Broker 模式

采用 broker 模式对分布式计算进行简单模拟。系统在一个进程内模拟分布式环境，因此不涉及网络编程和进程间通信，Broker 通过本地函数调用的方式实现 request 和 response 的转发。

采用 broker 模式对分布式计算进行简单的模拟，要求如下：
设计四个 server，一个 server 接收两个整数，求和并返回结果，一个 server 接收两个整数，求差并返回结果，一个 server 接收两个整数，求积并返回结果，一个 server 接收两个整数，求商并返回结果。

客户端通过 ID 访问所需的服务，即：将服务 ID 和将两个整数发送给 Broker，由 broker 找到正确的服务器并将整数发送给相应的服务器，服务器计算结果，再将结果交给 broker 转发给客户，客户显示结果。

搭建 Broker 系统框架

按照 Broker 模式的要求，实现 Client，Broker，Server 三种组件(设计为三个 class)，不要求设计 Client-side Proxy 和 Server-side Proxy。必须实现如下功能：

- Server 可以注册到 Broker（使用 ID 号）
- Broker 为 Server 提供注册服务
- Broker 可将 client 的 request 转发到正确的 server
- Broker 可见 server 的 response 转发到 client

main 函数实现如下功能：

- 创建一个 broker 对象
- 创建两个 server 对象并注册到 broker
- 创建一个 client 对象
- 接收用户输入，由 Client 发起 request，并等待 response
- 输出 response

详细实现：

Broker.java

```
import java.util.Vector;
public class Broker
{

    // 记录 server 的 ID 号，从 0 递增
    private int serverID;

    // 将所有注册的 server 保存到 vector 容器内
    private Vector<Server> serverVector;
```

```
    public Broker()
    {
        serverID = 0;
        serverVector = new Vector<Server>();
    }

    // 为 server 提供注册 broker 的功能，为每个 server 设置一个从 0 递增的 ID
号
    public void Register(Server s)
    {
        serverVector.add(serverID, s);
        serverID++;
    }

    // broker 根据 server 的 ID 号获得 response，以便转发给客户端显示结果
    public int GetResponseByRequestId(int x, int y, int id)
    {
        Server s = (Server) serverVector.get(id);
        s.GetRequest(x, y);
        return s.SetResponse();
    }
}
```

Server.java

```
public class Server {

    private int firstNum;
    private int secondNum;
    private int serverID;
    private int rusult;

    // 每次新增 server，都要给他一个 ID
    public Server(int id)
    {
        this.serverID = id;
    }

   // server 获得 request，即获取传入的两个整数
    public void GetRequest(int num1, int num2)
    {
        this.firstNum = num1;
        this.secondNum = num2;
```

```
    }

    // 根据 ID 号调用对应的服务计算结果，得到 response
    public int SetResponse()
    {
        if (serverID == 0)
        {
            rusult = firstNum + secondNum;
              return rusult;
        }
        else if(serverID == 1)
        {
            rusult = firstNum - secondNum;
              return rusult;
        }
        else if(serverID == 2)
        {
            rusult = firstNum * secondNum;
              return rusult;
        }
        else
        {
            rusult = firstNum / secondNum;
              return rusult;
        }
    }

}
```

Client.java

```
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DecimalFormat;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
```

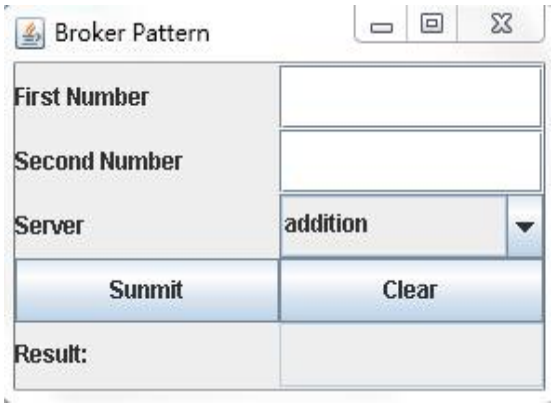
```
public class Client extends JFrame implements ActionListener
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private int firstNum;
    private int secondNum;
    private int serverID;
    private int rusult;

    private Broker broker;

    private JButton submitBtn;
    private JButton clearBtn;
    private JTextField firtInput;
    private JTextField secInput;
    private JTextField displayResult;

    // 提供加、减、乘、除四种服务供用户选择
    private String[] item = { "addition", "subtraction", "multiplication",
            "division" };
    private JComboBox<String> combobox = new JComboBox<String>(item);

    public Client(Broker b)
    {
        this.setTitle("Broker Pattern");

        broker = b;

        // 添加各种组件并注册相应的监听器
        setLayout(new GridLayout(5, 1));

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(1, 2));

        JPanel panel2 = new JPanel();
        panel2.setLayout(new GridLayout(1, 2));

        JLabel label1 = new JLabel("First Number");
        firtInput = new JTextField();
        JLabel label2 = new JLabel("Second Number");
        secInput = new JTextField();
```

