Pola Desain Perangkat Lunak

[Week] 8 – Structural Pattern, Adapter

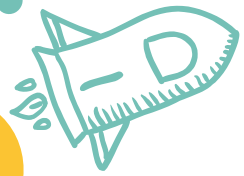
Prepared by: Tifanny Nabarian

Design Patterns Category

Creational Patterns	Creational patterns prescribe the way that objects are created.
Structural Patterns	• Structural patterns are concerned with how classes and objects are composed to form larger structures
Behavioral Patterns	• Behavioral patterns are concerned with algorithms and the assignment of responsibilities between objects.
Concurrency Patterns	• Concurrency patterns prescribe the way access to shared resources is coordinated or sequenced

Design Patterns Scope

		Purpose		
		Creational	Structural	Behavioral
Scope	Class	• Factory method	• Adapter	• Interpreter • Template method
	Object	• Abstract factory • Builder • Prototype • Singleton	• Adapter • Bridge • Composite • Decorator • Fasad • Flyweight • Proxy	• Chain of responsibility • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor



Structural Pattern

Adapter

Adapter Concept

Definisi GoF

- **Convert** the interface of a class into **another interface that clients expect**. Adapter lets classes work together that could not otherwise because of incompatible interfaces.

Real World Example

- A very common use of this pattern can be seen in an electrical outlet adapter/AC power adapter in international travels. These adapters act as a middleman when an electronic device (let's say, a laptop) that accepts a US power supply can be plugged into a European power outlet.

Adapter Concept

Computer World Example

- Suppose that you have an application that can be broadly classified into two parts: **user interface** (UI or front end) and **database** (back end). You can pass a data/object to database through the UI.
- For the 1st time, your database is **compatible** with those objects and can **store them smoothly**.
- Over a period of time, you may feel that you need to **upgrade your software**.
- But in this case, the first resistance comes from your database because **it cannot store these new types of objects**. In such a situation, you can use an **adapter** that takes care of the **conversion of the new objects to a compatible form** that your old database can accept.

