**Course Title:** Computer Science Level 6

**Module:** Object Orientated Programming

**Course Code:** 6N2108

**Tutor Name:** Keith Quille

**Assignment Title:** Skill Demo 2

**Student:** James Fleming

**Student PPSN:** 7706728A

**Date:** /04/2017

**Contents**

1. **Code**

* **Sprite**
* **Enemy**
* **Player**
* **Vector**
* **Main Program**

1. **Testing**

* **All properties of Enemy and Player print to screen**
* **enemy1 and player1 x,y changes after Jump, Retreat and Attack**
* **playerVect2 = playerVect1 \* a scaler of 7**
* **playerVect3 = playerVect2 + enemyVect1**

1. **User Manual**

* **Class Diagram**
* **XML**

**CODE**

**Sprite**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2DGameLibrary

{

public class Sprite

{

//Sprites variables

private string name;

private double xLoc;

private double yLoc;

private bool visible;

private bool alive;

//Constructors

/// <summary>

/// Default Constructor

/// </summary>

public Sprite()

{

this.name = "Sprite";

this.xLoc = 0.0;

this.yLoc = 0.0;

this.visible = true;

this.alive = true;

}

/// <summary>

/// Overloaded Constructor

/// </summary>

/// <param name="sName">Sprite Name : String</param>

/// <param name="locX">Sprite's X location : Double</param>

/// <param name="locY">Sprite's Y location : Double</param>

/// <param name="seeable">Is Sprite Visible? : Bool</param>

/// <param name="active">Is Sprite Alive? : Bool</param>

public Sprite(string sName, double locX, double locY, bool seeable, bool active)

{

this.name = sName;

this.xLoc = locX;

this.yLoc = locY;

this.visible = seeable;

this.alive = active;

}

//gets and sets

//gets

/// <summary>

/// gets the name of the sprite

/// </summary>

/// <returns>Sprite name : String</returns>

public string getName()

{

return this.name;

}

/// <summary>

/// gets the X location of the sprite

/// </summary>

/// <returns>X Location : Double</returns>

public double getXLoc()

{

return this.xLoc;

}

/// <summary>

/// gets the Y location of the sprite

/// </summary>

/// <returns>Y location : Double</returns>

public double getYLoc()

{

return this.yLoc;

}

/// <summary>

/// Bool to see if Sprite is visible or not

/// </summary>

/// <returns>Is Sprite visible? : Bool</returns>

public bool isVisible()

{

return this.visible;

}

/// <summary>

/// Bool to see if Sprite is alive or not

/// </summary>

/// <returns>Is Sprite alive? : Bool</returns>

public bool isAlive()

{

return this.alive;

}

//sets

/// <summary>

/// Set the nmae of the Sprite

/// </summary>

/// <param name="nameIn">Sprite name : String</param>

public void setName(string nameIn)

{

this.name = nameIn;

}

/// <summary>

/// sets the X location of the Sprite

/// </summary>

/// <param name="xLocIn">X location of the Sprite : Double</param>

public void setXLoc(double xLocIn)

{

this.xLoc = xLocIn;

}

/// <summary>

/// sets the Y location of the Sprite

/// </summary>

/// <param name="yLocIn">Y location of the Sprite : Double</param>

public void setYLoc (double yLocIn)

{

this.yLoc = yLocIn;

}

/// <summary>

/// sets if the sprite is visible or not

/// </summary>

/// <param name="visibleIn">is the Sprite is visible : bool</param>

public void setVisible(bool visibleIn)

{

this.visible = visibleIn;

}

/// <summary>

/// sets if the sprite is alive or not

/// </summary>

/// <param name="aliveIn">is the Sprite is alive : Bool</param>

public void setAlive(bool aliveIn)

{

this.alive = aliveIn;

}

//Behaviours

/// <summary>

/// gets the sprite to say "I am a sprite"

/// </summary>

public virtual void speak()

{

Console.WriteLine("I am a sprite");

}

/// <summary>

/// Jump increases Y location by 10

/// </summary>

public void jump()

{

this.yLoc = this.yLoc + 10;

