**OOP Lab #5**

**Properties and Encapsulation in C#**

**Property:**

A property is much like a combination of a variable and a method. It can’t take any parameters but you are able to process the value before it’s assigned to our returned.

Properties are like data fields but have logic behind them. Properties are also used for Encapsulation. Define like a field with Get & Set accessor code added.

**Types of Properties:**

* Read /Write Property
* Read only Property
* Write Property
* Auto Implemented Property

**Read/Write Property.**

namespace Getter\_and\_Setter

{

class Program

{

class Student

{

private string StdName;

private string FName; //Read / Write Property

private int CMSID;

public string Name

{

set

{

this.StdName = value;

}

get

{

return this.StdName;

}

}

public string Fname

{

set

{

this.FName = value;

}

get

{

return this.FName;

}

}

public int Cmsid

{

set

{

this.CMSID = (value);

}

get

{

return this.CMSID;

}

}

}

static void Main(string[] args)

{

Student s = new Student();

s.Name="Zohaib Amjad";

Console.WriteLine(s.Name);

s.Fname = "Amjad Maseeh";

Console.WriteLine(s.Fname);

s.Cmsid = int.Parse("51928");

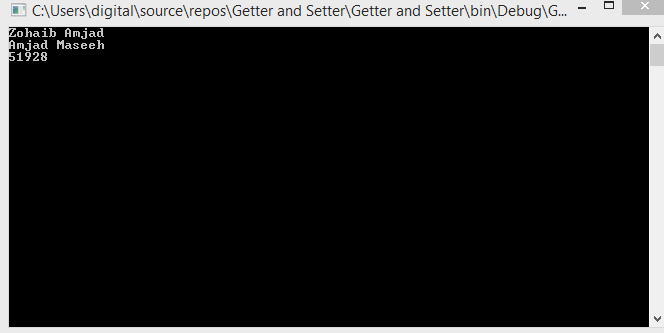
Console.WriteLine (s.Cmsid);

Console.ReadLine();

}

}

}



**Read Only Property:**

namespace Getter\_and\_Setter

{

class Program

{ //Read Only Property

class Student

{

string StdName = "Zohaib";

string FName;

int CMSID;

public string Stdname

{

get

{

return this.Stdname;

}

}

} // in this property we can not change data in other class .

static void Main(string[] args)

{

Student s = new Student();

s.StdName = "Amjad";

// when we try to assess this in other class and change it

//it gives error

}

}

}

**Write Property:**

namespace Getter\_and\_Setter

{

class Program

{

class Student

{

string StdName = "Zohaib";

string FName; //Write Property

int CMSID;

public string Stdname

{

set

{

this.Stdname =value;

}

}

} // in this property we can not get data only set data .

static void Main(string[] args)

{

Student s = new Student();

s.Stdname = "Zohaib";

Console.ReadLine();

}

**Auto Implemented Property:**

namespace Getter\_and\_Setter

{

class Program

{

class Student

{ // Auto ImplementedProperty

public string FirstName { get; set; }

public string LastName { get; set; }

}

static void Main(string[] args)

{

Student s = new Student();

s.FirstName = "Zohaib";

s.LastName = "Amjad";

Console.WriteLine(s.FirstName +" " + s.LastName);

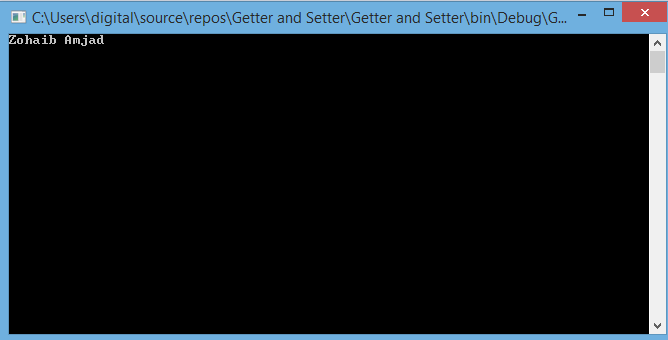
Console.ReadLine();

}

}

}

**Output**

****

**Encapsulation:**

* Encapsulation is one of forth fundamental OOP concepts. The other three are inheritance, polymorphism and abstraction.
* Encapsulation is a mechanism of wrapping the data acting on the data methods or property together as a single unit.
* In Encapsulation the variables of a class will be hidden from other classes and can be access only through the methods or properties of their current class and it is also known as data hiding.

**Code Example**

namespace Getter\_and\_Setter

{

class Program

{

class person

{// Encapsulation

private string name;

private int age;

public void setname(string name)

{

if (true)

{

}

this.name = name;

}

public void getname()

{

Console.WriteLine("your Name is:" + this.name) ;

}

public void setage(int Age)

{

this.age = Age;

}

public void getage()

{

Console.WriteLine("your Age is:" + this.age);

}

}

static void Main(string[] args)

{

person p1 = new person();

p1.setname("Zohaib Amjad");

p1.getname();

p1.setage(22);

p1.getage();

Console.ReadLine();

}

}

}

**Output**

