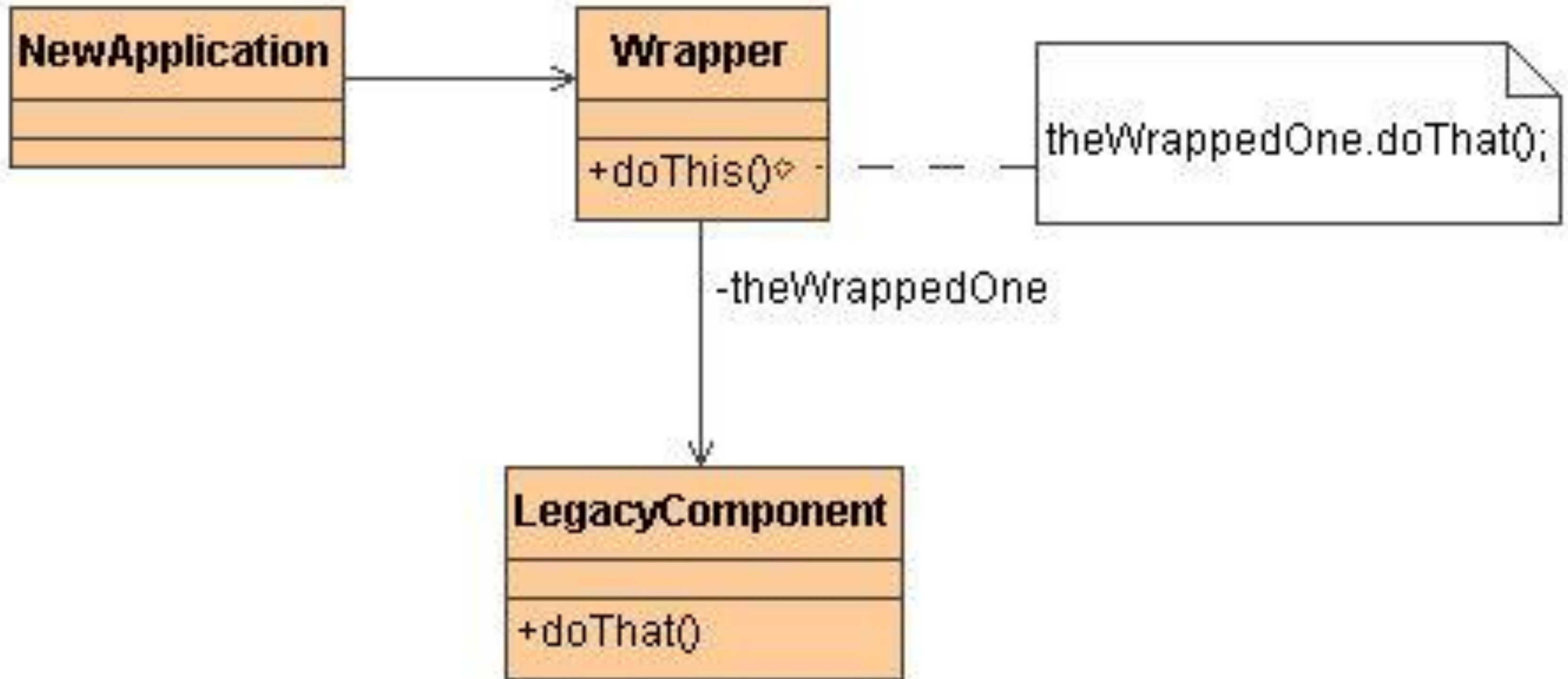
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Паттерны (шаблоны) проектирования

Структурные паттерны (примеры)

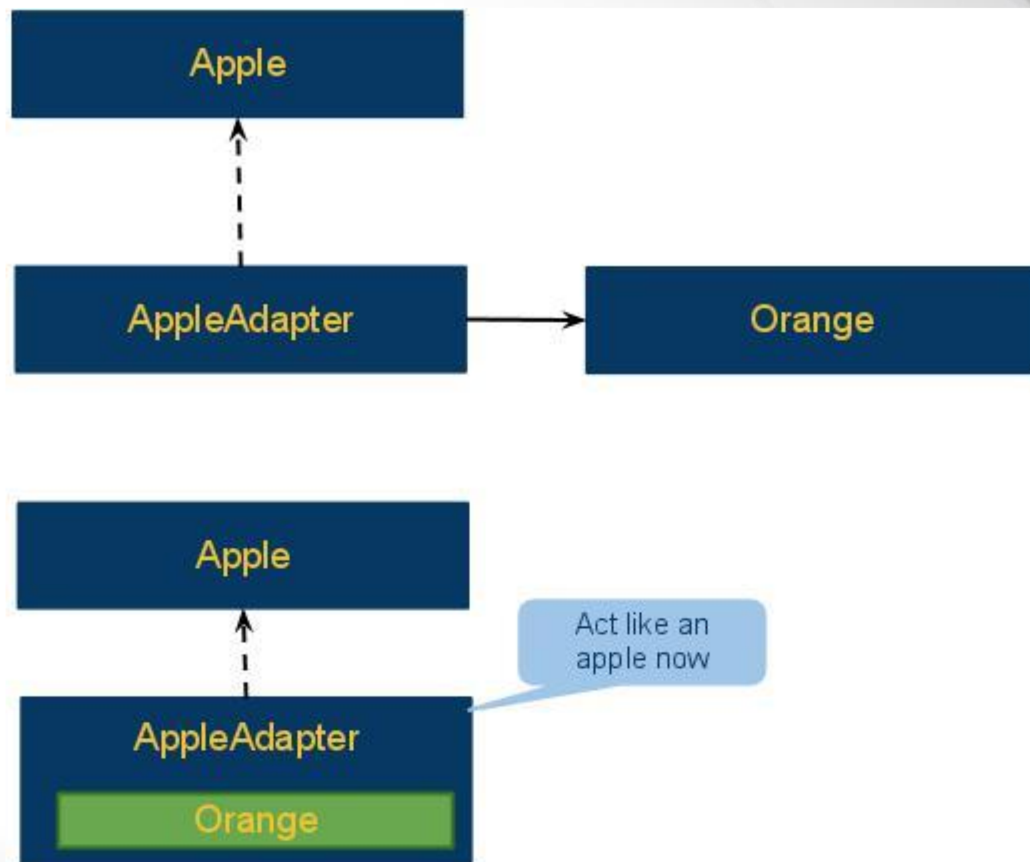
Евгений Беркунский
http://www.berkut.mk.ua
eberkunsky@gmail.com