}

/// <summary>

/// Prints the current location of the sprite

/// </summary>

public void Location()

{

Console.WriteLine("My current location is X " + getXLoc() + " , Y " + getYLoc());

}

}

}

**CODE**

**Enemy**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2DGameLibrary

{

public class Enemy : Sprite

{

//Enemy Variables

private string teamName;

private int playersDestroyed;

//constructor

/// <summary>

/// default constructor for an enemy

/// </summary>

public Enemy()

{

this.teamName = "Team 2";

this.playersDestroyed = 0;

}

//gets and sets

//gets

/// <summary>

/// gets the team Name of the Enemy

/// </summary>

/// <returns>Team Name of the Enemy : String</returns>

public string getTeamName()

{

return this.teamName;

}

/// <summary>

/// gets the Amount of Players the Enemies has Destroyed

/// </summary>

/// <returns>the Amount Players the Enemy has Destroyed : Int</returns>

public int getPlayersDestroyed()

{

return this.playersDestroyed;

}

//sets

/// <summary>

/// sets the Team Name of the Enemy

/// </summary>

/// <param name="teamNamein">Team Name of the Enemy : String</param>

public void setTeamName(string teamNamein)

{

this.teamName = teamNamein;

}

/// <summary>

/// Sets the Amount of Players the Enemy has Destroyed

/// </summary>

/// <param name="playersDestroyedIn">the Amount of Players the Enemy has Destroyed : Int</param>

public void setPlayersDestroyed(int playersDestroyedIn)

{

this.playersDestroyed = playersDestroyedIn;

}

//behaviours

/// <summary>

/// Overridden operator so that Enemy speaks differently to it's base class Sprite

/// </summary>

public override void speak()

{

Console.WriteLine("I am an Enemy");

}

/// <summary>

/// method to Allow the Enemy Attack by adding 10 to its X location

/// </summary>

public void attack()

{

this.setXLoc(this.getXLoc() + 10);

}

}

}

**CODE**

**Player**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2DGameLibrary

{

public class Player : Sprite

{

//Player variables

private string teamName;

private int enemiesDestroyed;

//constructor

/// <summary>

/// The default constructor for the Player

/// </summary>

public Player()

{

this.teamName = "Team 1";

this.enemiesDestroyed = 0;

}

//gets and sets

//gets

/// <summary>

/// gets the Team Name of the Player

/// </summary>

/// <returns>Team Name : String</returns>

public string getTeamName()

{

return this.teamName;

}

/// <summary>

/// find out how many enemies the Player has destroyed

/// </summary>

/// <returns>Enemies Player has Destroyed : Int</returns>

public int getEnemiesDestroyed()

{

return this.enemiesDestroyed;

}

//sets

/// <summary>

/// Sets the Team Name of Player

/// </summary>

/// <param name="teamNameIn">Team Name : String</param>

public void setTeamName(string teamNameIn)

{

this.teamName = teamNameIn;

}

/// <summary>

/// sets the Amount of Enemies the Player has destroyed

/// </summary>

/// <param name="enemiesDestroyedIn">Enemies Player has destroyed : Int</param>

public void setEnemiesDestroyed(int enemiesDestroyedIn)

{

this.enemiesDestroyed = enemiesDestroyedIn;

}

//Behaviours

/// <summary>

/// Overridden method to allow the Players Speak be different from that of the Sprite base class

/// </summary>

public override void speak()

{

Console.WriteLine("I am the Player");

}

/// <summary>

/// method to allow the player Retreat Xlocation - 10 when used

/// </summary>

public void retreat()

{

this.setXLoc(this.getXLoc() - 10);

}

}

}

**CODE**

**Vector**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_2DGameLibrary

{

public class Vector

{

//Vector Variables

private double i;

private double j;

//Constuctors

/// <summary>

/// Default Constructor for the Vector

/// </summary>

public Vector()

{

this.i = 0.0;

this.j = 0.0;

}

/// <summary>

/// Overloaded Constructor for the Vector

/// </summary>

/// <param name="ii"> i of the Vector : Double </param>

/// <param name="jj"> j of the Vector : Double </param>

public Vector(double ii, double jj)

