第六次作业

第六章

**第一题**

**创建三个包：p1、p2和p3**

**在p1包中实现一个类A1及其子类A2**

**在p2包中实现一个类B1及其子类B2**

**在p3包实现一个类Test，在该类中实现main方法**

**要求：**

**分别在A1、A2、B1及B2中创建不同访问限制的属性和方法**

**在main方法中实例化各个类，尝试访问不同访问限制的属性和方法，观察并理解出错信息（以及理解不出错的原因）**

**把test类放在不同的包里再试试看**

**把出错信息截图并解释出错原因**

文件结构如图1-1：

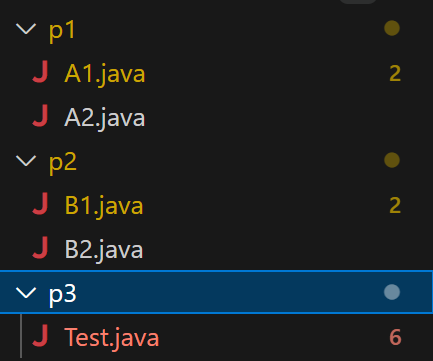


图1-1 第一题文件结构

A1实现如下：

package p1;

public class A1 {

    public int publicVar = 10;

    protected int protectedVar = 20;

    int defaultVar = 30;  // 默认访问权限

    private int privateVar = 40;

    public void publicMethod() {

        System.out.println("This is a public method in A1");

    }

    protected void protectedMethod() {

        System.out.println("This is a protected method in A1");

    }

    void defaultMethod() {

        System.out.println("This is a default method in A1");

    }

    private void privateMethod() {

        System.out.println("This is a private method in A1");

    }

}

A2 实现如下：

package p1;

public class A2 extends A1{

    // A2 继承自 A1，可以访问 A1 中的 protected 和 public 成员

    public void accessParentMembers() {

        System.out.println(publicVar);    // 可以访问 public 成员

        System.out.println(protectedVar); // 可以访问 protected 成员

        // System.out.println(defaultVar); // default 成员在不同包内不能直接访问

        // System.out.println(privateVar); // private 成员只能在其定义的类内访问

        publicMethod();     // 可以调用 public 方法

        protectedMethod();  // 可以调用 protected 方法

        // defaultMethod();  // default 方法在不同包内不能直接访问

        // privateMethod();  // private 方法只能在其定义的类内访问

    }

}

B1：

package p2;

public class B1 {

    public int publicVar = 50;

    protected int protectedVar = 60;

    int defaultVar = 70;  // 默认访问权限

    private int privateVar = 80;

    public void publicMethod() {

        System.out.println("This is a public method in B1");

    }

    protected void protectedMethod() {

        System.out.println("This is a protected method in B1");

    }

    void defaultMethod() {

        System.out.println("This is a default method in B1");

    }

    private void privateMethod() {

        System.out.println("This is a private method in B1");

    }

}

B2：

package p2;

public class B2 extends B1{

    public void accessParentMembers() {

        System.out.println(publicVar);    // 可以访问 public 成员

        System.out.println(protectedVar); // 可以访问 protected 成员

        // System.out.println(defaultVar); // default 成员在不同包内不能直接访问

        // System.out.println(privateVar); // private 成员只能在其定义的类内访问

        publicMethod();     // 可以调用 public 方法

        protectedMethod();  // 可以调用 protected 方法

        // defaultMethod();  // default 方法在不同包内不能直接访问

        // privateMethod();  // private 方法只能在其定义的类内访问

    }

}

Test：

package p3;

import p1.A1;

import p1.A2;

import p2.B1;

import p2.B2;

public class Test {

    public static void main(String[] args) {

        // 创建 A1 对象

        A1 a1 = new A1();

        System.out.println(a1.publicVar);  // 可以访问 public 成员

        System.out.println(a1.protectedVar); // 在不同包内，无法访问 protected 成员

        // System.out.println(a1.defaultVar); // 在不同包内，无法访问 default 成员

        // System.out.println(a1.privateVar); // private 成员只能在其定义的类内访问

        a1.publicMethod();     // 可以调用 public 方法

        // a1.protectedMethod();  // 在不同包内，无法调用 protected 方法

        // a1.defaultMethod();  // 在不同包内，无法调用 default 方法

        // a1.privateMethod();  // private 方法只能在其定义的类内调用

        // 创建 A2 对象

        A2 a2 = new A2();