Adapter/Адаптер



Adapter/Адаптер

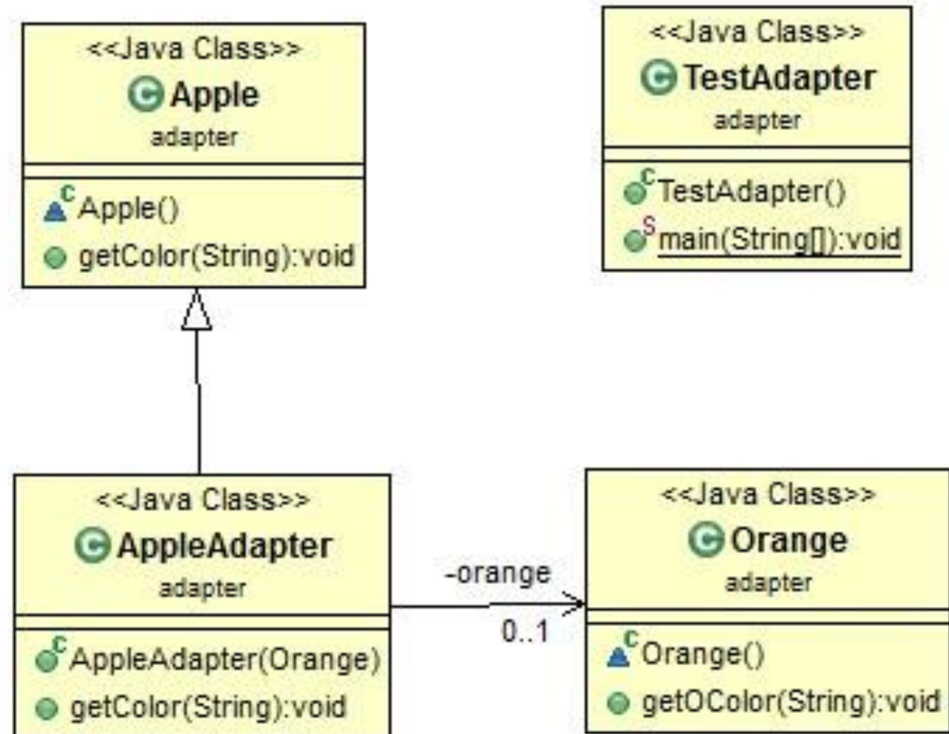
Яблок нет,
зато есть апельсины…



Adapter/Адаптер

Яблок нет,
зато есть апельсины…

… видимо, наколотые



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

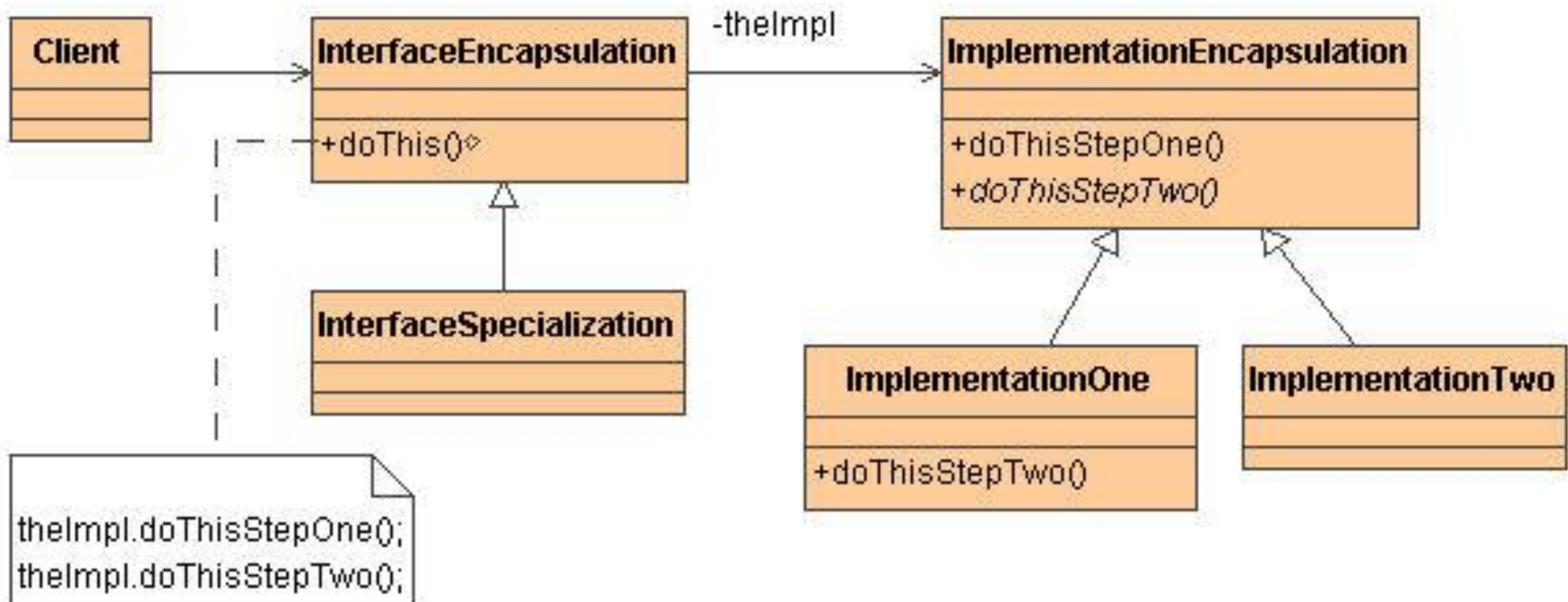
Adapter/Адаптер

```
class Apple {
    public void getAColor(String str) {
        System.out.println("Apple color is: " + str);
    }
}
class Orange {
    public void getOColor(String str) {
        System.out.println("Orange color is: " + str);
    }
}
class AppleAdapter extends Apple {
    private Orange orange;

    public AppleAdapter(Orange orange) {
        this.orange = orange;
    }

    public void getAColor(String str) {
        orange.getOColor(str);
    }
}
```

Adapter/Адаптер

```
public class TestAdapter {
    public static void main(String[] args) {
        Apple apple1 = new Apple();
        Apple apple2 = new Apple();
        apple1.getAColor("green");

        Orange orange = new Orange();

        AppleAdapter aa = new AppleAdapter(orange);
        aa.getAColor("red");
    }
}
```