{

this.i = ii;

this.j = jj;

}

//gets and sets

//gets

/// <summary>

/// gets the i component of the Vector

/// </summary>

/// <returns>i component : Double</returns>

public double getI()

{

return this.i;

}

/// <summary>

/// gets the j component of the Vector

/// </summary>

/// <returns>j component : Double</returns>

public double getJ()

{

return this.j;

}

//sets

/// <summary>

/// sets the i component of the Vector

/// </summary>

/// <param name="iIN">i component : Double</param>

public void setI(double iIN)

{

this.i = iIN;

}

/// <summary>

/// sets the j component of the Vector

/// </summary>

/// <param name="jIN">j component : Double</param>

public void setJ(double jIN)

{

this.j = jIN;

}

//behaviours

/// <summary>

/// Prints the current Vector values in i,j form

/// </summary>

public void printVector()

{

Console.WriteLine(this.i + "i , " + this.j+"j");

}

/// <summary>

/// Overloaded Operator for adding 2 vectors

/// </summary>

/// <param name="a">First Vector : Vector</param>

/// <param name="b">Second Vector : Vector</param>

/// <returns>Result of adding the two Vectors</returns>

public static Vector operator +(Vector a, Vector b)

{

//creates c vector that is the result of Vector a + Vector b

Vector c = new Vector();

//sets Vector c's i,j to ai + bi , aj + bj

c.setI(a.getI() + b.getI());

c.setJ(a.getJ() + b.getJ());

return c;

}

/// <summary>

/// Overloaded Operator for multiplying a Vector by a Scaler

/// </summary>

/// <param name="a">Vector : Vector</param>

/// <param name="scaler">Scaler : Double</param>

/// <returns>Vector Multiplied by Scaler : Vector</returns>

public static Vector operator \*(Vector a, double scaler)

{

//creates a vector that will be the result of vector a by the Scaler

Vector ans = new Vector();

//Sets Vector ans's i,j with the result of mulitplying Vector a's i,j by the scaler

ans.setI(a.getI() \* scaler);

ans.setJ(a.getJ() \* scaler);

return ans;

}

/// <summary>

/// Overloaded Operator for multiplying a Scaler by a Vector

/// </summary>

/// <param name="scaler">Scaler : Double</param>

/// <param name="a">Vector : Vector</param>

/// <returns>Scaler multiplied by Vector : Vector</returns>

public static Vector operator \*(double scaler, Vector a)

{

//creates a vector that will be the result of the Scaler by Vector a

Vector ans = new Vector();

//Sets Vector ans's i,j with the result of mulitplying the scaler by the Vector a's i,j

ans.setI(a.getI() \* scaler);

ans.setJ(a.getJ() \* scaler);

return ans;

}

}

}

**CODE**

**Main Program**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using \_2DGameLibrary;

namespace OOP\_Skills\_Demo\_2

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* PART 1 \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//Create an Enemy and set all its properties

Enemy enemy1 = new Enemy();

enemy1.setName("The Enemy");

enemy1.setXLoc(100.0);

enemy1.setYLoc(100.0);

enemy1.setVisible(true);

enemy1.setAlive(true);

enemy1.setTeamName("The Enemy Team");

enemy1.setPlayersDestroyed(0);

//Creates a Player and set all its properties

Player player1 = new Player();

player1.setName("The Player");

player1.setXLoc(0.0);

player1.setYLoc(0.0);

player1.setVisible(true);

player1.setAlive(true);

player1.setTeamName("The Player Team");

player1.setEnemiesDestroyed(0);

//Displaying the Enemy Information

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*" + enemy1.getName() + "\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Enemy Name : " +enemy1.getName());

Console.WriteLine("Enemy Location : (X" +enemy1.getXLoc()+" , Y"+enemy1.getYLoc()+")");

Console.WriteLine("Enemy Visible : " + enemy1.isVisible());

Console.WriteLine("Enemy Alive : " + enemy1.isAlive());

Console.WriteLine("Team Name : " + enemy1.getTeamName());