        System.out.println(a2.publicVar);  // 可以访问 public 成员

        System.out.println(a2.protectedVar); // 可以访问 protected 成员，因为 A2 是 A1 的子类

        // System.out.println(a2.defaultVar); // 在不同包内，无法访问 default 成员

        // System.out.println(a2.privateVar); // private 成员只能在其定义的类内访问

        a2.publicMethod();     // 可以调用 public 方法

        a2.protectedMethod();  // 可以调用 protected 方法，因为 A2 是 A1 的子类

        // a2.defaultMethod();  // 在不同包内，无法调用 default 方法

        // a2.privateMethod();  // private 方法只能在其定义的类内调用

        // 创建 B1 对象

        B1 b1 = new B1();

        System.out.println(b1.publicVar);  // 可以访问 public 成员

        System.out.println(b1.protectedVar); // 在不同包内，无法访问 protected 成员

        // System.out.println(b1.defaultVar); // 在不同包内，无法访问 default 成员

        // System.out.println(b1.privateVar); // private 成员只能在其定义的类内访问

        b1.publicMethod();     // 可以调用 public 方法

        // b1.protectedMethod();  // 在不同包内，无法调用 protected 方法

        // b1.defaultMethod();  // 在不同包内，无法调用 default 方法

        // b1.privateMethod();  // private 方法只能在其定义的类内调用

        // 创建 B2 对象

        B2 b2 = new B2();

        System.out.println(b2.publicVar);  // 可以访问 public 成员

        System.out.println(b2.protectedVar); // 可以访问 protected 成员，因为 B2 是 B1 的子类

        // System.out.println(b2.defaultVar); // 在不同包内，无法访问 default 成员

        // System.out.println(b2.privateVar); // private 成员只能在其定义的类内访问

        b2.publicMethod();     // 可以调用 public 方法

        b2.protectedMethod();  // 可以调用 protected 方法，因为 B2 是 B1 的子类

        // b2.defaultMethod();  // 在不同包内，无法调用 default 方法

        // b2.privateMethod();  // private 方法只能在其定义的类内调用

    }

}

此时出错信息如图1-2：

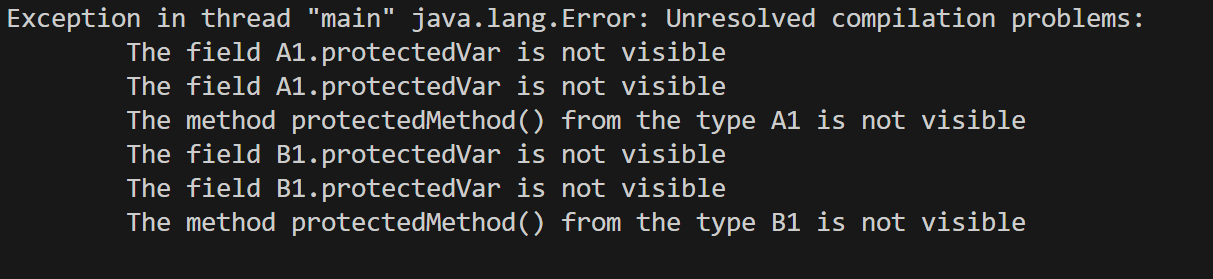


图1-2 Test 出错信息

分析出错原因：

在不同包内无法访问protected成员。

在不同包内无法访问default成员。

在不同包内也无法访问 protected 和 default 方法。

Private方法只能在其定义的类中调用。

**第二题**

**创建一个接口Print，在其中定义一个打印方法print()。**

**创建Person接口(即“人”)，它有setData()和getData()方法对“人”属性name、sex和birthday赋值和获得这些属性组成的字符串信息。**

**创建类Student实现Person接口和Print接口，并对自己的“学生”属性的成员变量sID、speciality设置值和获得它们值所组成的字符串信息，并用实现的Print接口中的打印输出这些信息。**

**在任意类中实现main方法，要求在该方法中创建类的实例并调用各方法，运行程序并得到计算结果**

本题文件布局如图2-1：

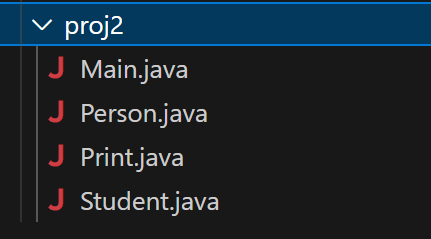


图2-1 第二题文件

其中，Print.java内容如下：

package hw6.proj2;

public interface Print {

    void print();

}

Student.java内容如下：

package hw6.proj2;

public class Student implements Person, Print{

    // Person 接口的成员变量

    private String name;

    private String sex;

    private String birthday;

    // Student 类特有的成员变量

    private String sID;  // 学号

    private String speciality;  // 专业

    // 实现 Person 接口的方法

    @Override

    public void setData(String name, String sex, String birthday) {

        this.name = name;

        this.sex = sex;

        this.birthday = birthday;