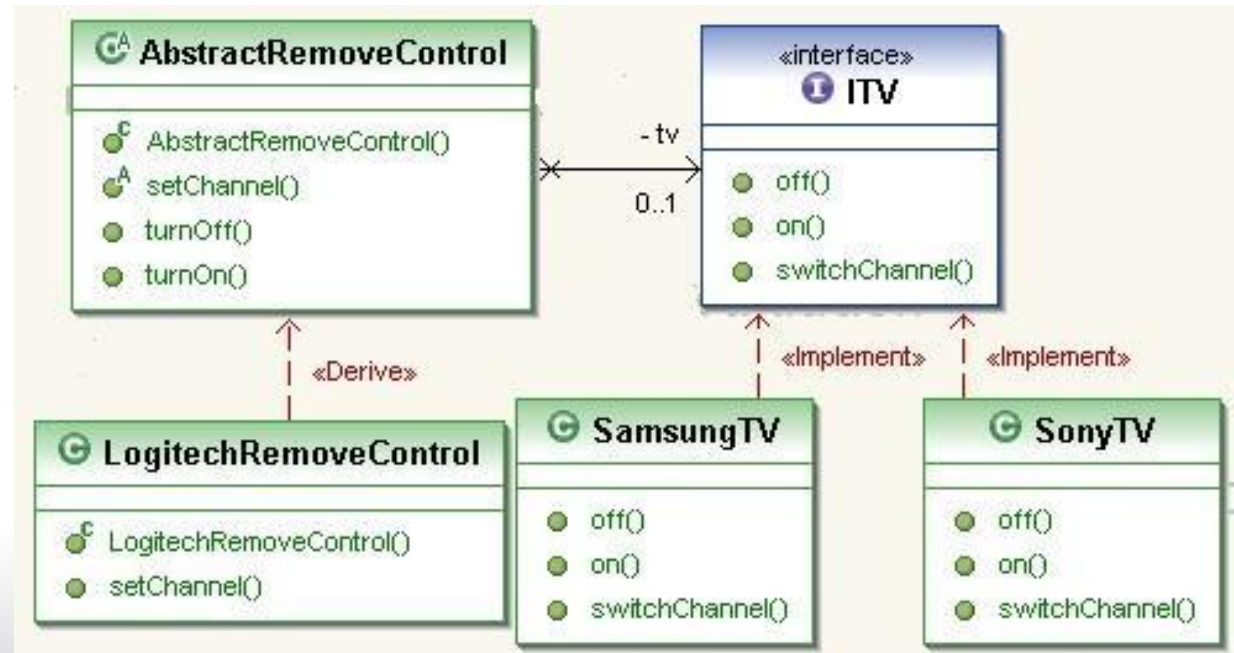
Bridge/Мост



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Bridge/Мост

Пример телевизора и пульта ДУ демонстрирует два слоя абстракции.
У вас есть интерфейс для ТВ и абстрактный класс для пульта дистанционного управления.



Bridge/Мост

Сначала определим интерфейс ТВ

```
public interface ITV {
    public void on();
    public void off();
    public void switchChannel(int channel);
}
```

Bridge/Мост

Samsung реализует интерфейс ТВ

```
public class SamsungTV implements ITV {
    @Override
    public void on() {
        System.out.println("Samsung is turned on.");
    }

    @Override
    public void off() {
        System.out.println("Samsung is turned off.");
    }

    @Override
    public void switchChannel(int channel) {
        System.out.println("Samsung: channel - " + channel);
    }
}
```

Bridge/Мост

Sony тоже реализует интерфейс ТВ

```
public class SonyTV implements ITV {

    @Override
    public void on() {
        System.out.println("Sony is turned on.");
    }

    @Override
    public void off() {
        System.out.println("Sony is turned off.");
    }

    @Override
    public void switchChannel(int channel) {
        System.out.println("Sony: channel - " + channel);
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Bridge/Мост

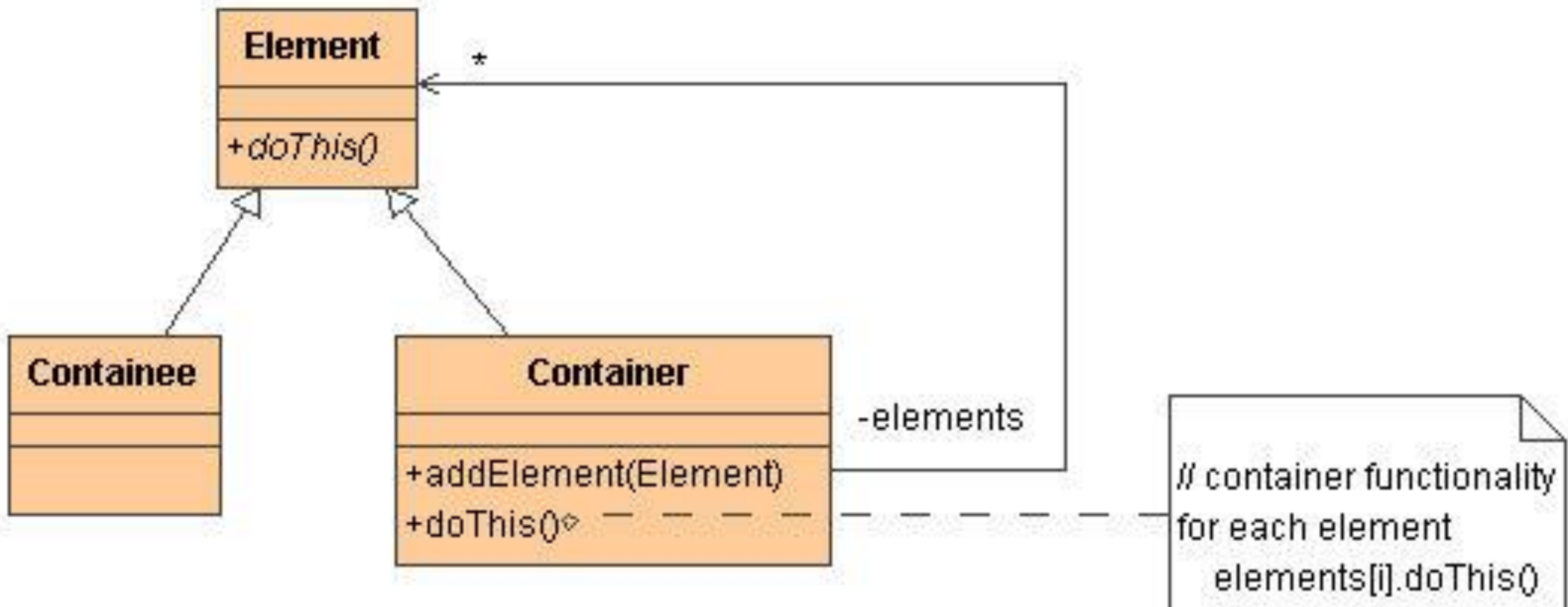
Пульт ДУ содержит ссылку на TV

```
public abstract class AbstractRemoteControl {
    private ITV tv;

