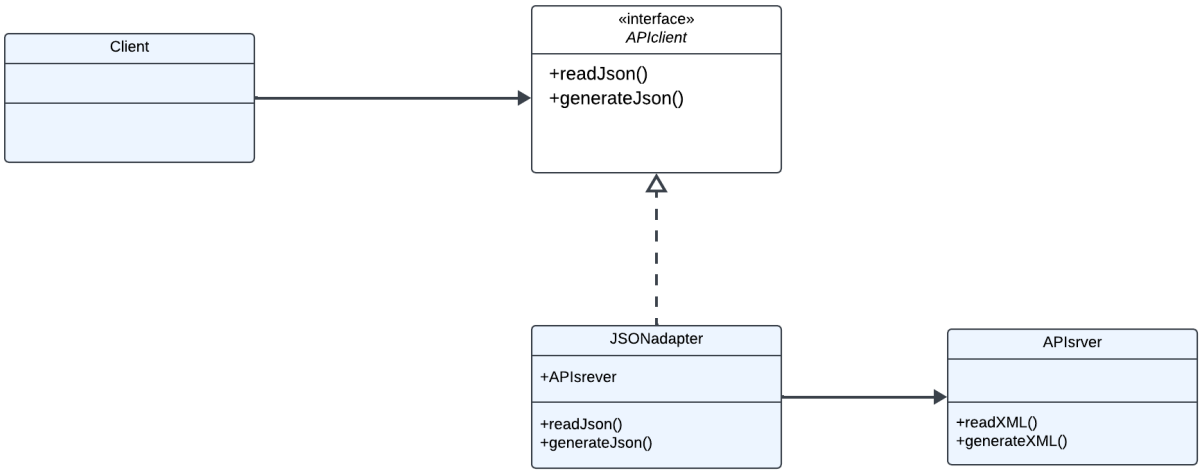
Design Patterns-3

Heba Saeed Hamdan202010678

-The 1’st direction “API client to API server ”using JSONadapter because the client read and generate content in JSON format:



- **APIclient “target”:**

```
public interface APIclient {
   public void readJson();
   public String generateJson();
 }
```

- **APIserver “Adaptee”:**

```
public class APIserver {
public void readXML(){
   System.out.println("data from server with XML format");
}
public String generateXML(){
return " Generating XML data from server";
}}
```

- **JSONadapter “Adapter”:**

```
public class JSONadapter implements APIclient {
APIserver xmlforamt;

   public JSONadapter(APIserver xmlformat) {
      this.xmlforamt=xmlformat;
   }

   @Override
   public void readJson() {
   xmlforamt.readXML();
   }
   @Override
   public String generateJson() {
   return xmlforamt.generateXML();
   }
}
```

- **Client class “test ”://Q3**

```
public class Main {
   public static void main(String[] args) {
    APIserver xmlformat=new APIserver();
    APIclient content=new JSONadapter(xmlformat);
      System.out.println(content.generateJson());
   }

}
```