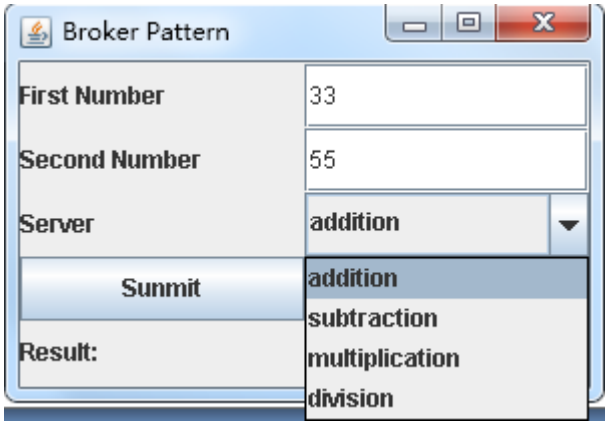
```
        panel.add(label1);
        panel.add(firtInput);

        panel2.add(label2);
        panel2.add(secInput);

        JPanel panel3 = new JPanel();
        panel3.setLayout(new GridLayout(1, 2));
        JLabel label3 = new JLabel("Server");

        panel3.add(label3);
        panel3.add(combobox);

        JPanel panel4 = new JPanel();
        panel4.setLayout(new GridLayout(1, 2));
        submitBtn = new JButton("Sunmit");
        clearBtn = new JButton("Clear");

        submitBtn.addActionListener(this);
        clearBtn.addActionListener(this);

        panel4.add(submitBtn);
        panel4.add(clearBtn);

        JPanel panel5 = new JPanel();
        panel5.setLayout(new GridLayout(1, 2));
        JLabel label5 = new JLabel("Result:");
        displayResult = new JTextField();
        displayResult.setEditable(false);

        panel5.add(label5);
        panel5.add(displayResult);

        add(panel);
        add(panel2);
        add(panel3);
        add(panel4);
        add(panel5);

    }

    // 实现监听事件处理程序
    public void actionPerformed(ActionEvent e)
```

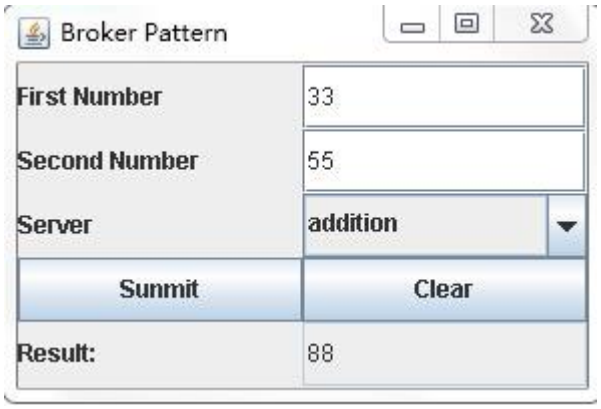
```
{
    if (e.getActionCommand() == "Sunmit")
    {
        try
        {
            int a = Integer.parseInt(firtInput.getText().trim());
            int b = Integer.parseInt(secInput.getText().trim());

            // 发送请求，包含两个整数以及服务 ID
            SetRequest(a, b, combobox.getSelectedIndex());

            // 获得由 broker 转发的 response 并显示到客户端
            int out = GetResponse();
            displayResult.setText(Integer.toString(out));

        } catch (Exception ex)
        {
            JOptionPane.showMessageDialog(null,
                    "Your input should be a type of integer!");
        }

    }
    else if (e.getActionCommand() == "Clear")
    { // 清空输入框
        firtInput.setText("");
        secInput.setText("");
        displayResult.setText("");

    }
}

// 添加 request，发送两个整数以及 server 的 ID 号
public void SetRequest(int num1, int num2, int id)
{
    this.firstNum = num1;
    this.secondNum = num2;
    this.serverID = id;
}

// 获取 broker 转发的 response，记录计算结果
public int GetResponse()
{
    rusult = broker.GetResponseByRequestId(firstNum, secondNum, serverID);
    return rusult;
```

```
    }

    public static void main(String[] args)
    {
        Broker b = new Broker();
        final Client client = new Client(b);

        // totalServer 记录提供的服务总数
        int totalServer = client.item.length;

        //  为每个 server 注册 broker，ID 号依次为 0 到 maxID

        for (int i = 0; i < totalServer; i++)
        {
            Server server = new Server(i);
            b.Register(server);
        }

        client.setVisible(true);
        client.setSize(300, 200);
        // client.pack(); //自动调节窗口大小

        client.setDefaultCloseOperation(EXIT_ON_CLOSE);

    }
}
```

运行效果：




Broker Pattern
First Number
33
Second Number
55
Server
addition
Sunmit
addition
subtraction
multiplication
division
Result:


Broker Pattern
First Number
33
Second Number
55
Server
addition
Sunmit
Clear
Result:
88