    public AbstractRemoteControl(ITV tv){
        this.tv = tv;
    }

    public void turnOn(){
        tv.on();
    }

    public void turnOff(){
        tv.off();
    }

    public void setChannel(int channel){
        tv.switchChannel(channel);
    }
}
```

Bridge/Мост

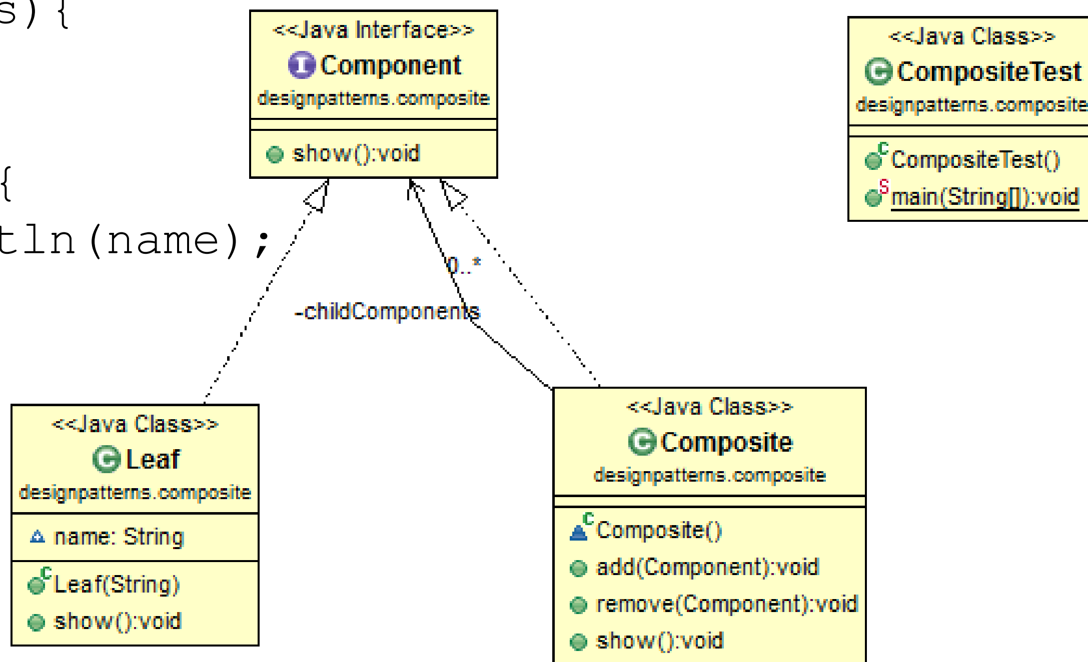
```
public class LogitechRemoteControl
        extends AbstractRemoteControl {
    public LogitechRemoteControl(ITV tv) {
        super(tv);
    }
    public void setChannelKeyboard(int channel){
        setChannel(channel);
        System.out.println(
            "Logitech use keyword to set channel.");
    }
}
public class Main {
    public static void main(String[] args){
        ITV tv = new SonyTV();
        LogitechRemoteControl lrc =
                    new LogitechRemoteControl(tv);
        lrc.setChannelKeyboard(100);
    }
}
```

Composite/Компоновщик



Composite/Компоновщик

```
interface Component {
    public void show();
}

class Leaf implements Component {
    String name;
    public Leaf(String s){
        name = s;
    }
    public void show() {
        System.out.println(name);
    }
}
```



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Composite/Компоновщик

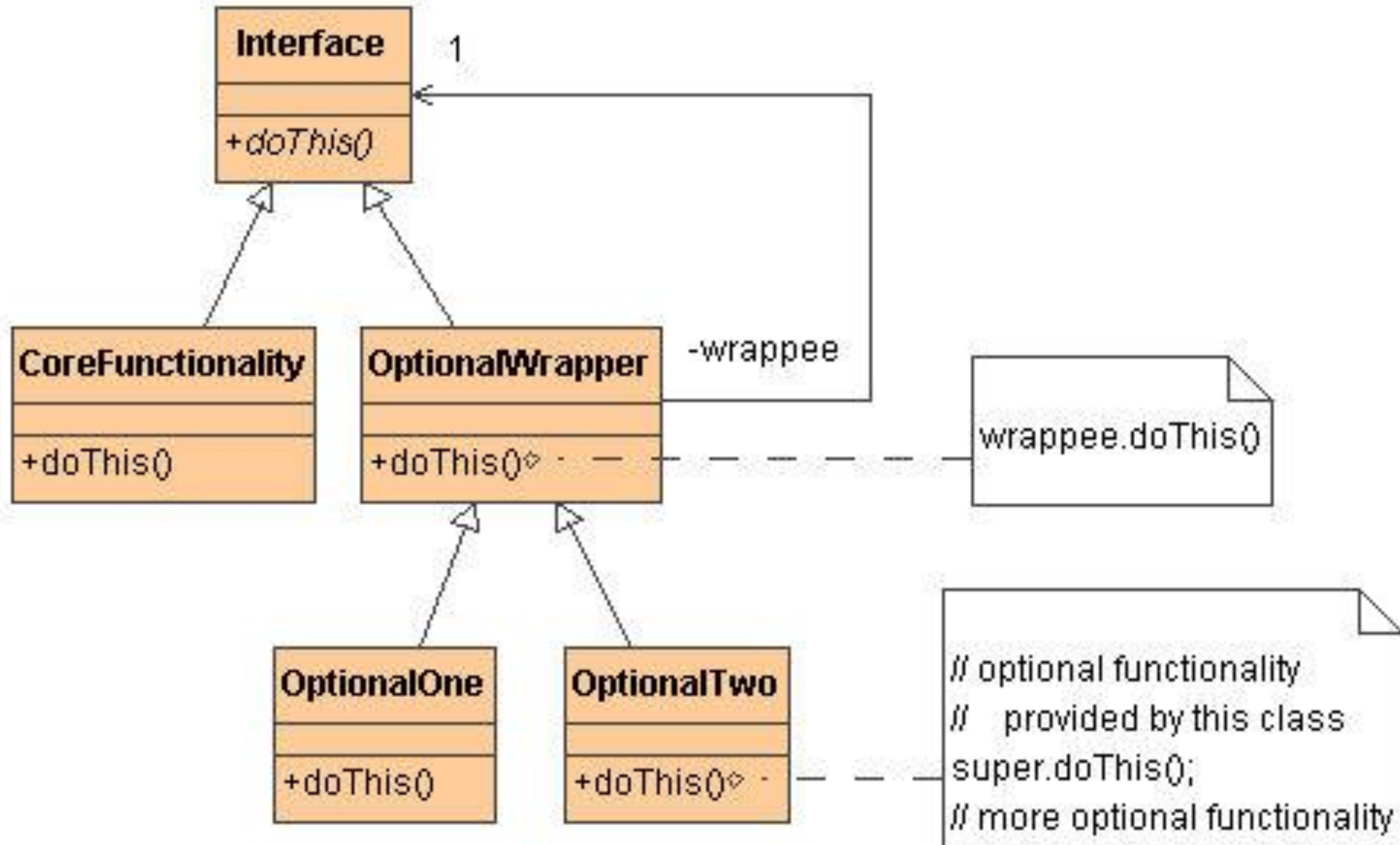
```
class Composite implements Component {

    private List<Component> childComponents =
        new ArrayList<Component>();

    public void add(Component component) {
       childComponents.add(component);
    }

    public void remove(Component component) {
       childComponents.remove(component);
    }

    @Override
    public void show() {
        for (Component component : childComponents) {
            component.show();
        }
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Composite/Компоновщик

```
public class CompositeTest {
    public static void main(String[] args) {
        Leaf leaf1 = new Leaf("1");
        Leaf leaf2 = new Leaf("2");
        Leaf leaf3 = new Leaf("3");
        Leaf leaf4 = new Leaf("4");
        Leaf leaf5 = new Leaf("5");
        Composite composite1 = new Composite();
        composite1.add(leaf1);
        composite1.add(leaf2);
        Composite composite2 = new Composite();
        composite2.add(leaf3);
        composite2.add(leaf4);
        composite2.add(leaf5);
        composite1.add(composite2);
        composite1.show();
    }
}
```

