Factory Pattern

- Aplicação

A aplicação escolhida foi um aplicativo bancário, para representar o design de software escolhido foi selecionado um exemplo dentro da aplicação. Os tipos de contas bancárias (poupança, conta corrente). Todas as contas derivam de uma conta abstrata chamada **IAccount**, que define saques, depósitos, taxas de juros que precisam ser aplicadas as contas concretas (poupança e conta corrente).
Se um cliente quiser saber a taxa de juros de sua conta poupança, utilizando o método escolhido, é criado uma instância da conta poupança, e então o cliente recebe o interesse invocando o método **interest()**.

A vantagem do método é que uma nova conta pode ser adicionada sem mudar nenhuma linha de código na instância do **client**.

- Beneficios

1. Factory Pattern é o padrão mais utilizado atualmente
2. Elimina a necessidade de vincular uma classe específica do aplicativo
3. O código apenas trata da interface
4. Fornece maneiras de criar várias instâncias de classes
5. Permite derivar uma subclasse para criar diferentes controles para exibir dados
6. Conecta hierarquia de classes com acoplamento mínimo
7. Implementação do produto pode variar mas o cliente continua o mesmo

- Implementação

1. Criando uma interface - **IAccount**

```
namespace FactoryPattern
{
        public interface IAccount
        {
                string Withdraw (int amount);
                string Deposit (int amount);
                double InterestRate();
        }
}
```

2. Criando classe concreta - conta poupança

```
namespace FactoryPattern
{
///
/// Concrete class SavingsAccount
///

        public class SavingsAccount : IAccount
        {
        #region IAccount Members

                public string Withdraw(int amount)
                {
                        throw new NotImplementedException();
                }

                public string Deposit(int amount)
                {
                        throw new NotImplementedException();
                }

                public double InterestRate()
                {
                        return 12.5;
                }

        #endregion
        }
}
```

3. Criando classe concreta - conta corrente

```
namespace FactoryPattern
{

///
/// Concrete class CheckingAccount
///

        public class CheckingAccount : IAccount
        {
                #region IAccount Members

                public string Withdraw(int amount)
                {
                        throw new NotImplementedException();
                }

                public string Deposit(int amount)
                {
                        throw new NotImplementedException();
                }

                public double InterestRate()
                {
                        return 10.24;
                }

        #endregion
        }
}
```

4. Criando objeto no modelo Factory Pattern

```
namespace FactoryPattern
{
// FactoryObject Enum to configure object
        public enum FactoryObject
        {
                SavingAccount,
                CheckingAccount
        }
}
```

5. Criando classe no modelo Factory Pattern

```
namespace FactoryPattern
{
///
/// Factory class to create object
///

        public static class Factory
        {
                public static IAccount CreateObject(FactoryObject factoryObject)
                {
                        IAccount objIAccount = null;

                        switch (factoryObject)
                        {
                        case FactoryObject.SavingAccount:
                        objIAccount = new SavingsAccount();
                        break;

                        case FactoryObject.CheckingAccount:
                        objIAccount = new CheckingAccount();
                        break;

                        default:
                        break;
                        }

                return objIAccount;
                }
        }
}
```

6. Acesso pelo cliente

```
namespace FactoryPattern
{
        public class Program
        {
                public static void Main(string[] args)
                {
                //Create object by factory pattern
                IAccount objSavingAccount = Factory.CreateObject(FactoryObject.SavingAccount);
                IAccount objCheckingAccount =
                Factory.CreateObject(FactoryObject.CheckingAccount);

                //Access object
                Console.WriteLine("Saving Account Interest Rate: " +
                objSavingAccount.InterestRate());
                Console.WriteLine("Checking Account Interest Rate: " +
                objCheckingAccount.InterestRate());

                Console.ReadLine();

                }
        }
}
```