Adapter Concept

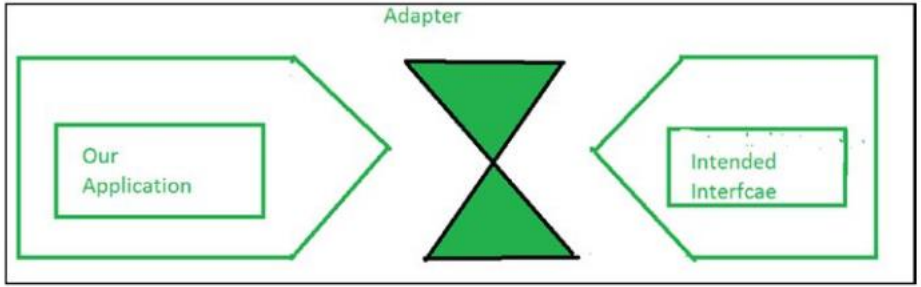


Figure 8-1. *Before using an adapter*

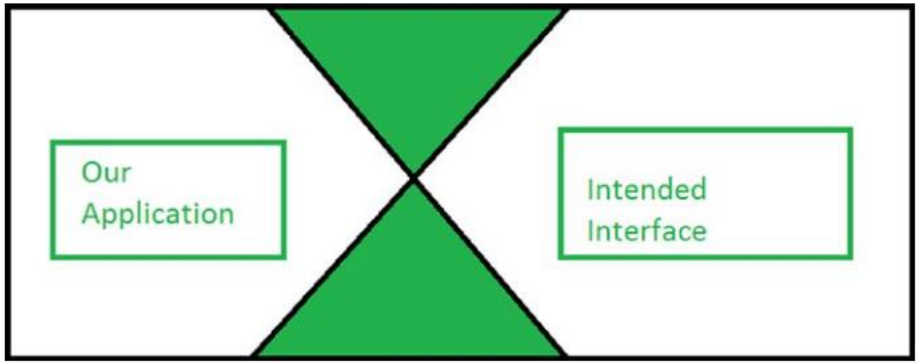


Figure 8-2. *After using an adapter*

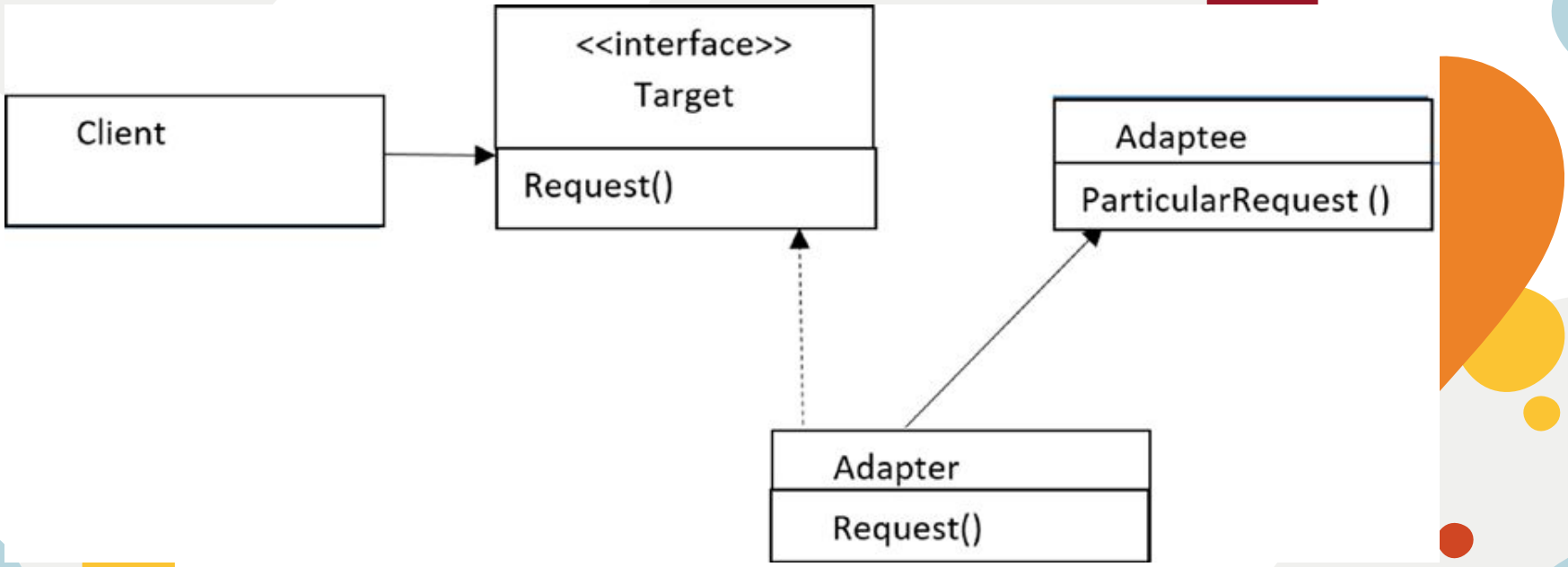
Adapter Example

Note In Java, you can consider the java.io.InputStreamReader class and the java.io.OutputStreamWriter class as examples of object adapters. They adapt an existing InputStream/OutputStream object to a Reader/Writer interface. You will learn about class adapters and object adapters shortly.

Tipe-tipe Adapter

- **Object Adapter**
  - Adapt through object compositions
- **Class Adapter**
  - Class adapters adapt through subclassing. They are the promoters of multiple inheritance. But you know that in Java, multiple inheritance through classes is not supported. (You need **interfaces** to implement the concept of multiple inheritance.)

