**Salman Ahmed Khan**

**19K-1043**

**DDR Assignment 2**

**Reflection in .Net Frame Work**

**Part (A)**

The reflection is defined in System.Reflection namespace that allows a programmer to obtain information and load the assemblies, types, classes, interfaces, methods, properties, constructor etc. Reflection is used to create a runtime instance of objects, invoke them and access them.

For example: We create an interface of IEat that allows a class to inherit its eat method. Let’s say we have a class Cat that is an animal and it eats as well. At runtime, we might want to know its interface type so we may perform certain actions on its instantiated object. Here, reflection assist us to know the type of its interface and through this we can apply our logic.

using System;

using System.Reflection;

public class HelloWorld

{

public interface IEat{

void eat();

}

class Cat: IEat {

public string \_food;

Cat(string f) {

this.\_food = f;

}

public void eat() {

Console.WriteLine("Cat eating ", this.\_food);

}

}

public static void Main(string[] args)

{

Console.WriteLine ("Hello Mono World");

Cat c = new Cat("Meat");

c.eat();

Type t = typeof(Cat);

Type[] interfaces = t.GetInterfaces();

foreach (Type i in interfaces)

if (i.Name.ToString().ToLower().Contains(“cat”))

Console.WriteLine(“->{0}”, i.Name);

}

}

**Part (B)**

Reflection helps you to get information regarding assemblies, classes etc trough which you can instantiate object at runtime. However, in factory design pattern, switch cases or if else statement is used to get object at runtime. With the help of reflection, we don’t have to uses cases and statement rather we can use the assemble.GetExecutingAssembly and GetType method to get the class assembly and type and could easily return the object.

public Shape FactoryInstance(string shapeName){

var assembly = Assembly.GetExecutingAssembly();

var type = assembly.GetType(shapeName).FullName;

return (Shape) Activator.CreateInstanceFrom(assembly.Location, type).Unwrap();

}