Console.WriteLine("Players Destroyed : " + enemy1.getPlayersDestroyed());

//displaying the Player Information

Console.WriteLine("");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*" + player1.getName() + "\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("Player Name : " + player1.getName());

Console.WriteLine("Player Location : (X" + player1.getXLoc() + " , Y" + player1.getYLoc() + ")");

Console.WriteLine("Player Visible : " + player1.isVisible());

Console.WriteLine("Player Alive : " + player1.isAlive());

Console.WriteLine("Team Name : " + player1.getTeamName());

Console.WriteLine("Enemies Destroyed : " + player1.getEnemiesDestroyed());

Console.WriteLine("");

Console.WriteLine("Press any Key to Continue");

Console.ReadKey();

//Clears the display

Console.Clear();

//Get both Enemy and Player to Jump

enemy1.jump();

player1.jump();

//Displaying the Enemy's and Player's new position

Console.Write(enemy1.getName() +" : ");

enemy1.Location();

Console.WriteLine("");

Console.Write(player1.getName() + " : ");

player1.Location();

Console.WriteLine("");

Console.WriteLine("Press any Key to Continue");

Console.ReadKey();

//Enemy Attack and Print new Location

enemy1.attack();

Console.Write(enemy1.getName() + " : ");

enemy1.Location();

Console.WriteLine("");

//Player Retreat and Print new Location

player1.retreat();

Console.Write(player1.getName() + " : ");

player1.Location();

Console.WriteLine("");

Console.WriteLine("Press any Key to Continue");

Console.ReadKey();

//clear screen for part 2

Console.Clear();

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* PART 2 \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//creating the player vector 1

Vector PlayerVect1 = new Vector(player1.getXLoc(), player1.getYLoc());

//creating player vector 2(player vector 1 multiplied by 7)

Vector playerVect2 = new Vector();

playerVect2 = PlayerVect1 \* 7;

//printing resulting Vector to the Screen

Console.Write("playerVect2 = ");

playerVect2.printVector();

//creating the enemy vector 1

Vector EnemyVect1 = new Vector(enemy1.getXLoc(), enemy1.getYLoc());

//creating the player vector 3(player vector 2 + enemy vector 1)

Vector playerVect3 = new Vector();

playerVect3 = playerVect2 + EnemyVect1;

//printing resulting Vector to the Screen

Console.Write("playerVect3 = ");

playerVect3.printVector();

Console.ReadKey();