Tipe-tipe Adapter - Object Adapter



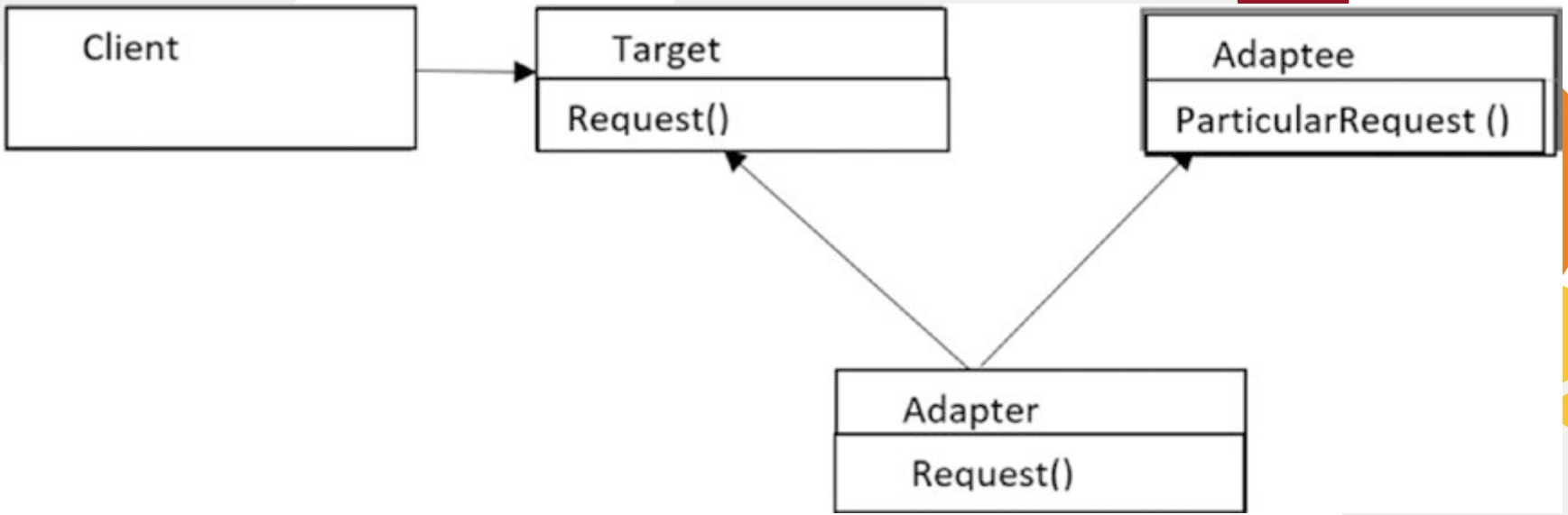
Tipe-tipe Adapter - Object Adapter

```
class Triangle implements TriInterface
{
    public double base;//base
    public double height;//height
    public Triangle(double base, double height)
    {
        this.base = base;
        this.height = height;
    }

    @Override
    public void aboutTriangle() {
        System.out.println("Triangle object with base: "+ this.base +" unit
        and height :" +this.height+ " unit.");

    }
```

Tipe-tipe Adapter - Class Adapter



Tipe-tipe Adapter - Class Adapter

```
class TriangleClassAdapter extends Triangle implements
RectInterface
{
    public TriangleClassAdapter(double base, double height) {
        super(base, height);
    }

    @Override
    public void aboutRectangle()
    {
        aboutTriangle();
    }
```

Let's Practice!