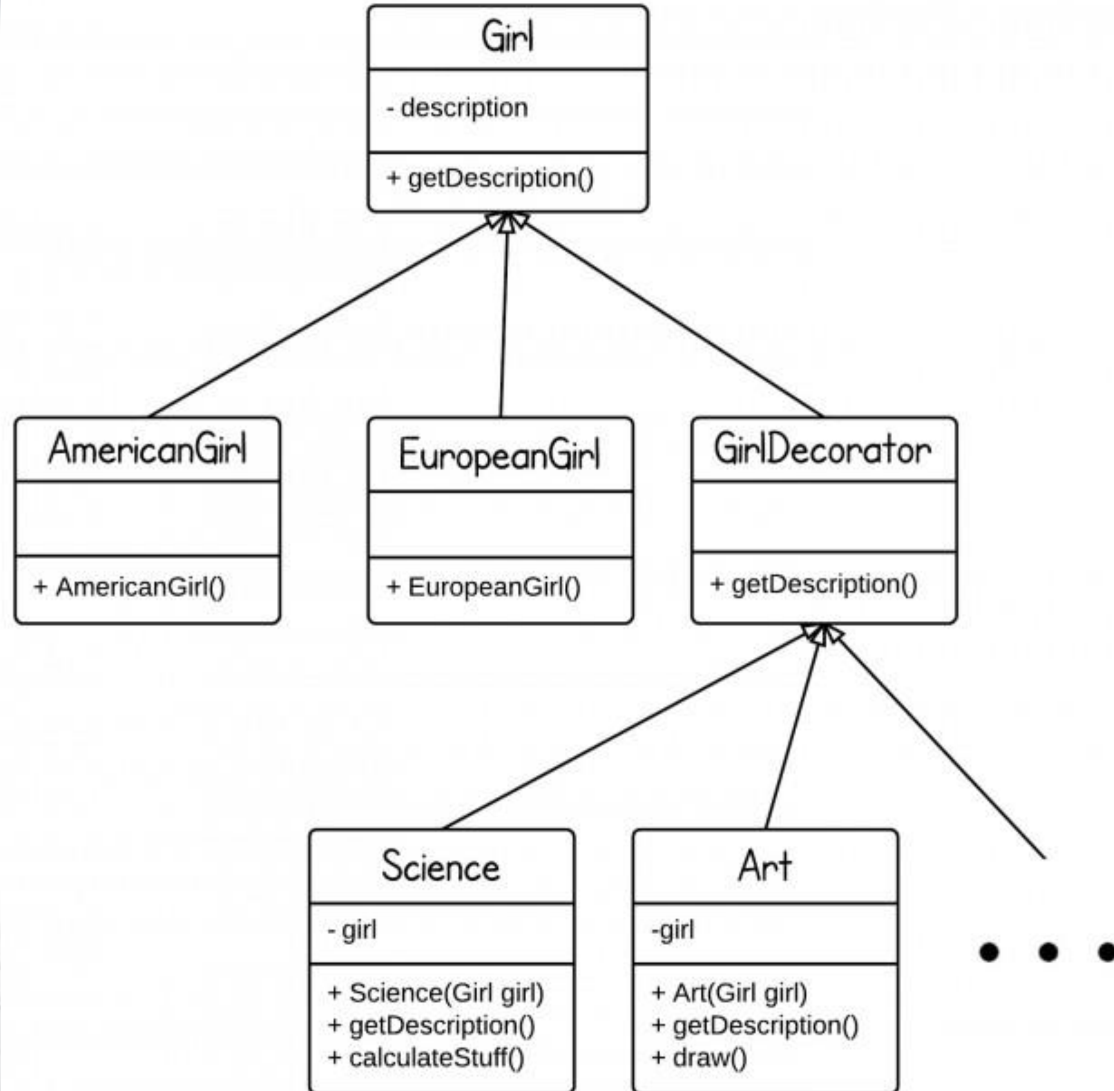
Decorator/Декоратор



Decorator/Декоратор

Предположим, что на некотором сайте Вы ищете девушку.
Есть девушки из разных стран, таких как Америка, Китай, Франция и т.д.
Они могут быть разные по внешности и увлечениям.
Если каждый тип девушки - это отдельный класс Java, то получатся тысячи классов.
Это является серьезной проблемой, которая называется «взрыв классов».
Кроме того, эта конструкция не является расширяемой.
Всякий раз, когда появляется новый тип девушки, новый класс должен быть создан.

Decorator/Декоратор



Decorator/Декоратор

```
public abstract class Girl {
    String description = "no particular";
    public String getDescription(){
        return description;
    }
}

public class AmericanGirl extends Girl {
    public AmericanGirl(){
        description = "+American";
    }
}

public class EuropeanGirl extends Girl {
     public EuropeanGirl() {
        description = "+European";
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Decorator/Декоратор

```
public abstract class GirlDecorator extends Girl {
    public abstract String getDescription();
}

public class Science extends GirlDecorator {
    private Girl girl;
    public Science(Girl g) {
        girl = g;
    }
    @Override
    public String getDescription() {
        return girl.getDescription() + "+Like Science";
    }
    public void caltulateStuff() {
        System.out.println("scientific calculation!");
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Decorator/Декоратор

```
public abstract class GirlDecorator extends Girl {
    public abstract String getDescription();
}

public class Art extends GirlDecorator {
    private Girl girl;
    public Art(Girl g) {
        girl = g;
    }
    @Override
    public String getDescription() {
        return girl.getDescription() + "+Like Art";
    }
    public void draw() {
        System.out.println("draw pictures!");
    }
}
```

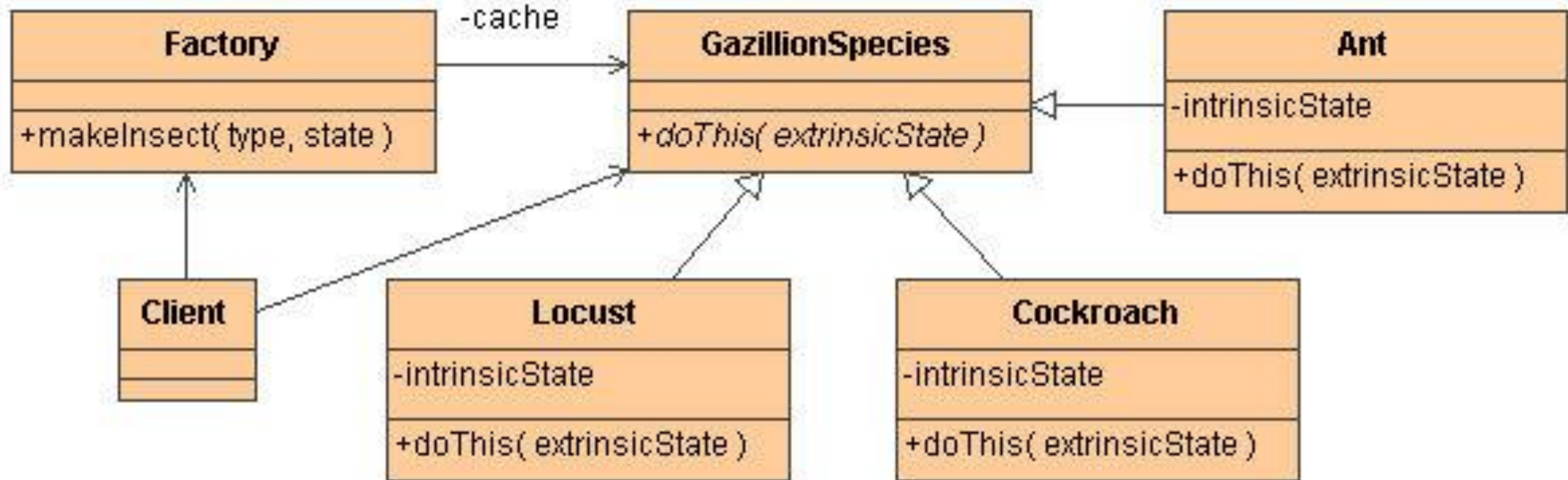
Decorator/Декоратор

```
public class Main {
    public static void main(String[] args) {
        Girl g1 = new AmericanGirl();
        System.out.println(g1.getDescription());
        Science g2 = new Science(g1);
        System.out.println(g2.getDescription());
        Art g3 = new Art(g2);
        System.out.println(g3.getDescription());
    }
}
```

