**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»**

**Кафедра**

**Автоматизованих систем обробки інформації та управління**

**Звіт**

**з комп’ютерного практикуму №5**

**з дисципліни «Технології створення програмних продуктів»**

**Виконав:**

**Михайленко О.М.**

**Студент гр. ІС-41, ФІОТ, 3 курс,**

**залікова книжка IC-4118**

Київ 2017

**Породжуючі шаблони**

**1)Фабричний метод:**

abstract class AbstractTool : PictureBox

{

public abstract Control FactoryMethod();

}

class BucketTool : AbstractTool

{

Color currentColor;

public BucketTool(Image image)

{

…

}

public override Control FactoryMethod()

{

PictureBox newObject = new PictureBox();

newObject.BackColor = currentColor;

return newObject;

}

class ObjectTool : AbstractTool //Control

{

public ObjectTool(Image image)

{

…

}

public override Control FactoryMethod()

{

PictureBox newObject = new PictureBox();

newObject.Image = this.Image;

newObject.SizeMode = PictureBoxSizeMode.StretchImage;

newObject.BackColor = Color.Transparent;

return newObject;

}

}

AbstractTool currentTool;

currentTool = new BucketTool(pictureBox2.Image);

…

this.createNewMapObject(currentTool.FactoryMethod(), this.PointToClient(Cursor.Position));

**2-3)Одинак + Лінива ініціалізація:**

class Map : PictureBox

{

private static Map instance;

public static Map Instance

{

get { return instance ?? (instance = new Map()); }

}

protected Map() { }

}

**Структурні шаблони**

**1)Адаптер**

class Map : PictureBox

{

private static Map instance;

public static Map Instance

{

get { return instance ?? (instance = new Map()); }

}

protected Map() { }

}

class SerializableMap

{

int Width { get; set; }

int Height { get; set; }

Point Location { get; set; }

Image Image { get; set; }

Color BackColor { get; set; }

List<CustomPictureBox> Controls { get; set; }

public void restore(Map e)

{

e.Width = Width;

e.Height = Height;

e.Location = Location;

e.Image = Image;

e.BackColor = BackColor;

foreach(CustomPictureBox current in Controls)

{

e.Controls.Add(current.restore());

}

}

public SerializableMap(Map e)

{

Width = e.Width;

Height = e.Height;

Location = e.Location;

Image = e.Image;

BackColor = e.BackColor;

Controls = new List<CustomPictureBox>();

foreach(Control current in e.Controls)

{

Controls.Add(new CustomPictureBox((PictureBox)current));

}

}

}

**2)Декоратор**

abstract class Component

{

public abstract void DoSomething();

}

class ConcreteComponent : Component

{

public override void DoSomething()

{

Console.Write("Hello");

}

}

abstract class Decorator : Component

{

protected Component component;

public void SetComponent(Component component)

{

this.component = component;

}

public override void DoSomething()

{

if (component != null)

component.DoSomething();

}

}

class ConcreteDecorator : Decorator

{

public override void DoSomething()

{

base.DoSomething();

}

}

**3)Компанувальник**

class Map : PictureBox

{

private static Map instance;

public static Map Instance

{

get { return instance ?? (instance = new Map()); }

}

protected Map() { }

}

foreach (Control c in map.Controls)

{

if (map.Controls.GetChildIndex(c) >= numberOfLines()

&& c.Location == newObject.Location

&& map.Controls.GetChildIndex(c) != map.Controls.GetChildIndex(newObject))

{

if(newObject.BackColor == Color.Transparent)

newObject.BackColor = c.BackColor;

c.Dispose();

}

}

**Шаблони поведінки**

**1)Ітератор**

foreach(Control current in e.Controls)

{

Controls.Add(new CustomPictureBox((PictureBox)current));

}

**2)Команда**

this.pictureBox4.Click += new System.EventHandler(this.pictureBox4\_Click);

newObject.MouseClick += map\_MouseClick;

private void map\_MouseClick(object sender, MouseEventArgs e)

{

if (currentTool != null)

{

if((string)currentTool.Tag == "DeleteTool")

{

deleteMapObject(this.PointToClient(Cursor.Position));

}

else

this.createNewMapObject(currentTool.FactoryMethod(), this.PointToClient(Cursor.Position));

}

}

**3)Нульовий об’єкт**

AbstractClass realClass = new RealClass();

realClass.doSomething();

AbstractClass nullClass = new NullClass();

nullClass.doSomething();

public abstract class AbstractClass

{

public abstract void doSomething();

}

public class RealClass : AbstractClass

{

public override void doSomething()

{

Console.WriteLine("RealClass::doSomething");

}

}

public class NullClass : AbstractClass

{

public override void doSomething()

{

}

}