به نام خدا

آزمایش سوم آزمایشگاه مهندسی نرم افزار

کاربرد عملی اصول شئ گرایی SOLID با استفاده از روش Test Driven Development

علی‌رضا دیزجی ۹۶۱۰۵۷۴۵

حمیدرضا هدایتی ۹۶۱۰۹۹۳۹

دانشگاه صنعتی شریف

پاییز ۱۴۰۰

۱- پیش‌فرض‌های پروژه:

در این آزمایش از Intellij IDE و Maven Package Manager استفاده شده و برای پیاده‌سازی متدهای تست از کتاب‌خانه‌های Junit همانطور که در صورت آزمایش اشاره شده استفاده شده است.

۲- محاسبه مساحت مستطیل:

- **پیاده‌سازی تست‌ها:**

تست‌های زیر برای محاسبه مساحت مستطیل افزوده شده است و کد آن پیاده‌سازی شده است. که متشکل از سه نوع مستطیل معروف و یک مستطیل غیر معتبر با طول و عرض منفی است.

- **اجرای تست‌ها:**

با توجه به اینکه class مستطیل همچنان پیاده‌سازی نشده بود به کامپایل ارور cannot find symbol برخوردیم که در نتیجه به دنبال تعریف کلاس‌ها رفتیم.

- **برطرف کردن مشکلات کامپایل:**

با پیاده‌سازی کلاس پدر shape و اینترفیس areaCalculation و در نهایت class اصلی Rectangle مشکلات زمان کامپایل تست‌ها را حل کردیم.

- **اجرای تست‌ها:**

با پیاده‌سازی کلاس‌ها و اجرای دوباره تست‌ها به مشکل نوشته نشدن بدنه تابع اصلی برخوردیم و در اولین تست fail شدیم:

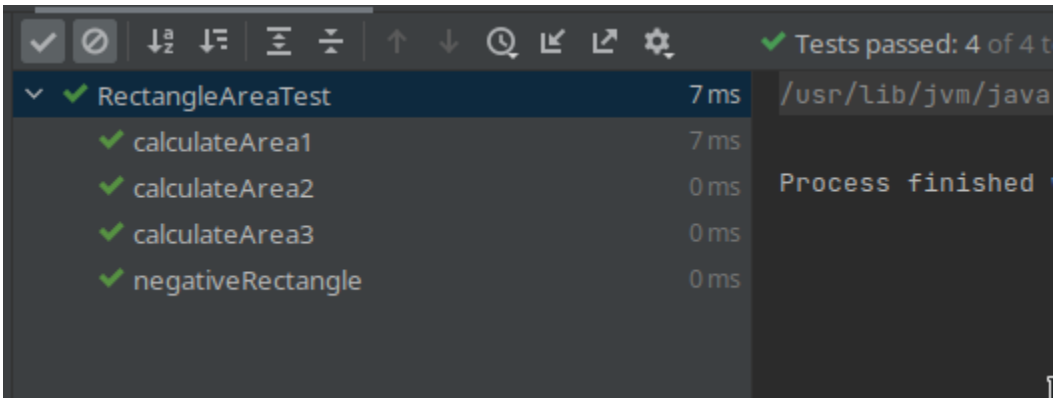
- **برطرف کردن مشکلات زمان اجرا:**

با پیاده‌سازی تابع calculateArea و همچنین چک کردن مثبت بودن طول و عرض مشکلات زمان اجرا را تلاش کردیم حل کنیم:

```java
public class Rectangle extends Shape implements areaCalculation {
    private int width, height;
    public Rectangle(int width, int height) throws Exception{
        if (width <= 0 || height <= 0)
            throw new Exception("Negative Rectangle");
        this.width = width;
        this.height = height;
    }

    @Override
    public int calculateArea() {
        return this.width * this.height;
    }
}
```

- **اجرای تست‌ها:**

با اجرای دوباره تست‌ها به نتیجه قابل قبول زیر رسیدیم:



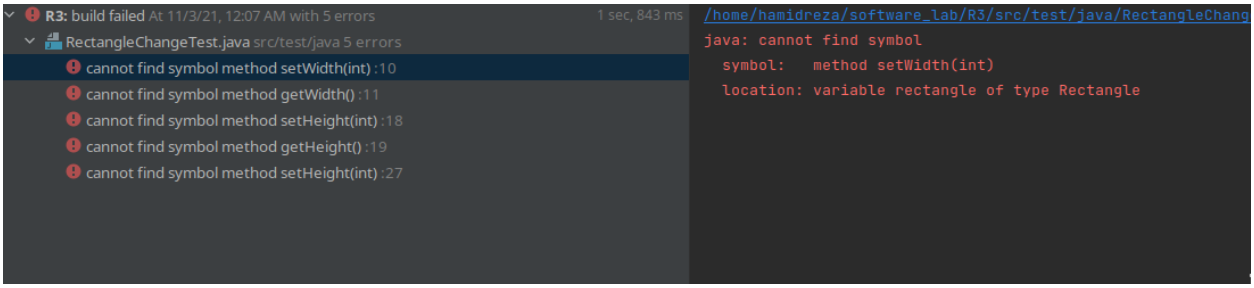
۳- تغییر طول و عرض مستطیل:

- **پیاده‌سازی تست‌ها:**

برای این بخش نخست تست‌های مربوطه برای set کردن طول و عرض و سپس get کردن آن را پیاده‌سازی کردیم که یکی از آن‌ها set کردن طول و یکی دیگر set کردن عرض و یکی دیگر set کردن غیرقابل قبول (منفی) این دو است.