Можно написать и такое:

```
Girl g = new Science(new Art(new AmericanGirl()));
```

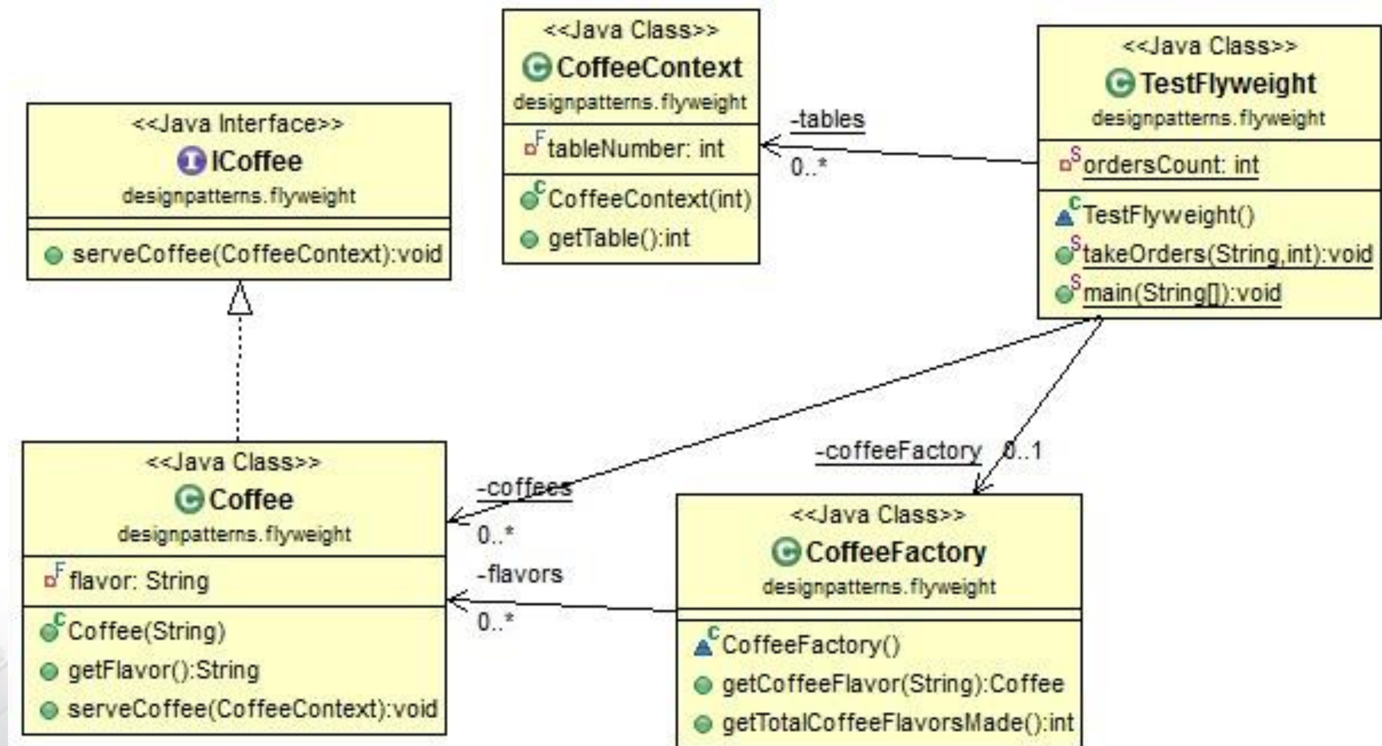
Flyweight/Приспособленец



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Flyweight/Приспособленец

Этот шаблон используется для минимизации использования памяти. Все, что он делает - это разделяет столько данных, сколько возможно с другими подобными объектами.



Flyweight/Приспособленец

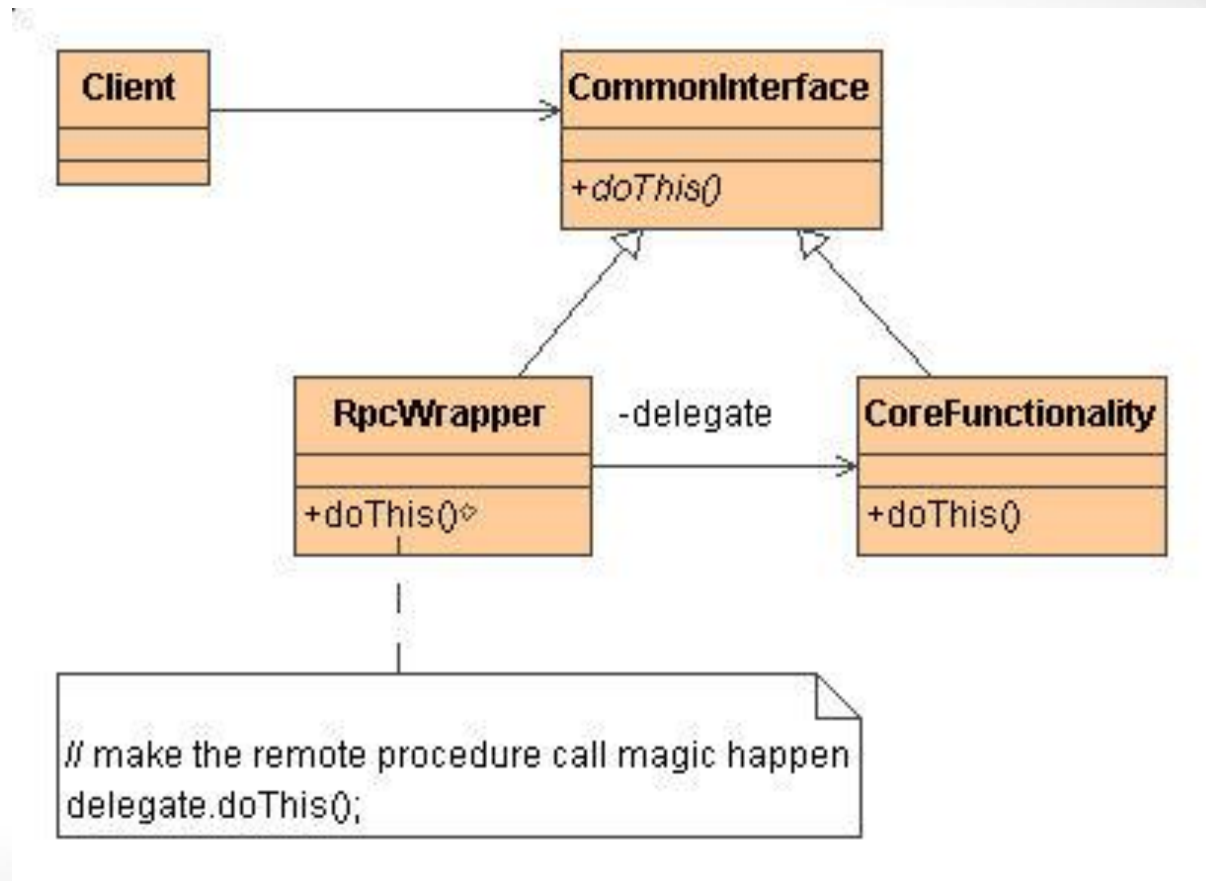
```
  interface ICoffee {
      public void serveCoffee(CoffeeContext
  context);
  }

// Concrete Flyweight object
class Coffee implements ICoffee {
    private final String flavor;
    public Coffee(String newFlavor) {
        this.flavor = newFlavor;
        System.out.println("Coffee is created! - " + flavor);
    }
    public String getFlavor() {
        return this.flavor;
    }
    public void serveCoffee(CoffeeContext context) {
        System.out.println("Serving " + flavor + " to table "
                    + context.getTable());
    }
}
```

