYPos(getBodyYPos());**

**//The jet is invisible to begin with, because it uses**

**//the background colour of WHITE, but you can use BLACK**

**//to help you see it while testing**

**this.jet.setColour(OUColour.WHITE);**

**this.jet.setDiameter(10);**

**//sets the jet position relative to the body, using the helper methods**

**this.jet.setXPos(getJetXPos());**

**this.jet.setYPos(getJetYPos());**

**}**

**private int getBodyXPos()**

**{**

**//to be written in Q1(a)(i)**

**}**

**private int getBodyYPos()**

**{**

**//to be written in Q1(a)(ii)**

**}**

**private int getJetXPos()**

**{**

**//to be written in Q1(a)(iii)**

**}**

**private int getJetYPos()**

**{**

**//to be written in Q1(a)(iv)**

**}**

**/\*\***

**\* Moves the nose of the receiver**

**\* by anInt units.**

**\* Moves other rocket components relative to the**

**\* position of the nose.**

**\*/**

**public void moveRocketBy(int anInt)**

**{**

**}**

**/\*\***

**\* Sets the diameters of the receiver's jets to**

**\* 6 and sets their colour to red.**

**\*/**

**public void pulse1()**

**{**

**//to be written in Q1(c)(i)**

**}**

**/\*\***

**\* Sets the diameters of the receiver's jets to**

**\* 12, decrements their xPos by 3 and sets**

**\* their colour to orange.**

**\*/**

**public void pulse2()**

**{**

**//to be written in Q1(c)(ii)**

**}**

**/\*\***

**\* Sets the diameters of the receiver's jets to**

**\* 24, decrements their xPos by 6 and sets**

**\* their colour to red.**

**\*/**

**public void pulse3()**

**{**

**//to be written in Q1(c)(iii)**

**}**

**/\*\***

**\* Simulates the ignition of the rocket's jets**

**\*/**

**public void ignition()**

**{**

**//to be written in Q1(d)**

**}**

**/\*\***

**\* Moves the entire rocket by speed units upwards, repeated**

**\* 100 times, animating the jet as it goes.**

**\*/**

**public void animateRocket(int speed)**

**{**

**//to be written in Q1(e)**

**}**

**/\*\***

**\* Prompts the user to enter the number of units they want the**

**\* rocket to move upwards at a time (speed)**

**\***

**\* If the number of units provided would eventually cause the tip of the**

**\* nose to go past the top of the Graphical Display, the user**

**\* is informed via a dialogue box that the rocket will not launch.**

**\***

**\* Otherwise the rocket launches as required.**

**\*/**

**public void launch()**

**{**

**//to be written in Q1(f)**

**}**

**/\*\***

**\* Causes execution to pause by time number of milliseconds**

**\*/**

**public void delay(int time)**

**{**

**try**

**{**

**Thread.sleep(time);**

**}**

**catch (Exception e)**

**{**

**System.out.println(e);**

**}**

}

}