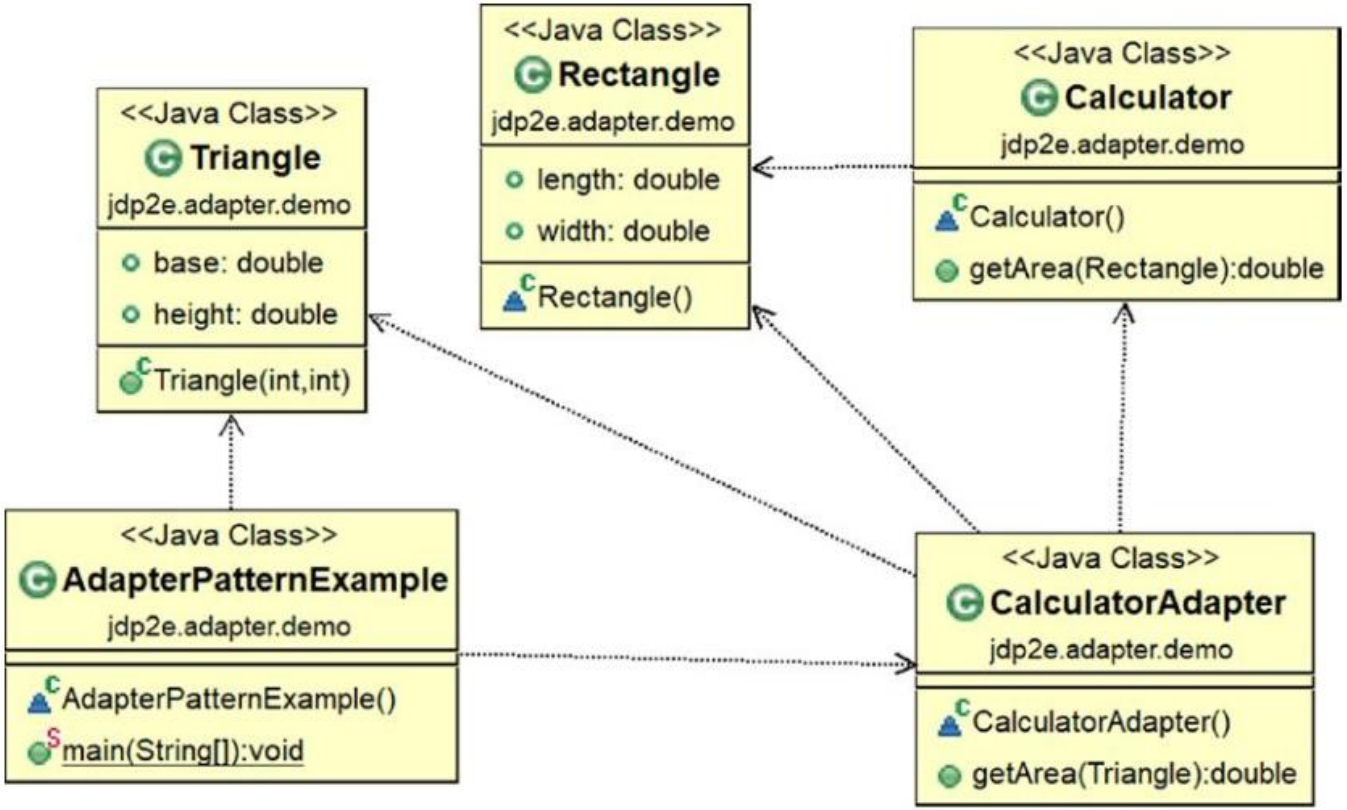
Adapter Example - Illustration

A simple use of this pattern is described in the following example.

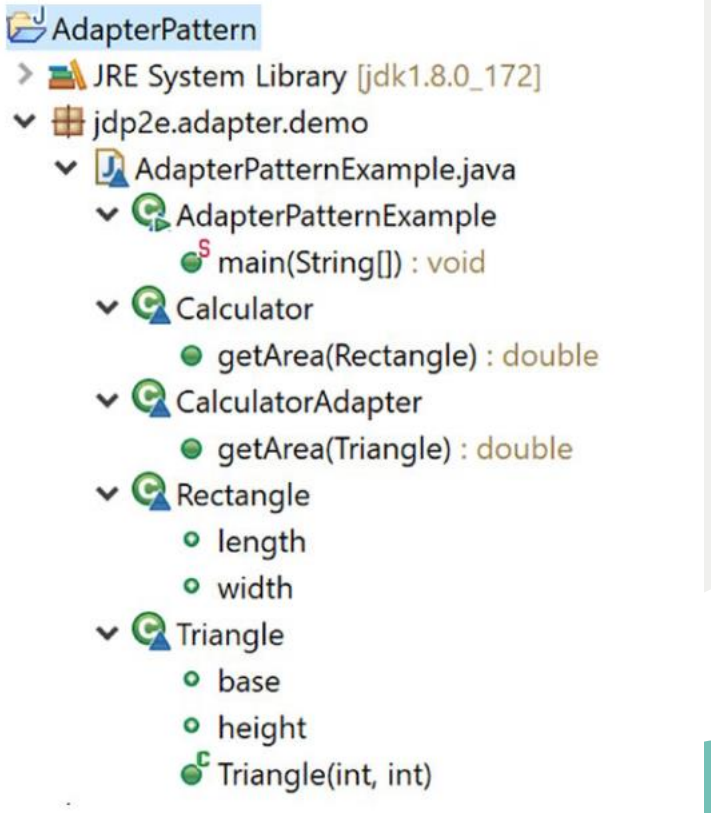
In this example, you can easily calculate the area of a rectangle. If you notice the Calculator class and its `getArea()` method, you understand that you need to supply a rectangle object in the `getArea()` method to calculate the area of the rectangle. Now suppose that you want to calculate the area of a triangle, but your constraint is that you want to get the area of it through the `getArea()`method of the Calculator class. So how can you achieve that?

To deal with this type of problem, I made CalculatorAdapter for the Triangle class and passed a triangle in its `getArea()` method. In turn, the method treats the triangle like a rectangle and calculates the area from the `getArea()` method of the Calculator class.

Class Diagram



Adapter Example (Package Explorer)



Silahkan Kerjakan
Tugas Praktikkum..