Flyweight/Приспособленец

```
//The FlyweightFactory!
class CoffeeFactory {
    private HashMap<String, Coffee> flavors =
            new HashMap<String, Coffee>();
    public Coffee getCoffeeFlavor(String flavorName) {
        Coffee flavor = flavors.get(flavorName);
        if (flavor == null) {
            flavor = new Coffee(flavorName);
            flavors.put(flavorName, flavor);
        }
        return flavor;
    }
    public int getTotalCoffeeFlavorsMade() {
        return flavors.size();
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Proxy/Заместитель



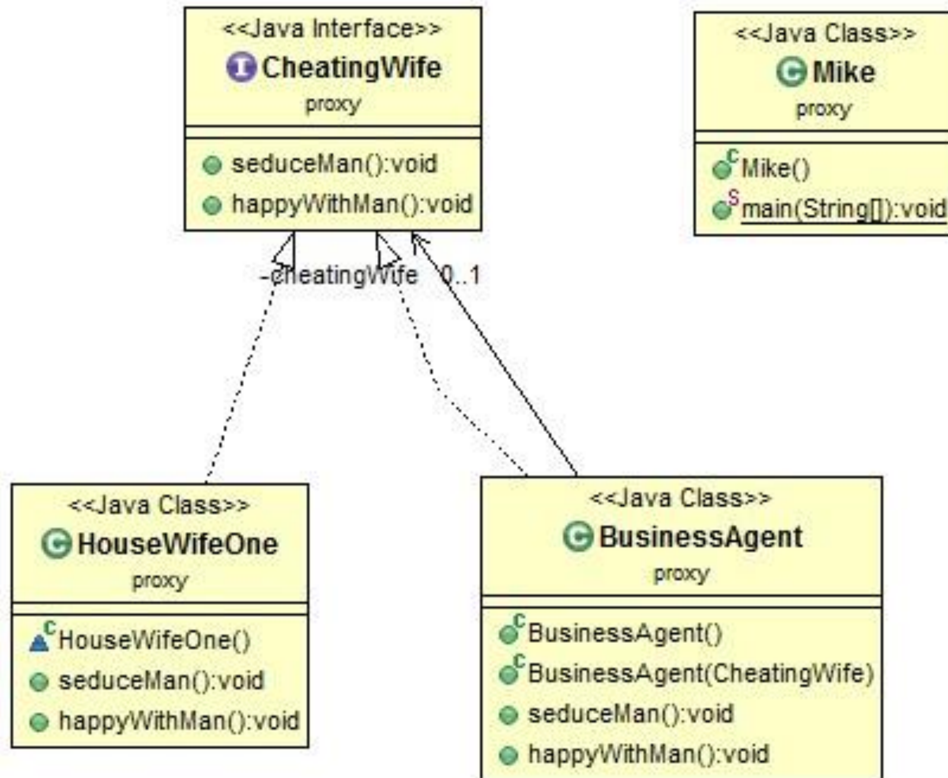
Proxy/Заместитель

Пусть этот пример будет на языке оригинала

Some bad man, for whatever reasons, always wants to sleep with some good man’s wife. Among those wives, some want to sleep with those bad men, but others do not.
The bad men can not ask directly to those wives. Because they are not sure whether the one being asked would like to do bad things.
It would be a very bad situation if he makes a bad judgment.
So there should be an agent/proxy to do this kind of business for those bad men.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Proxy/Заместитель



Proxy/Заместитель

```
interface CheatingWife {
// think about what this kind of women can do
    public void seduceMan();
// such as eye contact with men
    public void happyWithMan();
// happy what? You know that.
}