}

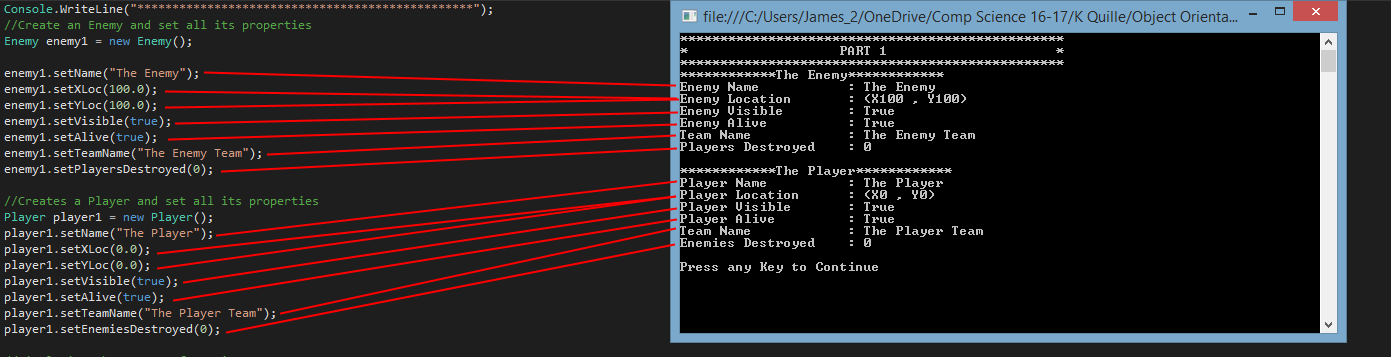
}

}

**Testing**

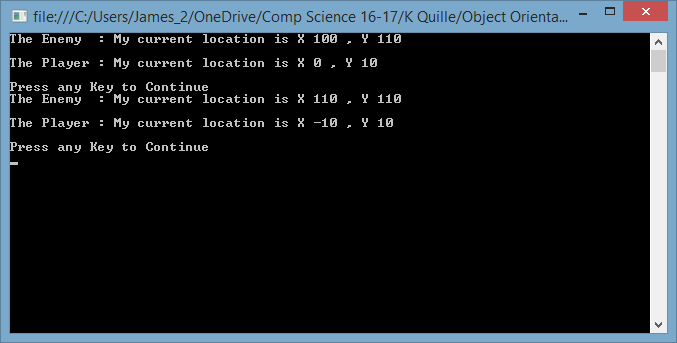
**All properties of Enemy and Player print to screen**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| To see if all properties of enemy1 and player1 print successfully to the screen | The properties of enemy1 and player1 print successfully to the screen | All the properties printed to the screen correctly | Success |

****

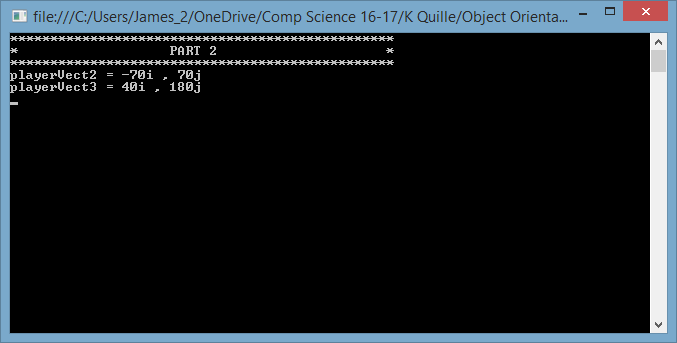
**enemy1 and player1 x,y changes after Jump, Retreat and Attack**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| To see if the Jump, Retreat and Attack methods work for enemy1 and player1 | After the Jump, Attack and Retreat methods are used enemy1's x,y should equal 110,100 and player1's x,y should equal -10,10 | enemy1's x,y equals 110,100 and player1's x,y equals -10,10 | success |

****

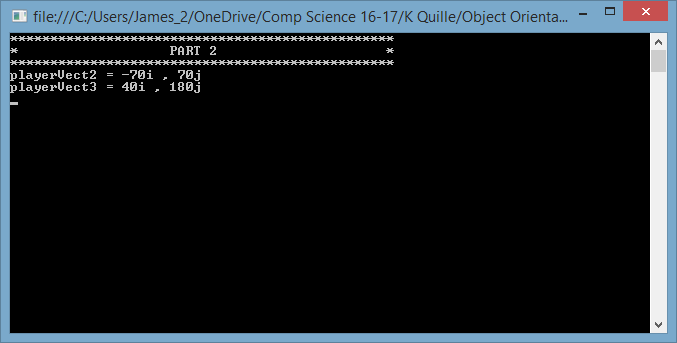
**playerVect2 = playerVect1 \* a scaler of 7**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| To see if the \* overloaded operator works by multiplying playerVect1 by a Scaler of 7 | playerVect2's i,j to equal -70,70 | playerVect2's i,j is  -70,70 | success |