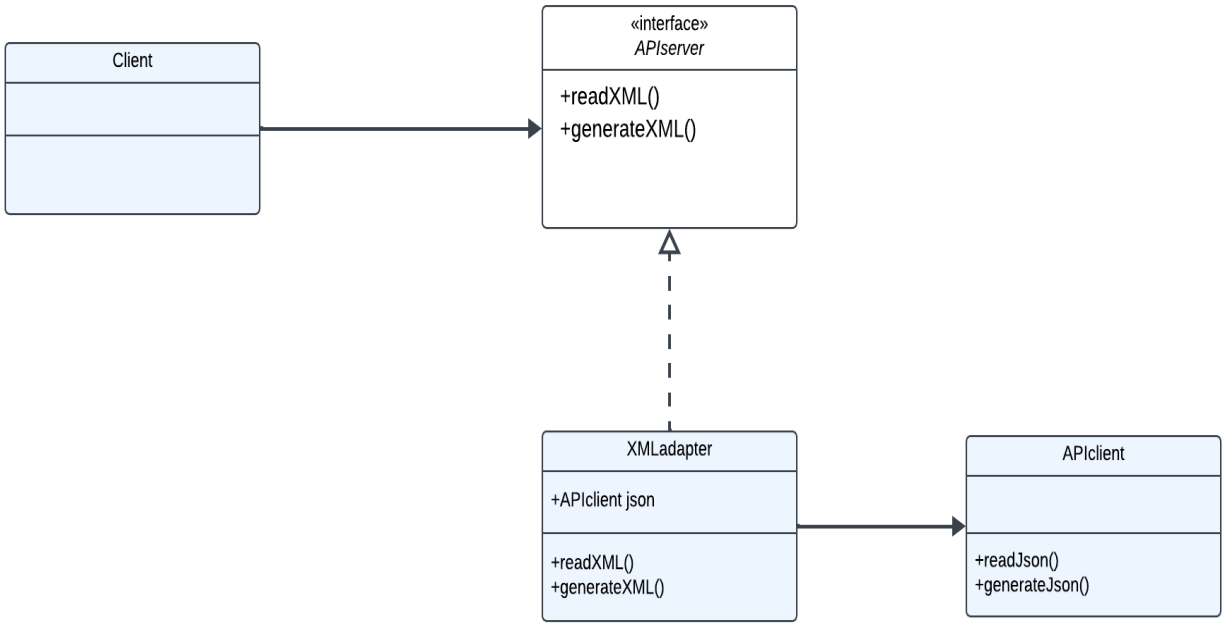
```java
public class Main {

    public static void main(String[] args) {
     APIserver xmlformat=new APIserver();
     APIclient content=new JSONadapter(xmlformat);
        System.out.println(content.generateJson());
    }

}
```

```
Output - assign3 (run)
run:
 Generating XML data from server
BUILD SUCCESSFUL (total time: 0 seconds)
```

-The second direction “API server to API client ”using XMLadapter because the server read and generate content in XML format:



- **APIserver “target”:**

```
public interface APIserver {
 public void readXML();
 public String generateXML();
```

```
    }
```

- **APIclient “Adaptee”:**

```
public class APIclient {
  public void readJson(){
     System.out.println("Data from client with JSON format");
   }
   public String generateJson(){
   return "Generating JSON data from client";
   }
}
```

- **XMLadapter “Adapter”:**

```
    public class XMLadapter implements APIserver{
    APIclient jsonformat;
public XMLadapter(APIclient jsonformat) {
        this.jsonformat = jsonformat;
      }

      @Override
      public void readXML() {
      jsonformat.readJson();
```

```
    }

    @Override
    public String generateXML() {
    return jsonformat.generateJson();
    }

    }
```

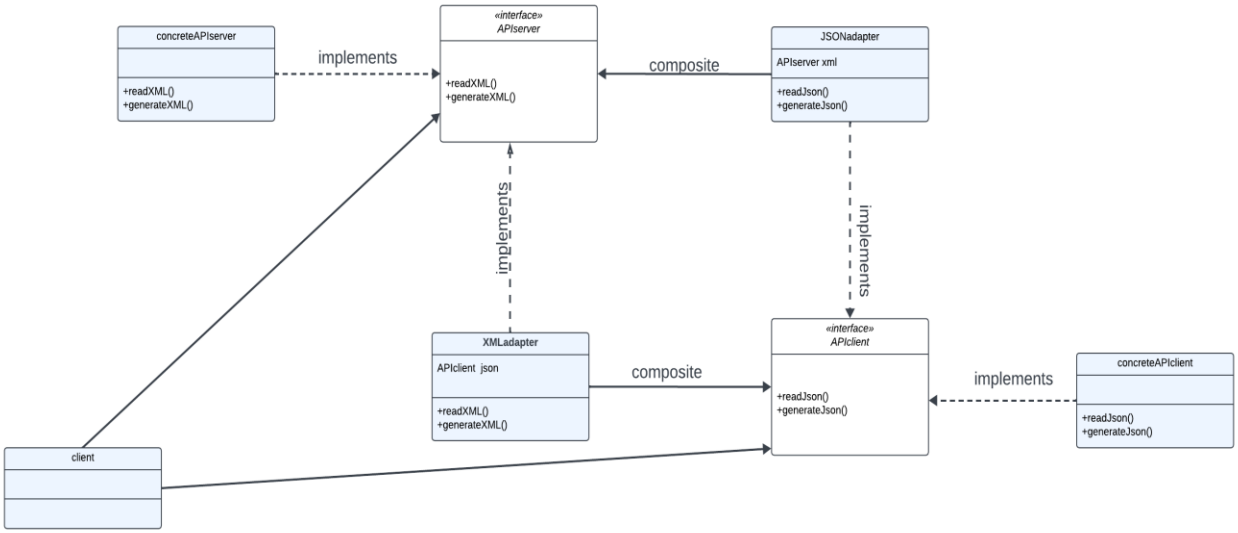
- **Client "Test"://Q3**

```
public class Main {
    public static void main(String[] args) {
      APIclient json=new APIclient();
      APIserver content=new XMLadapter(json);
      System.out.println(content.generateXML());
    }
}
```

```
/**
 *
 * @author Lenovo
 */
public class Main {
        public static void main(String[] args) {
          APIclient json=new APIclient();
          APIserver content=new XMLadapter(json);
          System.out.println(content.generateXML());
        }
}
```

```
Output
Debugger Console   assign3 (run)
run:
Generating JSON data from client
BUILD SUCCESSFUL (total time: 0 seconds)
```