class HouseWifeOne implements CheatingWife {
    public void seduceMan() {
        System.out .println("HouseWifeOne seduce men, "+
              "such as making some sexy poses ...");
    }
    public void happyWithMan() {
        System.out.println("HouseWifeOne is happy with man");
    }
}
```

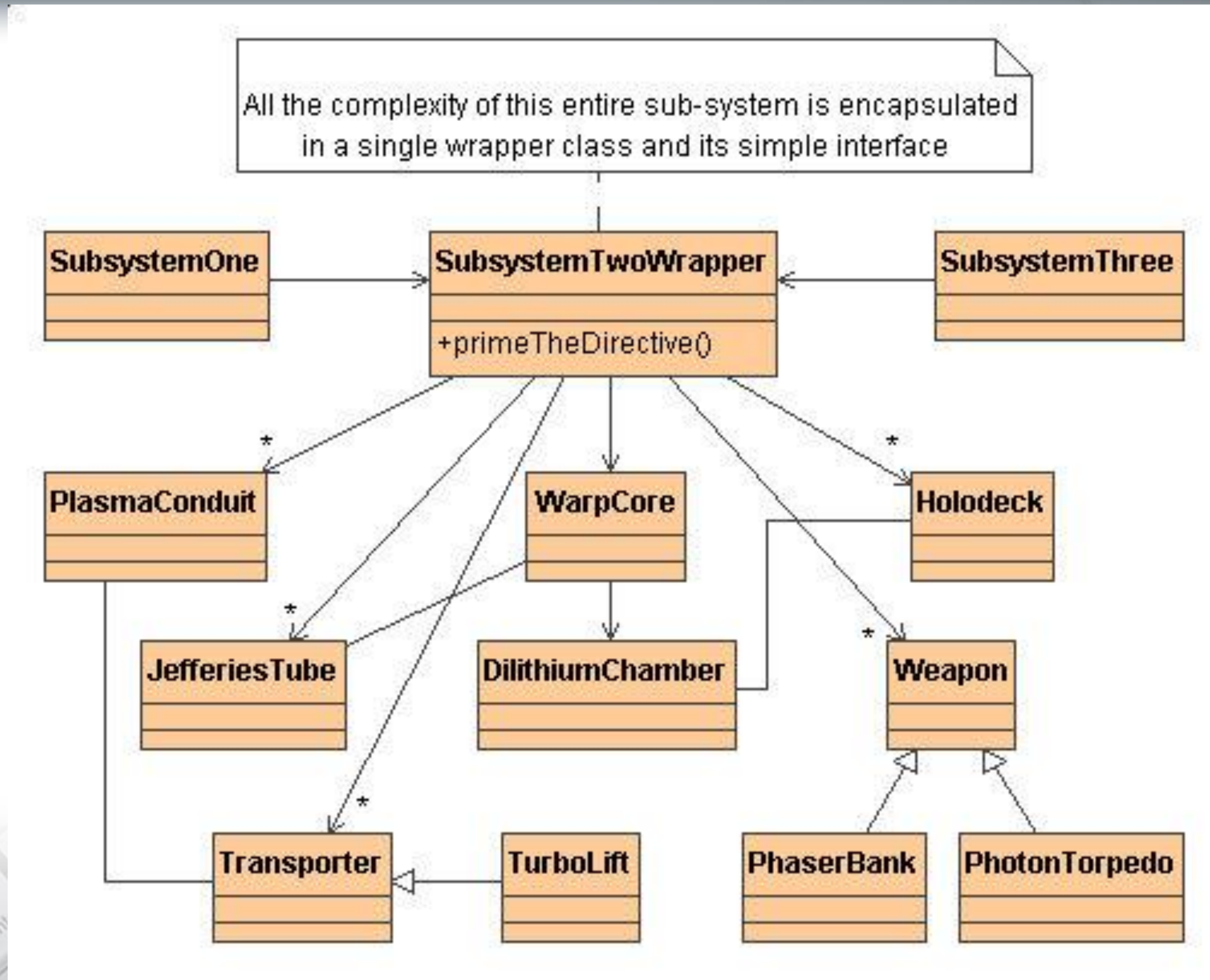
Proxy/Заместитель

```
class BusinessAgent implements CheatingWife {
    private CheatingWife cheatingWife;
    public BusinessAgent() {
        this.cheatingWife = new HouseWifeOne();
    }
    public BusinessAgent(CheatingWife cheatingWife) {
        this.cheatingWife = cheatingWife;
    }
    public void seduceMan() {
        this.cheatingWife.seduceMan();
    }
    public void happyWithMan() {
        this.cheatingWife.happyWithMan();
    }
}
```

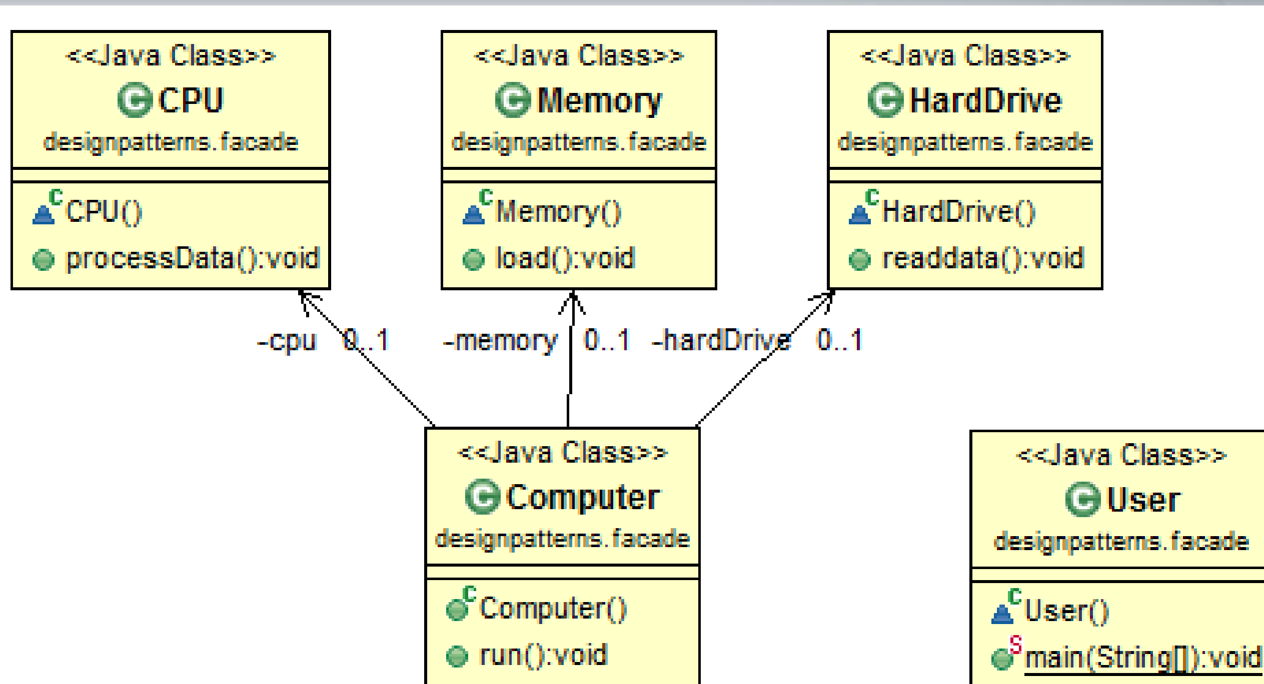
Proxy/Заместитель

```
// see? it looks that agent/proxy is doing
public class Mike {
    public static void main(String[] args) {
        BusinessAgent businessAgent = new BusinessAgent();
        businessAgent.seduceMan();
        businessAgent.happyWithMan();
    }
}
```

Facade/Фасад



Facade/Фасад




```
class User {
    public static void main(String[] args) {
        Computer computer = new Computer();
        computer.run();
    }
}
```

Facade/Фасад

```
// компоненты компьютера
class CPU { public void processData() { } }
class Memory { public void load() { } }
class HardDrive { public void readdata() { } }
/* Facade */
class Computer {
    private CPU cpu;
    private Memory memory;
    private HardDrive hardDrive;
    public Computer() {
        this.cpu = new CPU();
        this.memory = new Memory();
        this.hardDrive = new HardDrive();
    }
    public void run() {
        cpu.processData();
        memory.load();
        hardDrive.readdata();
    }
}
```

Структурные паттерны

Вопросы?

Структурные паттерны

Спасибо!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Паттерны (шаблоны) проектирования

Структурные паттерны (примеры)

Евгений Беркунский
http://berkut.homelinux.com
eberkunsky@gmail.com