SpaceShip.java

```
import java.awt.*;

public class SpaceShip2 extends Polygon{

	//determines the speed of the ship
	private double xVelocity = 0, yVelocity = 0;

	//obtain the game board width and height
	int gBWidth = Game_Board.boardWidth; //x
	int gBHeight = Game_Board.boardHeight; //y

	//center of SpaceShip
	private double centerX = gBWidth/2;
	private double centerY = gBHeight/2;

	//x&y coordinates of the ship wrt center this is to allow rotation of the ship
	public static int[] polyXArray = {-13,14,-13,-5,-13};
	public static int[] polyYArray = {-15,0,15,0,-15};

	//width and height of ship
	private int shipWidth = 27, shipHeight = 30; //for collision detection make a rectangle area around ship

	//upper left hand corner of ship
	private double uLeftXPos = getXCenter() + this.polyXArray[0];
	private double uLeftYPos = getYCenter() + this.polyYArray[0];

	//user can rotate the ship
	private double rotationAngle=0, movingAngle=0;

	//-------------------------------------------------------------------------------------------------

	public SpaceShip2() {
		super(polyXArray,polyYArray,5); //five pointed space ship
	}//end constructor
	//---------------------------------------------------------------------------------------------
	public double getXCenter() { return centerX;}
	public double getYCenter() { return centerY;}
	public void setXCenter(double xCent) {this.centerX = xCent;}
	public void setYCenter(double yCent) {this.centerY = yCent;}
	public void increaseXPos(double incAmt) {this.centerX += incAmt;}
	public void increaseYPos(double incAmt) {this.centerY += incAmt;}

	//getters and setters for upper left corner of ship which is the ships handle
	public double getuLeftXPos() {return uLeftXPos;}
	public double getuLeftYPos() {return uLeftYPos;}
	public void setuLeftXPos(double xULPos) {this.uLeftXPos = xULPos;}
	public void setuLeftYPos(double yULPos) {this.uLeftYPos = yULPos;}

	//getters and setters for uppper left hand of ship
	public int getShipWidth() {return shipWidth;}
	public int getShipHeight() {return shipHeight;}

	//getter and setters to increase and decrease ship velocity
	public double getXVelocity() {return xVelocity;}
	public double getYVelocity() {return yVelocity;}
	public void setXVelocity(double xVel) {this.xVelocity = xVel;}
	public void setYVelocity(double yVel) {this.yVelocity = yVel;}
	public void increaseXVelocity(double xVelInc) {this.xVelocity += xVelInc;}
	public void increaseYVelocity(double yVelInc) {this.yVelocity += yVelInc;}
	public void decreaseXVelocity(double xVelDec) {this.xVelocity -= xVelDec;}
	public void decreaseYVelocity(double yVelDec) {this.yVelocity -= yVelDec;}

	//set and allow for increase of ship movement angle
	public void setMovingAngle(double moveAngle) {this.movingAngle = moveAngle;}
	public double getMovingAngle() {return movingAngle;}
	public void increaseMovingAngle(double moveAngle) {this.movingAngle += moveAngle;}

	//cos of angle gives opposite value of angle. angle * Math.PI / 180 converts degrees to radians
	//cos of angle provides x pos and sin of angle provides y pos
	public double shipXMoveAngle(double xMoveAngle) {
		return(double)(Math.cos(xMoveAngle*(Math.PI/80)));
	}//end shipXMoveAngle

	public double shipYMoveAngle(double yMoveAngle) {
		return (Double) (Math.sin(yMoveAngle*(Math.PI/180)));
	}//end shipXMoveAngle

	public double getRotationAngle() {return rotationAngle;}

	public void increaseRotationAngle() {
		if(getRotationAngle() >= 355) { rotationAngle = 0;}//end if
		else {rotationAngle += 5;}//end else
	}//end increaseRotationAngle

	public void decreaseRotationAngle() {
		if(getRotationAngle() < 0) { rotationAngle = 355;}//end if
		else {rotationAngle -= 5;}//end else
	}//end decreaseRotationAngle

	public Rectangle getBounds() {
		return new Rectangle(getShipWidth()-14, getShipHeight()-15, getShipWidth(),getShipHeight());
	}

	public void move() {

		this.increaseXPos(this.getXVelocity());

		if(this.getXCenter() < 0)
		{this.setXCenter(gBWidth);}
		else if(this.getXCenter() > gBWidth)
		{this.setXCenter(0);}

		this.increaseYPos(this.getYVelocity());

		if(this.getYCenter() < 0){
		 this.setYCenter(gBHeight);
		}
		else if(this.getYCenter() > gBHeight) {
		 this.setYCenter(0);
		}
	}//end move()

}//end class
```