****

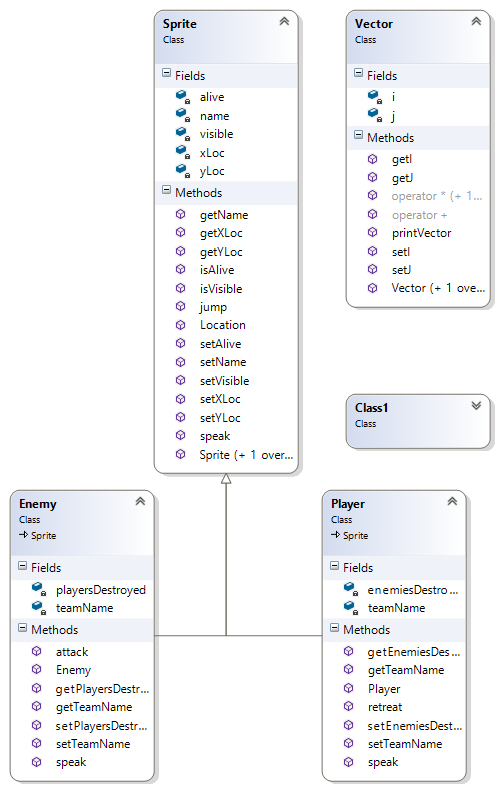
**playerVect3 = playerVect2 + enemyVect1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| To see if the + overloaded operator works by adding playerVect2 (-70,70) and enemyVect1(110,110) to get playerVect3 | playerVect3 to equal 40,180 | playerVect3 equals 40,180 | success |

****

**USER MANUAL**

**Class Diagram**

****

**USER MANUAL**

**XML**

**<?xml version="1.0"?>**

**<doc>**

**<assembly>**

**<name>2DGameLibrary</name>**

**</assembly>**

**<members>**

**<member name="T:\_2DGameLibrary.Class1">**

**<summary>**

**empty class**

**</summary>**

**</member>**

**<member name="T:\_2DGameLibrary.Enemy">**

**<summary>**

**Class for an Enemy that inherits from the Sprite class**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.#ctor">**

**<summary>**

**default constructor for an enemy**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.getTeamName">**

**<summary>**

**gets the team Name of the Enemy**

**</summary>**

**<returns>Team Name of the Enemy : String</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.getPlayersDestroyed">**

**<summary>**

**gets the Amount of Players the Enemies has Destroyed**

**</summary>**

**<returns>the Amount Players the Enemy has Destroyed : Int</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.setTeamName(System.String)">**

**<summary>**

**sets the Team Name of the Enemy**

**</summary>**

**<param name="teamNamein">Team Name of the Enemy : String</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.setPlayersDestroyed(System.Int32)">**

**<summary>**

**Sets the Amount of Players the Enemy has Destroyed**

**</summary>**

**<param name="playersDestroyedIn">the Amount of Players the Enemy has Destroyed : Int</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.speak">**

**<summary>**

**Overridden operator so that Enemy speaks differently to it's base class Sprite**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Enemy.attack">**

**<summary>**

**method to Allow the Enemy Attack by adding 10 to its X location**

**</summary>**

**</member>**

**<member name="T:\_2DGameLibrary.Player">**

**<summary>**

**Class for a Player that inherits from the Sprite Class**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.#ctor">**

**<summary>**

**The default constructor for the Player**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.getTeamName">**

**<summary>**

**gets the Team Name of the Player**

**</summary>**

**<returns>Team Name : String</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.getEnemiesDestroyed">**

**<summary>**

**find out how many enemies the Player has destroyed**

**</summary>**

**<returns>Enemies Player has Destroyed : Int</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.setTeamName(System.String)">**

**<summary>**

**Sets the Team Name of Player**

**</summary>**

**<param name="teamNameIn">Team Name : String</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.setEnemiesDestroyed(System.Int32)">**

**<summary>**

**sets the Amount of Enemies the Player has destroyed**

**</summary>**

**<param name="enemiesDestroyedIn">Enemies Player has destroyed : Int</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.speak">**

**<summary>**

**Overridden method to allow the Players Speak be different from that of the Sprite base class**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Player.retreat">**

**<summary>**

**method to allow the player Retreat Xlocation - 10 when used**

**</summary>**

**</member>**

**<member name="T:\_2DGameLibrary.Sprite">**

**<summary>**

**Class for a Sprite**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.#ctor">**

**<summary>**

**Default Constructor**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.#ctor(System.String,System.Double,System.Double,System.Boolean,System.Boolean)">**

