Homework 3

单例模式

```
package com.goat;

class LazyNotThreadSafeSingleton {
    private static LazyNotThreadSafeSingleton instance;

    private LazyNotThreadSafeSingleton() {
    }

    public static LazyNotThreadSafeSingleton getInstance() {
        if (instance == null) {
            instance = new LazyNotThreadSafeSingleton();
        }

        return instance;
    }
}

class LazyThreadSafeSingleton {
    private static LazyThreadSafeSingleton instance;

    private LazyThreadSafeSingleton() {
    }

    public static synchronized LazyThreadSafeSingleton getInstance() {
        if (instance == null) {
            instance = new LazyThreadSafeSingleton();
        }

        return instance;
    }
}

class EagerThreadSafeSingleton {
    private static EagerThreadSafeSingleton instance = new
EagerThreadSafeSingleton();

    private EagerThreadSafeSingleton() {
    }

    public static EagerThreadSafeSingleton getInstance() {
        return instance;
    }
}

class LazyThreadSafeDCLSingleton {
    private volatile static LazyThreadSafeDCLSingleton instance;

    private LazyThreadSafeDCLSingleton() {
    }

    public static LazyThreadSafeDCLSingleton getInstance() {
        if (instance == null) {
            synchronized (LazyThreadSafeDCLSingleton.class) {
                if (instance == null) {
                    instance = new LazyThreadSafeDCLSingleton();
                }
            }
        }

        return instance;
    }
}

class LazyThreadSafeInnerStaticSingleton {
    private static class SingletonHolder {
        private static final LazyThreadSafeInnerStaticSingleton instance = new
LazyThreadSafeInnerStaticSingleton();
    }

    private LazyThreadSafeInnerStaticSingleton() {
    }

    public static LazyThreadSafeInnerStaticSingleton getInstance() {
        return SingletonHolder.instance;
    }
}

enum LazyThreadSafeEnumSingleton {
    INSTANCE
}

public class Main {

    public static void main(String[] args) {
        LazyNotThreadSafeSingleton lntS1 =
LazyNotThreadSafeSingleton.getInstance();
        LazyNotThreadSafeSingleton lntS2 =
LazyNotThreadSafeSingleton.getInstance();
        System.out.println("two LazyNotThreadSafeSingleton objects are equal?
" + lntS1.equals(lntS2));
```

```
        LazyThreadSafeSingleton ltS1 = LazyThreadSafeSingleton.getInstance();
        LazyThreadSafeSingleton ltS2 = LazyThreadSafeSingleton.getInstance();
        System.out.println("two LazyThreadSafeSingleton objects are equal? " +
ltS1.equals(ltS2));

        EagerThreadSafeSingleton etS1 =
EagerThreadSafeSingleton.getInstance();
        EagerThreadSafeSingleton etS2 =
EagerThreadSafeSingleton.getInstance();
        System.out.println("two EagerThreadSafeSingleton objects are equal? "
+ etS1.equals(etS2));

        LazyThreadSafeDCLSingleton ltdclS1 =
LazyThreadSafeDCLSingleton.getInstance();
        LazyThreadSafeDCLSingleton ltdclS2 =
LazyThreadSafeDCLSingleton.getInstance();
        System.out.println("two LazyThreadSafeDCLSingleton objects are equal?
" + ltdclS1.equals(ltdclS2));

        LazyThreadSafeInnerStaticSingleton ltIsS1 =
LazyThreadSafeInnerStaticSingleton.getInstance();
        LazyThreadSafeInnerStaticSingleton ltIsS2 =
LazyThreadSafeInnerStaticSingleton.getInstance();
        System.out.println("two LazyThreadSafeInnerStaticSingleton objects are
equal? " + ltIsS1.equals(ltIsS2));

        System.out.println("two LazyThreadSafeEnumSingleton objects are equal?
" +
LazyThreadSafeEnumSingleton.INSTANCE.equals(LazyThreadSafeEnumSingleton.INSTAN
CE));
    }
}
```

组合模式

```
package com.ll;

import java.util.ArrayList;
import java.util.List;

interface Component {
    void showComponentName();
}
```

```java
class WinForm implements Component {
    private List<Component> componentList = new ArrayList<Component>();
    private final String name;

    public WinForm(String name) {
        this.name = name;
    }

    public void add(Component component) {
        this.componentList.add(component);
    }

    @Override
    public void showComponentName() {
        System.out.println("WinForm" + "(" + name + ")");
        for (Component component : this.componentList) {
            component.showComponentName();
        }

    }

}

class Picture implements Component {
    private final String name;

    public Picture(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("Picture(" + this.name + ")");
    }
}

class Button implements Component {
    private final String name;

    public Button(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("Button(" + this.name + ")");
    }
}
```

```java
class Frame implements Component {
    private List<Component> componentList = new ArrayList<Component>();
    private final String name;

    public Frame(String name) {
        this.name = name;
    }

    public void add(Component component) {
        this.componentList.add(component);
    }

    @Override
    public void showComponentName() {
        System.out.println("Frame" + "(" + name + ")");
        for (Component component : this.componentList) {
            component.showComponentName();
        }

    }
}

class Label implements Component {
    private final String name;

    public Label(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("Label(" + this.name + ")");
    }
}

class TextBox implements Component {
    private final String name;

    public TextBox(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("TextBox(" + this.name + ")");
    }
}

class PasswordBox implements Component {
    private final String name;

    public PasswordBox(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("PasswordBox(" + this.name + ")");
    }
}

class CheckBox implements Component {
    private final String name;

    public CheckBox(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("CheckBox(" + this.name + ")");
    }
}

class LinkLable implements Component {
    private final String name;

    public LinkLable(String name) {
        this.name = name;
    }

    @Override
    public void showComponentName() {
        System.out.println("LinkLable(" + this.name + ")");
    }
}

public class Main {

    public static void main(String[] args) {
        // write your code here
        WinForm winForm = new WinForm("WINDOW窗口");
        Picture pic = new Picture("LOGO图片");
        Button butLogin = new Button("登录");
        Button butRegister = new Button("注册");
        Frame frame1 = new Frame("FRAME1");
        Label usernameLabel = new Label("用户名");
        TextBox text = new TextBox("文本框");
        Label password = new Label("密码");
        PasswordBox pbox = new PasswordBox("密码框");
        CheckBox cbox = new CheckBox("复选框");
        TextBox tbox = new TextBox("记住用户名");
        LinkLable llabel = new LinkLable("忘记密码");

        frame1.add(usernameLabel);
        frame1.add(text);
        frame1.add(password);
        frame1.add(pbox);
        frame1.add(cbox);
        frame1.add(tbox);
        frame1.add(llabel);
        winForm.add(pic);
        winForm.add(butLogin);
        winForm.add(butRegister);
        winForm.add(frame1);
        winForm.showComponentName();
    }
}
```