**Programming Paradigms – Encapsulation**

**LO1: Explain what a constructor is and how it is useful**

**LO2: Examine existing classes and identify different features**

**LO3: Create classes using constructors**

**Starter:**

Continue with your text based game from last lesson.

**EVIDENCE:** Take screenshots of your results here

**Constuctors**

A constructor in C# is a member of a class. It is a method in the class which gets executed when a class object is created. Usually we put the initialization code in the constructor. The name of the constructor is always the same name as the class. A C# constructor can be public or private.

Writing a constructor in the class is pretty simple, have a look at the following sample:

1. **public** **class** mySampleClass
2. {
3. **public** mySampleClass()
4. {
5. // This is the constructor method.
6. }
7. // rest of the class members goes here.
8. }

**Attributes and parameters**

Constructors can be used to define the attributes of a class. To do this you should first define the class’s variables at the top of the program. These variables should be private.

Separate variables, which can be used as pointers to the private variables should be set up within the constructor.

This can be seen below.

private int height, width;

private string name;

public Shape(int sHeight, int sWidth, string sName)

{

height = sHeight;

width = sWidth;

name = sName;

}

**Predict 1 –Constructors**

namespace Constructors

{

public class Enemy

{

private int health, damage, shield, turns;

private string name, battleCry;

public Enemy(int eHealth, int eDamage, int eShield, int eTurns, string eName, string eBattleCry)

{

health = eHealth;

damage = eDamage;

shield = eShield;

turns = eTurns;

name = eName;

battleCry = eBattleCry;

}

public int Health {

get

{

return health;

}

set

{

health = value;

}

}

public int Damage

{

get

{

return damage;

}

set

{

damage = value;

}

}

public int Shield

{

get

{

return shield;

}

set

{

shield = value;

}

}

public int Turns

{

get

{

return turns;

}

set

{

turns = value;

}

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string BattleCry

{

get

{

return battleCry;

}

set

{

battleCry = value;

}

}

}

}

namespace Constructors

{

class Program

{

static void Main(string[] args)

{

Enemy zombie = new Enemy(100, 20, 10, 7, "Walker", "Bleaaaurgh...");

int myHealth = 70;

int myShield = 40;

int myDamage = 25;

int myTurns = 8;

string myVoice = "Aaaargh...";

Console.WriteLine(zombie.Turns);

while(zombie.Turns > 0 && myTurns > 0)

{

Console.WriteLine("A zombie approaches, what do you do?");

string option = Console.ReadLine().ToLower();

if(option == "attack")

{

zombie.Health -= myDamage;

Console.WriteLine("The zombies health is now {0}", zombie.Health);

zombie.Turns -= 1;

}

else if(option == "dodge")

{

Console.WriteLine("You successfully evaded the attack");

zombie.Turns -= 1;

}

else

{

myHealth -= zombie.Damage;

Console.WriteLine("The zombie struck you");

}

if(myHealth < 0)

{

char[] final = myVoice.ToCharArray();

foreach (char x in final)

{

Console.Write(x);

Thread.Sleep(50);

}

Console.WriteLine("\nYou are dead");

myTurns = 0;

}

else if(zombie.Health < 0)

{

char[] final = zombie.BattleCry.ToCharArray();

foreach (char x in final)

{

Console.Write(x);

Thread.Sleep(50);

}

Console.WriteLine("\nThe {0} is dead!", zombie.Name);

zombie.Turns = 0;

}

}

}

}

}

**Q: What is the output of the program above, using the following inputs?**

1. **Attack x 6 – You kill the zombie**
2. **Dodge x 6 – You evade all the zombie’s attacks**
3. **Run x 6 – The zombie kills you**

**Modify – Encapsulation**

**Q: Make the following modifications to the above program.**

1. **Add 1 extra scenario to the game.**
2. **Add a Friendly class which will take the place of the myHealth, myDamage, myTurns, and myVoice variables**
3. **Utilise the shield variables so that they are taken off before the health**

**Make – Encapsulated classes**

**Q: Add constructors to your text based game from last week.**