**<summary>**

**Overloaded Constructor**

**</summary>**

**<param name="sName">Sprite Name : String</param>**

**<param name="locX">Sprite's X location : Double</param>**

**<param name="locY">Sprite's Y location : Double</param>**

**<param name="seeable">Is Sprite Visible? : Bool</param>**

**<param name="active">Is Sprite Alive? : Bool</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.getName">**

**<summary>**

**gets the name of the sprite**

**</summary>**

**<returns>Sprite name : String</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.getXLoc">**

**<summary>**

**gets the X location of the sprite**

**</summary>**

**<returns>X Location : Double</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.getYLoc">**

**<summary>**

**gets the Y location of the sprite**

**</summary>**

**<returns>Y location : Double</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.isVisible">**

**<summary>**

**Bool to see if Sprite is visible or not**

**</summary>**

**<returns>Is Sprite visible? : Bool</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.isAlive">**

**<summary>**

**Bool to see if Sprite is alive or not**

**</summary>**

**<returns>Is Sprite alive? : Bool</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.setName(System.String)">**

**<summary>**

**Set the nmae of the Sprite**

**</summary>**

**<param name="nameIn">Sprite name : String</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.setXLoc(System.Double)">**

**<summary>**

**sets the X location of the Sprite**

**</summary>**

**<param name="xLocIn">X location of the Sprite : Double</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.setYLoc(System.Double)">**

**<summary>**

**sets the Y location of the Sprite**

**</summary>**

**<param name="yLocIn">Y location of the Sprite : Double</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.setVisible(System.Boolean)">**

**<summary>**

**sets if the sprite is visible or not**

**</summary>**

**<param name="visibleIn">is the Sprite is visible : bool</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.setAlive(System.Boolean)">**

**<summary>**

**sets if the sprite is alive or not**

**</summary>**

**<param name="aliveIn">is the Sprite is alive : Bool</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.speak">**

**<summary>**

**gets the sprite to say "I am a sprite"**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.jump">**

**<summary>**

**Jump increases Y location by 10**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Sprite.Location">**

**<summary>**

**Prints the current location of the sprite**

**</summary>**

**</member>**

**<member name="T:\_2DGameLibrary.Vector">**

**<summary>**

**Class for Vector Maths**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.#ctor">**

**<summary>**

**Default Constructor for the Vector**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.#ctor(System.Double,System.Double)">**

**<summary>**

**Overloaded Constructor for the Vector**

**</summary>**

**<param name="ii"> i of the Vector : Double </param>**

**<param name="jj"> j of the Vector : Double </param>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.getI">**

**<summary>**

**gets the i component of the Vector**

**</summary>**

**<returns>i component : Double</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.getJ">**

**<summary>**

**gets the j component of the Vector**

**</summary>**

**<returns>j component : Double</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.setI(System.Double)">**

**<summary>**

**sets the i component of the Vector**

**</summary>**

**<param name="iIN">i component : Double</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.setJ(System.Double)">**

**<summary>**

**sets the j component of the Vector**

**</summary>**

**<param name="jIN">j component : Double</param>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.printVector">**

**<summary>**

**Prints the current Vector values in i,j form**

**</summary>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.op\_Addition(\_2DGameLibrary.Vector,\_2DGameLibrary.Vector)">**

**<summary>**

**Overloaded Operator for adding 2 vectors**

**</summary>**

**<param name="a">First Vector : Vector</param>**

**<param name="b">Second Vector : Vector</param>**

**<returns>Result of adding the two Vectors</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.op\_Multiply(\_2DGameLibrary.Vector,System.Double)">**

**<summary>**

**Overloaded Operator for multiplying a Vector by a Scaler**

**</summary>**

**<param name="a">Vector : Vector</param>**

**<param name="scaler">Scaler : Double</param>**

**<returns>Vector Multiplied by Scaler : Vector</returns>**

**</member>**

**<member name="M:\_2DGameLibrary.Vector.op\_Multiply(System.Double,\_2DGameLibrary.Vector)">**

**<summary>**

**Overloaded Operator for multiplying a Scaler by a Vector**

**</summary>**

**<param name="scaler">Scaler : Double</param>**

**<param name="a">Vector : Vector</param>**

**<returns>Scaler multiplied by Vector : Vector</returns>**

**</member>**

**</members>**

**</doc>**