在前面的练习中，当web项目中有两个或两个以上的Servlet实现类时，而且每一个Servlet接口的实现类都需要实现Servlet接口中的所有方法。

但是大部分时候仅需要将需求代码写到service方法中，其他方法一直是空着的，这样看着很别扭，所以我们是否可以就创建一个抽象的实现Servlet接口的类，叫适配器类Adapter。

当我们的Servlet对象只需要使用到service方法而其他方法不需要时，

该适配器类就将service方法定义成抽象方法， 而其他方法在该适配器类中进行实现，

然后我们的项目类只需要继承这个适配器类，将其中的抽象方法实现，其他方法不需要就不用重写。代码页面简洁，实用，而且还可以在适配器类中添加其他方法。

这样的设计模式叫做适配器模式

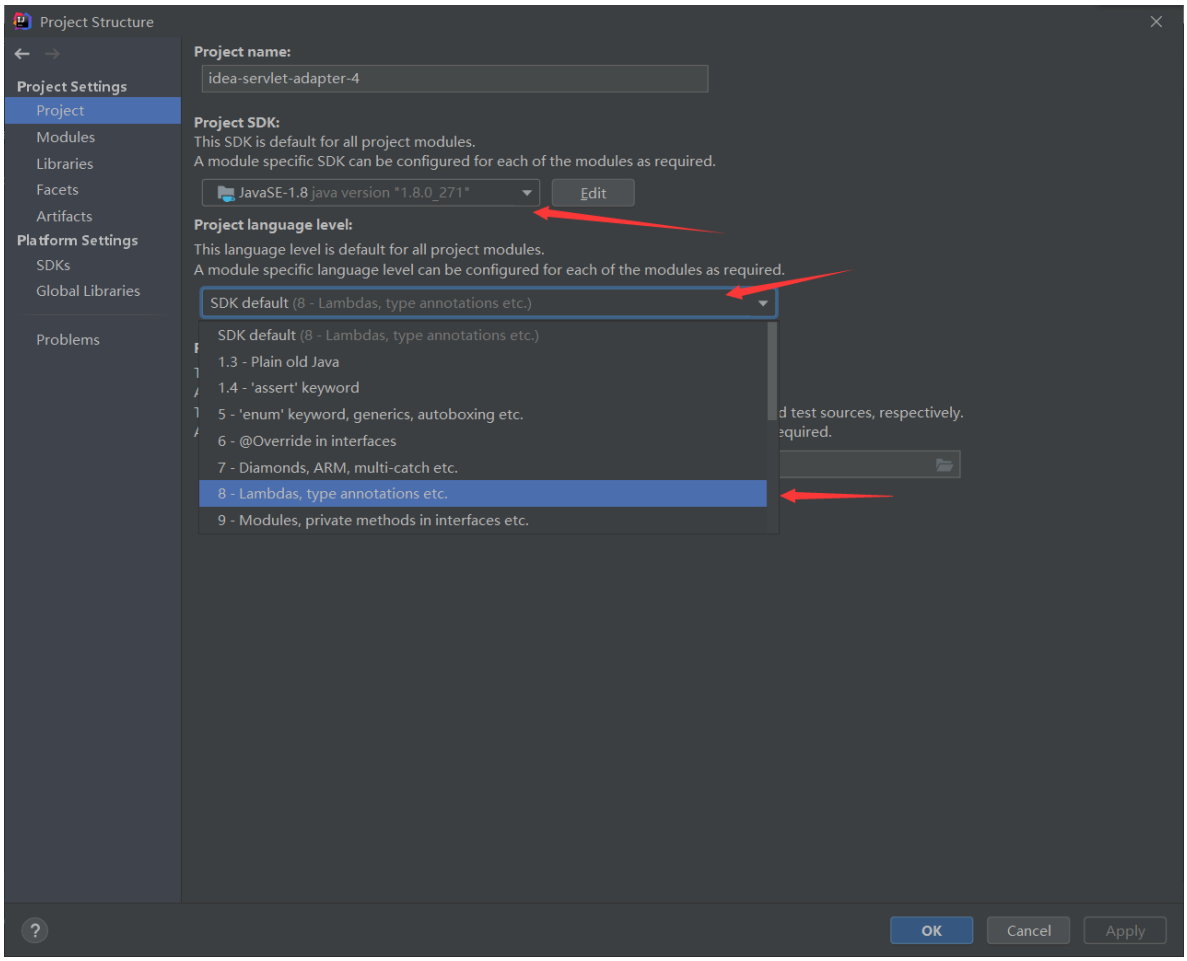
设计模式的分类：

```
--创建型：解决对象的创建问题
--行为型：该模式与方法、行为、算法有关的设计模式
--结构型：更多类，更多的对象组合成更大的结构解决某个特定的问题

--Gof95（1995年，四人组提出的23种设计模式）
    *单例模式
    *工程模式
    *适配器模式
    *迭代模式【集合】
    *策略模式【集合】
    *装饰器模式【IO流中】
    ...
--JavaEE模式
    ...
```