GAME_BOARD.JAVA

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.util.ArrayList;
import java.util.concurrent.ScheduledThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
import javax.swing.JComponent;
import javax.swing.JFrame;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
```

```java
import java.awt.geom.AffineTransform;

//set up the KeyListner for space ship inside the Game_Board constructor


public class Game_Board extends JFrame{

	private static final long serialVersionUID = 1L;
	public static int boardWidth = 1000;
	public static int boardHeight = 1000;

	//--- rotation of space ship on game board
	public static boolean keyHeld = false;//rotation
	public static int keyHeldCode;


	public static void main(String[] args) {
		new Game_Board(); //create game board
	}//end main


	//-----------------------------------------------------------------------------------------------
	//redraw screen over and over again
	class RepaintTheBoard  implements Runnable{

		Game_Board theBoard;
		public RepaintTheBoard(Game_Board theBoard) {
			this.theBoard = theBoard;
		}//end constructor

		@Override
		public void run() {
			theBoard.repaint();
		}//end run

	}//end class
	//-------------------------------------------------------------------------------------------------------
	//constructor
	public Game_Board() {
		//define defaults
		this.setSize(boardWidth, boardHeight);
		this.setTitle("Asteroid Crunch");
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		addKeyListener(new KeyListener() {
	//listens to key strokes
			@Override
			public void keyTyped(KeyEvent e) {
				// TODO Auto-generated method stub
			}

			@Override
			public void keyPressed(KeyEvent e) {
				//--- for rotation is key being held down to rotate ship d 68 held down rotate clockwise
				if(e.getKeyCode() == 87) {
					System.out.println("Forward"); //w is forward
					keyHeldCode = e.getKeyCode();
					keyHeld = true;
				} else if (e.getKeyCode() == 83) {
					System.out.println("Backwards"); //s is backward
					keyHeldCode = e.getKeyCode();
					keyHeld = true;
				} else if (e.getKeyCode() == 68) {
					System.out.println("Rotate Clockwise"); //d rotate clockwise
					keyHeldCode = e.getKeyCode();
```

```
                                        keyHeld = true;
                                } else if (e.getKeyCode() == 65) {
                                        System.out.println("Rotate CounterClockwise"); //a rotate counterclockwise
                                        keyHeldCode = e.getKeyCode();
                                        keyHeld = true;
                                }

                        }//end keyPressed

                        @Override
                        public void keyReleased(KeyEvent e) {
                                //key codes w:87 a:65 s:83 d:68
                                //these are the key codes I will be looking for to operate this space ship
                                /***
                                if(e.getKeyCode() == 87) {
                                        System.out.println("Forward"); //w is forward
                                } else if (e.getKeyCode() == 83) {
                                        System.out.println("Backwards"); //s is backward
                                }; //end if TEST
                                ***/
                                keyHeld = false;
                        }//end KeyReleased


                });//end KeyListener



                GameDrawingPanel gamePanel = new GameDrawingPanel();//create first version of the board

                this.add(gamePanel, BorderLayout.CENTER);

                ScheduledThreadPoolExecutor executor = new ScheduledThreadPoolExecutor(5); //core pool size keep these threads in pool even if idle

                //redraw the game board every 20 milliseconds
                executor.scheduleAtFixedRate(new RepaintTheBoard(this), 0L, 20L, TimeUnit.MILLISECONDS);

                this.setVisible(true);
        }//end constructor
        //---------------------------------------------------------------------------------------------------

}//end Game_Board
//---------------------------------------------------------------------------------------------------


class GameDrawingPanel extends JComponent{
        private static final long serialVersionUID = 1L;
        public ArrayList<Rock> rocks = new ArrayList<Rock>();
        int[] polyXArray = Rock.sPolyXArray; //shape of a rock
        int[] polyYArray = Rock.sPolyYArray; //shape of a rock

        //---- space ship
        //SpaceShip theShip = new SpaceShip(); //we create the ship here we are also creating board and asteroids here
        SpaceShip2 theShip = new SpaceShip2(); //new and improved space ship

        int width = Game_Board.boardWidth;
        int height = Game_Board.boardHeight;

        //constructor create 50 rocks and store in an array
        public GameDrawingPanel() {
                for(int i=0; i < 50; i++) {
                        int randomStartXPos = (int) (Math.random() * (Game_Board.boardWidth - 40) + 1); //create the rock

                        int randomStartYPos = (int) (Math.random() * (Game_Board.boardHeight - 40) + 1);
                        //add rock to array
                        rocks.add(new Rock(Rock.getpolyXArray(randomStartXPos), Rock.getpolyYArray(randomStartYPos), 13, randomStartXPos, randomStartYPos));
                        Rock.rocks = rocks; //sync the local and ?