Another solution that contain one main :



- **API client as “target” in case if get JSON format:**

```
public interface APIclient {
   public void readJson();
   public String generateJson();
}
```

- **concreteAPIclient jsust for implimintation of readJson and generateJson:**

```
public class APIclientconcrete implements APIclient{

    public void readJson(){
      System.out.println("Data from client with JSON format");
   }
   public String generateJson(){
   return "Generating JSON data from client";
   }
}
```

- **API server as “target” in case if get XML format:**

```
public interface APIserver {
   public void readXML();
   public String generateXML();
}
```

- **concreteAPIserver jsust for implimintation of readXML and generateXML:**

```
public class APIserverconcrete implements APIserver {
 public void readXML(){
    System.out.println("Data from server with XML format");
}
public String generateXML(){
return " Generating XML data from server";
}
}
```

- **adapter class:**

```
public class JSONadapter implements APIclient{
   APIserver xml;
   public JSONadapter(APIserver xml) {
      this.xml = xml;
   }
   @Override
   public void readJson(){
   xml.readXML();
   }
   @Override
   public String generateJson(){
   return xml.generateXML();
   }
}
```

- **adapter class:**

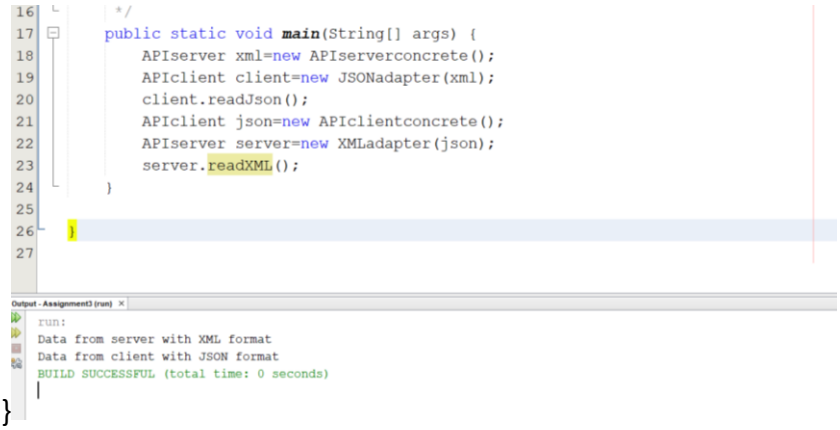
```
public class XMLadapter implements APIserver {
   APIclient json;

   public XMLadapter(APIclient json) {
      this.json = json;
   }
   @Override
   public void readXML(){
   json.readJson();
```

```
    }
    @Override
    public String generateXML(){
    return json.generateJson();
    }
}
```

- **Test drive:**

```
public static void main(String[] args) {
    APIserver xml=new APIserverconcrete();
    APIclient client=new JSONadapter(xml);
    client.readJson();
    APIclient json=new APIclientconcrete();
    APIserver server=new XMLadapter(json);
    server.readXML();}
```



}