下面我们创建一个web项目，创建一个抽象适配器类GenericServlet--顾名思义：通用的Servlet。

但是这里有几个错误需要关注，因为本电脑上安装了JDK8和JDK11，在创建Java普通项目和创建一个Java空工程然后再在该工程中添加模块其中有些配置会有区别，但是都一定要注意Project SDK和Project language level版本要一致，如下图所示，1.8就要对应8



GenericServlet适配器类代码如下

```
package com.javaweb.servlet;

import javax.servlet.*;
import java.io.IOException;

/**
 * GenericServlet是一个适配器类，这个适配器是一个Servlet
 * 以后程序员无需实现Servlet接口，只需要继承GenericServlet抽象类，单单重写service方法就行了
 */

public abstract class GenericServlet implements Servlet {

    private ServletConfig config;

    @Override
    //有时候我们希望在服务器启动阶段执行一段特殊的代码，如果我们在子类中重写init方法将特殊的代码写进去，那么里面
```

```
    //的赋值语句就没法执行了，config是私有的就没法得到成员的ServletConfig对象，然而把定义代码和赋值代码搬到子类中
    //显得有些多余，就不像适配器模式了，那么我们可以在适配器类中定义一个init的无参数方法，在适配器中的init有参数方法
    //中去调用，在子类中覆盖init无参方法，将特殊程序写到重写的无参的init方法中
    public void init(ServletConfig servletConfig) throws ServletException {
        this.config = servletConfig;
        this.init();
    }

    //定义一个无参的init方法，以防服务器启动阶段有特殊需求执行代码

    /**
     * 在初始化时刻需要执行一段特殊的程序，在子类中重写下面无参数的init方法
     */
    public void init(){}

    @Override
    public ServletConfig getServletConfig() {
        return config;
    }

    public abstract void service(ServletRequest servletRequest, ServletResponse servletResponse) throws ServletException, IOException;

    @Override
    //该方法是返回一段信息，该信息包含项目的作者、版本以及版权等
    public String getServletInfo() {
        return null;
    }

    @Override
    public void destroy() {

    }

        //-------------------------以下所有方法是扩展方法--------------------------

    public ServletContext getServletContext(){
        return config.getServletContext();
    }

    //.......
}
```

子类如下

```
package com.javaweb.servlet;

import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class HelloServlet extends GenericServlet{
```

```
    //子类继承父类，将父类中的方法和变量都继承了，若没有重写父类的方法，子类对象调用某个方法时是执行父类中的
    //方法代码，这可以在父类的方法中添加断点测试，若重写了父类的某个方法，子类对象调用该方法时执行的是重写后的
    //方法代码，此时若想访问父类中的该方法，可以在重写的方法中用super.方法名的方式调用父类中的该方法。super表示
    //子类对象中的父类型特征，不是一个父类型对象的引用

    @Override
    public void init() {
        System.out.println("Hello init...");
    }

    @Override
    public void service(ServletRequest servletRequest, ServletResponse servletResponse) throws ServletException, IOException {

        servletResponse.setContentType("text/html;charset=UFT-8");
        PrintWriter out = servletResponse.getWriter();
        out.print("hello servlet!");

    }
}
```

SUN公司已经写好了一个适配器抽象类也叫GenericServlet，完整类名：javax.servlet.GenericServlet。我们不用写，以后编程直接继承该类就可以了，并且SUN公司写的init方法解决法案与上面的例子一直，并且该适配器类还扩展了许多有用的方法。