    }

    @Override

    public String getData() {

        return "Name: " + name + ", Sex: " + sex + ", Birthday: " + birthday;

    }

    // Student 类特有的方法，设置学生的学号和专业

    public void setStudentData(String sID, String speciality) {

        this.sID = sID;

        this.speciality = speciality;

    }

    // 获取学生的学号和专业信息

    public String getStudentData() {

        return "Student ID: " + sID + ", Speciality: " + speciality;

    }

    // 实现 Print 接口的 print() 方法，打印所有信息

    public void print() {

        System.out.println(getData());

        System.out.println(getStudentData());

    }

}

Person.java内容如下：

package hw6.proj2;

public interface Person {

    void setData(String name, String sex, String birthday);

    String getData();

}

Main.java内容如下：

package hw6.proj2;

public class Main {

    public static void main(String[] args) {

        // 创建 Student 实例

        Student student = new Student();

        // 调用 setData() 设置学生的个人信息

        student.setData("Alice", "Female", "1999-08-12");

        // 调用 setStudentData() 设置学生的学号和专业

        student.setStudentData("20240001", "Computer Science");

        // 打印学生的所有信息

        student.print();

    }

}

运行截图如图2-2：

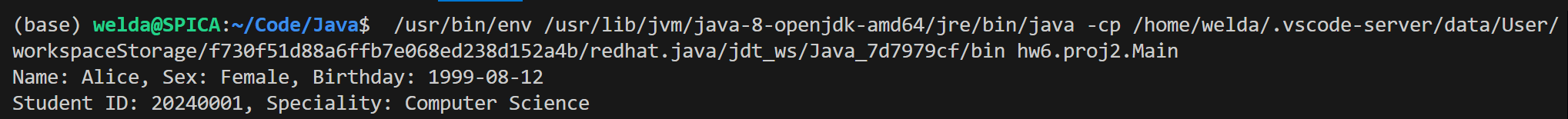


图2-2 第二题运行结果截图

**第三题**

**任意写一个类，要求：**

**其中要有成员内部类，该内部类要有自己的方法。该方法可以是简单的打印输出一段信息。**

**在外部类实现main方法，在该方法中创建该外部类的实例，然后创建内部类的实例。**

**最终在main方法中调用内部类的方法，得到运行结果**

**——————————————**

**补充：感兴趣的同学可以尝试一下匿名类。可以自行定义一个简单接口。然后通过实现该接口从而创建该匿名类的实例，然后调用该匿名类中实现的接口定义方法**

代码：

package hw6.proj3;

public class OuterClass {

    // 外部类的属性

    private String outerName = "OuterClass";

    // 成员内部类

    public class InnerClass {

        // 内部类的方法，打印输出信息

        public void display() {

            System.out.println("This is the InnerClass. Accessing outer class property: " + outerName);

        }

    }

    // 外部类的 main 方法

    public static void main(String[] args) {

        // 创建外部类的实例

        OuterClass outer = new OuterClass();

        // 通过外部类的实例创建内部类的实例

        OuterClass.InnerClass inner = outer.new InnerClass();

        // 调用内部类的方法

        inner.display();

    }

}

运行结果如图3-1：

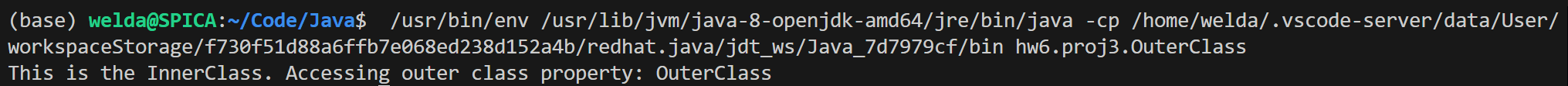


图3-1 第三题运行结果

补充代码：

package hw6.proj3;

interface SimpleInterface {

    void doSomething();

}

public class AnonymousClassExample {

    public static void main(String[] args) {

        SimpleInterface instance = new SimpleInterface() {

            public void doSomething() {

                System.out.println("Anonymous class, ohhhhhhhh!");

            }

        };

        instance.doSomething();

    }

}

运行结果如图3-2所示：

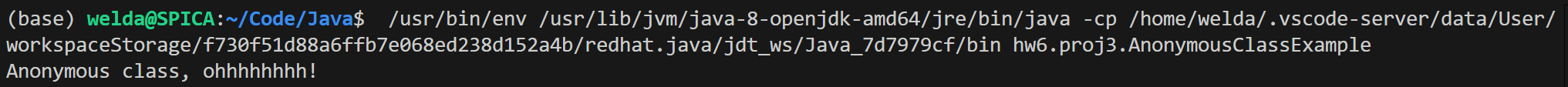


图3-2 补充内容运行截图