                }//end for
        }//end constructor


        //---------------------------------------------------------------------------------------------------------

        public void paint(Graphics g) {
                Graphics2D graphicSettings = (Graphics2D)g;
                AffineTransform identity = new AffineTransform();
                graphicSettings.setColor(Color.BLACK);
                graphicSettings.fillRect(0, 0, getWidth(), getHeight());
                graphicSettings.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
                graphicSettings.setPaint(Color.WHITE);
                //draw them on the screen
                for(Rock rock: rocks) {
                        rock.move();
                        graphicSettings.draw(rock);
                }//end for


                //--- space ship
                if(Game_Board.keyHeld == true && Game_Board.keyHeldCode == 68)
                {//forward rotation
                        //rotate ship by 10 degrees
                        //SpaceShip.rotationAngle += 10;
                        theShip.increaseRotationAngle();
                        System.out.println("Ship Angle: " + theShip.getRotationAngle());



                } else if(Game_Board.keyHeld == true && Game_Board.keyHeldCode == 65)
                {//reverse rotation
                        //SpaceShip.rotationAngle -= 10;
                        theShip.decreaseRotationAngle();
                        System.out.println("Ship Angle: " + theShip.getRotationAngle());

                } else if(Game_Board.keyHeld == true && Game_Board.keyHeldCode == 87)
                {//accelerate forward direction
                        //steering the ship - align ship moving direction with it's current angle
                        theShip.setMovingAngle(theShip.getRotationAngle()); //we're rotating ship in proper direction
                        theShip.increaseXVelocity(theShip.shipXMoveAngle(theShip.getMovingAngle()) * 0.1);//accelerate in that direction
                        theShip.increaseYVelocity(theShip.shipYMoveAngle(theShip.getMovingAngle()) * 0.1);
                }//end if else
                 else if(Game_Board.keyHeld == true && Game_Board.keyHeldCode == 83)
           {//decelerate the ship or reverse direction
                //steering the ship - align ship moving direction with it's current angle
                theShip.setMovingAngle(theShip.getRotationAngle()); //we're rotating ship in proper direction
                theShip.decreaseXVelocity(theShip.shipXMoveAngle(theShip.getMovingAngle()) * 0.1);
                theShip.decreaseYVelocity(theShip.shipYMoveAngle(theShip.getMovingAngle()) * 0.1);
           }//end if else

                theShip.move();
                graphicSettings.setTransform(identity);
                graphicSettings.translate(theShip.getXCenter(), theShip.getYCenter());
                //graphicSettings.rotate(Math.toRadians(theShip.getRotationAngle()));
                graphicSettings.rotate(Math.toRadians(theShip.getRotationAngle()));


                //graphicSettings.translate(Game_Board.boardWidth/2, Game_Board.boardHeight/2); //set ship in center of screen
                //graphicSettings.rotate(Math.toRadians(SpaceShip.rotationAngle));//need to understand that 10 is Rads for an angle
                graphicSettings.draw(theShip);
        }//end paint
```

```
}//end Game_Board class
```

```
import java.awt.Polygon;
import java.util.ArrayList;
import java.awt.Rectangle; //need for asteroid collisions sets boundaries


//----------------------------------------------------------------
class Rock extends Polygon{

         private static final long serialVersionUID = 1L;
         int uLeftXPos,uLeftYPos;
         int xDirection = 1;
         int yDirection = 1;
         int width = Game_Board.boardWidth;
         int height = Game_Board.boardHeight;
         int[] polyXArray, polyYArray;

         public static int[] sPolyXArray = {10,17,26,34,27,36,26,14,8,1,5,1,10};
         public static int[] sPolyYArray = {0,5,1,8,13,20,31,28,31,22,16,7,0};

         public static ArrayList<Rock> rocks = new ArrayList<Rock>(); //stay in sync with game board this is Rock.rocks in game board

         //constructor
         public Rock(int[] polyXArray, int[] polyYArray,int pointsInPoly, int randomStartXPos, int randomStartYPos) {

