**Strategy Pattern**

|  |
| --- |
| **Interface** |
| //strategy interface  public interface orderBehaviour  {  void Serve();  }    **Implementations of Behaviour**  #region strategy implement classes  class DineIn : orderBehaviour  {  public void Serve()  {  MessageBox.Show("Your Table is ready to be served");  }  }  class TakeAway : orderBehaviour  {  public void Serve()  {  MessageBox.Show("Your meal is ready for take away");  }  } |
| **Context class** |
| public Order(int orderid, List<Item> l,int price)  {  OrderID = 1;  OrderPrice = price;  OrderItems = l;  }  **public void SetOrderType(orderBehaviour orderBehaviour)**  **{**  **this.orderBehaviour = orderBehaviour;**  **}**  **public string ShowOrderType()**  **{**  **if (orderBehaviour is TakeAway)**  **{**  **return "Take Away";**  **} else**  **return "Dine In";**  **}**    // to change orderBehaviour on runtime  **public void ChangeOrderType(orderBehaviour orderBehaviour)**  **{**  **this.orderBehaviour = orderBehaviour;**  **}**    /// TO save order to sql  public void ServeOrder()  {  if ( OrderID > 0)  {  SqlConnection con = connection.Get();  SqlCommand cmd = new SqlCommand("insert into Order\_Table(id,Items,Price,Type,Quantity,DateTime) values(@id,@items,@price,@type,@quantity,@date)", con);  con.Open();  cmd.Parameters.AddWithValue("@id", OrderID);  string itemsList = "";  int quan = 0 ;  foreach (Item item in OrderItems)  {  itemsList +="[" + item.Id +" "+ item.Name + "]";  quan += item.quantity;  }    cmd.Parameters.AddWithValue("@items", itemsList);  cmd.Parameters.AddWithValue("@price", OrderPrice);  cmd.Parameters.AddWithValue("@type", ShowOrderType());  cmd.Parameters.AddWithValue("@quantity", quan);  cmd.Parameters.AddWithValue("@date", DateTime.Now.ToString());  cmd.ExecuteNonQuery();  con.Close();  MessageBox.Show("Record Inserted Successfully");  }  else  {  MessageBox.Show("Please Provide Details!");  }  **orderBehaviour.Serve();**  }    } |
| **Class Diagram** |
| **//diagram** |

**Singleton Pattern**

|  |
| --- |
| **Singleton Class** |
| public class connection : SQLConnector  {    private connection() { }    //singleton pattern  private static SqlConnection Sc = null; **//single object instance**  public static SqlConnection Get()  **//get instance method**  {  if (Sc == null)  {  Sc = new SqlConnection();  Sc.ConnectionString = @"Data Source=.;Initial Catalog=McSQL;Integrated Security=True";  Sc.Open();    }  return Sc;    }  } |
| **Classes that use this singleton object** |
| * Form1.cs * Item\_Manager.cs * Order.cs * OrderHistory.cs |
| **Example: Code used in Item\_Manager class** |
| public void AddItem(Item item)  {  //insert query    if (item != null)  {  **SqlConnection con = connection.Get(); //asking for SqlConnection instance**  SqlCommand cmd = new SqlCommand("insert into Item\_Table(id,Name,Price,Category) values(@id,@name,@price,@category)", con);  con.Open();  cmd.Parameters.AddWithValue("@id", item.Id);  cmd.Parameters.AddWithValue("@name", item.Name);  cmd.Parameters.AddWithValue("@price", item.price);  cmd.Parameters.AddWithValue("@category", item.Category);  cmd.ExecuteNonQuery();  con.Close();  MessageBox.Show("Record Inserted Successfully");  }  else  {  MessageBox.Show("Please Provide Details!");  }  } |
| **Class Diagram** |
| **//diagram** |

**Factory Pattern**

|  |
| --- |
| **Factory Interfaces** |
| **//Item Abstract class // Interface**  public abstract class Item  {  public int Id { get; set; }  public string Name { get; set; }  public string price { get; set; }  public string Category { get; set; }    public int quantity { get; set; }    public string getPrice(String name)  {  return price;  }  }  **//Implementations of Item**  class Burger : Item  {  }  class IceCream : Item  {  }  class Drinks : Item  {  } |
| **Factory class** |
| class ItemsFactory  {  public Item CreateItem(string type)  {  Item item;    if (type.Equals("Burger"))  item = new Burger();  else if (type.Equals("IceCream"))  item = new IceCream();    else if (type.Equals("Drinks"))  item = new Drinks();  else  {  MessageBox.Show("Invalid Item");  item = new Burger();  }  return item;  }  } |
| **Example : calling factory in Order.cs**  OrderItem() **asks** for Item from ItemsFactory() by passing only item type in string |
| public Item OrderItem(string type)  {    Item item;  ItemsFactory itemsFactory = new ItemsFactory();    item = itemsFactory.CreateItem(type);  return item;    } |
| **Diagram** |
| //diagram |

**Adapter Pattern**

|  |
| --- |
| **Adapter Interface** |
| public abstract class SQLConnector  {  public abstract SqlConnection GetSql();  public abstract MySqlConnection GetMySql();  } |
| **Implementations**  public class connection : SQLConnector  {    private connection() { }    //adapter pattern  //This will return a mySql connection  public override mySqlConnection.MySqlConnection GetMySql()  {  mySqlConnection.MySqlConnection mySql = new mySqlConnection.MySqlConnection();  return mySql;  }    //This will return a MSSQLconnection  public override SqlConnection GetSql()  {  return Get();  }  //singleton pattern  private static SqlConnection Sc = null; //single object  public static SqlConnection Get() //get instance method  {  **//this code is already defined in Singleton pattern**  }  } |
| **Adapter Class**  **// Here is MySQLConnection is adapting connection.cs**  public class mySqlConnection : SQLConnector  {  string myConnectionString = "server=localhost;database=testDB;uid=root;pwd=abc123;";    private static MySqlConnection Sc = null; //single object  public static MySqlConnection Get()  {  if (Sc == null)  {  Sc = new MySqlConnection();  Sc.ConnectionString = @"Data Source=.;Initial Catalog=McSQL;Integrated Security=True";  Sc.Open();    }  return Sc;  }  } |
| **Class Digram** |
| //Diagram |