                  //calling Polygon superclass creates a polygon asteroid
                  super(polyXArray,polyYArray,pointsInPoly);

                  //random positions for asteroids
                  this.xDirection = (int) (Math.random()*4+1);
                  this.yDirection = (int) (Math.random()*4+1);
                  this.uLeftXPos = randomStartXPos;
                  this.uLeftYPos = randomStartYPos;
         }//end constructor

         //--------------Collision Detection------------------------------------------------------------------------
         public Rectangle getBounds() {
                  //int rockWidth = super.xpoints[0]+26;
                  //int rockHeight = super.ypoints[0] + 31;
                  return  new Rectangle(super.xpoints[0],super.ypoints[0], 31, 36);
         }//end getBounds




         //----------------------------------------------------------------------------------------------------

         public void move() {


                  //--- get boundary of each rock to detect a collision
                  Rectangle rockToCheck = this.getBounds();
                  for(Rock rock: rocks) {
                           Rectangle otherRock = rock.getBounds();
                           if(rock != this && otherRock.intersects(rockToCheck)) { //if there is collision
                                    int tempXDirection = this.xDirection;
                                    int tempYDirection = this.yDirection;
                                    this.xDirection = rock.xDirection;
                                    this.yDirection = rock.yDirection;
                                    rock.xDirection = tempXDirection;
                                    rock.yDirection = tempYDirection;
                           }//end if
```

```
		}//end for

		//-- every time I call this I reset uLeftXPos and uLeftYPos and it should stay fixed
		int uLeftXPos = super.xpoints[0];
		int uLeftYPos = super.ypoints[0];
		if(uLeftXPos >= (Game_Board.boardWidth-10) || uLeftXPos <= 10 )
		{xDirection = - xDirection;}
		if(uLeftYPos >= (Game_Board.boardHeight-10) || uLeftYPos <= 10)
		{yDirection = -yDirection;}

		//new direction points must move polygon points must move
		for(int i = 0; i < super.xpoints.length; i++) {

			super.xpoints[i] += xDirection;
			super.ypoints[i] += yDirection;
		}//end for
	}//end move


	public static int[] getpolyXArray(int randomStartXPos) {
		int[] tempPolyXArray = (int[])sPolyXArray.clone();
		for(int i = 0; i < tempPolyXArray.length; i++) {
			tempPolyXArray[i] += randomStartXPos; //beginning a new asteroid at a random position
		}//end for
		return tempPolyXArray;
	}//end getPolyX

	public static int[] getpolyYArray(int randomStartYPos) {
		int[] tempPolyYArray = (int[])sPolyYArray.clone();
		for(int i = 0; i < tempPolyYArray.length; i++) {
			tempPolyYArray[i] += randomStartYPos; //beginning a new asteroid at a random position
		}//end for
		return tempPolyYArray;
	}//end getPolyY

}//end class Rock
```

UML DIAGRAMS
Asteroids referred to as Rock objects
Rock what are all the things this rock needs?
Location
uLeftXPos: integer
uLeftYPos: integer
ability to change direction when it hits an edge
xDirection: integer
yDirection: integer

boardWidth: integer
boardHeight: integer

Each asteroid is a polygon which is a series of x and y points connected by a line
polyXArray[]: integer
polyYArray[]: integer

starting Position of each polygon sp
sppolyXArray[]: integer
sppolyYArray[]: integer

Rock(polyXArray[]: integer, polyYArray[]: integer, pointsInPoly : integer, randomStartXPos: integer, randomStartYPos: integer):void
the coordinate shapes and the position shapes are all stored in the asteroid object

move(): void

getPolyXArray(randomStartXPos:integer):integer[]

getPolyYArray(randomStartYPos:integer):integer[]
==
Game_Board

boardWidth: integer
boardHeight: integer

main(args: String[]): void

RepaintTheBoard

theBoard: game_board

RepaintTheBoard(game_board): void

run(): void thread

GameDrawingPanel

rocks: ArrayList<Rock>

polyXArray: int[]
polyYArray: int[]

width: integer
height: integer

GameDrawingPanel() : void //constructor

paint(g : Graphics): void

==================================
UML PhotonTorpedo

gBWidth: int
gBHeight: int
centerX: double
centerY: double
polyXArray: int[]
polyYArray: int[]
torpedoWidth: int
torpedoHeight: int
onScreen: boolean
movingAngle: double
xVelocity: double
yVelocity: double