- **اجرای تست‌ها:**

همانطور که انتظار می‌رفت به دلیل نوشته نشدن توابع set و get این بخش هم با کامپایل ارور مواجه شد:



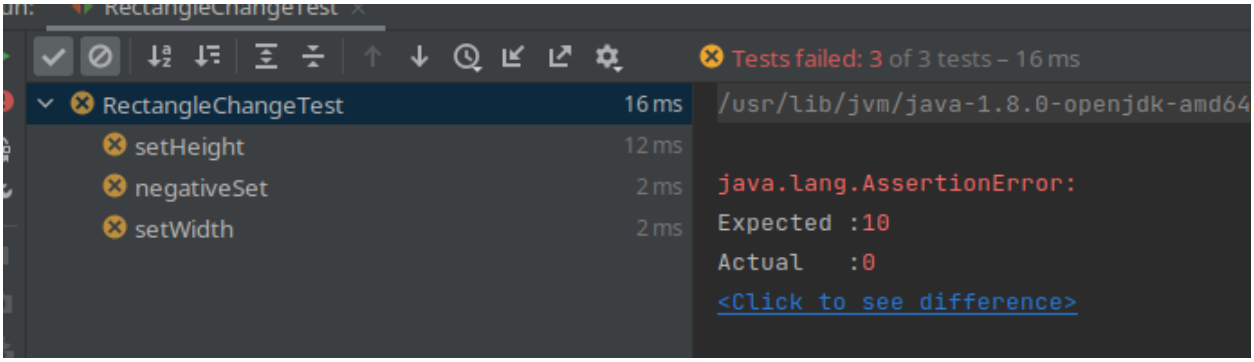
- **برطرف کردن مشکلات کامپایل:**

با نوشته شدن setter و getter ها برای width و height تلاش به حل مشکل کردیم:

```
    public void setHeight(int height) throws Exception{

    }

    public int getHeight() {
        return 0;
    }

    public void setWidth(int width) throws Exception{

    }

    public int getWidth() {
        return 0;
    }
}
```

- **اجرای تست‌ها:**

با پیاده‌سازی قالب توابع مورد نیاز همانطور که انتظار میرفت مشکل کامپایل ارور حل شد ولی همچنان تست‌ها fail می‌شدند.



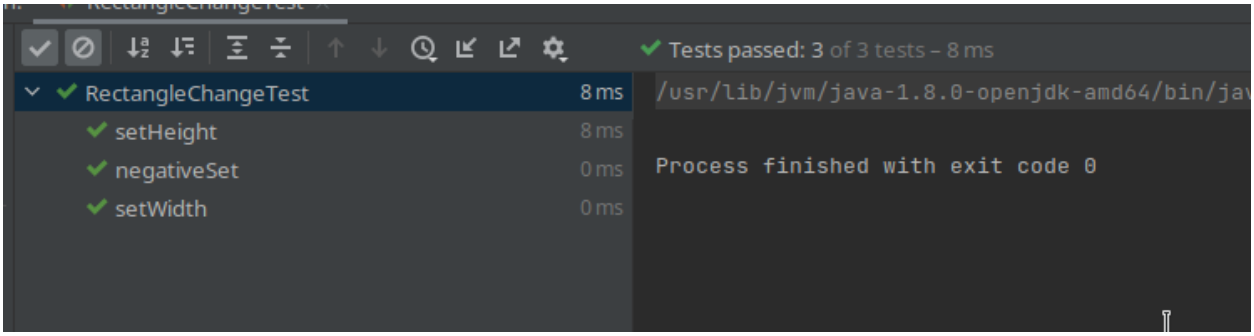
- **برطرف کردن مشکلات زمان اجرا:**

با پیاده‌سازی بدنه توابع تلاش کردیم که تست‌ها را pass کنیم:

```
    public void setHeight(int height) throws Exception{
        if (height <= 0)
            throw new Exception("Negative Rectangle");
        this.height = height;
    }

    public int getHeight() {
        return height;
    }

    public void setWidth(int width) throws Exception{
        if (width <= 0)
            throw new Exception("Negative Rectangle");
        this.width = width;
    }

    public int getWidth() {
        return width;
    }
}
```

- **اجرای تست‌ها:**

و این بار با اجرای دوباره تست‌ها توانستیم تست‌ها را با موفقیت pass کنیم:



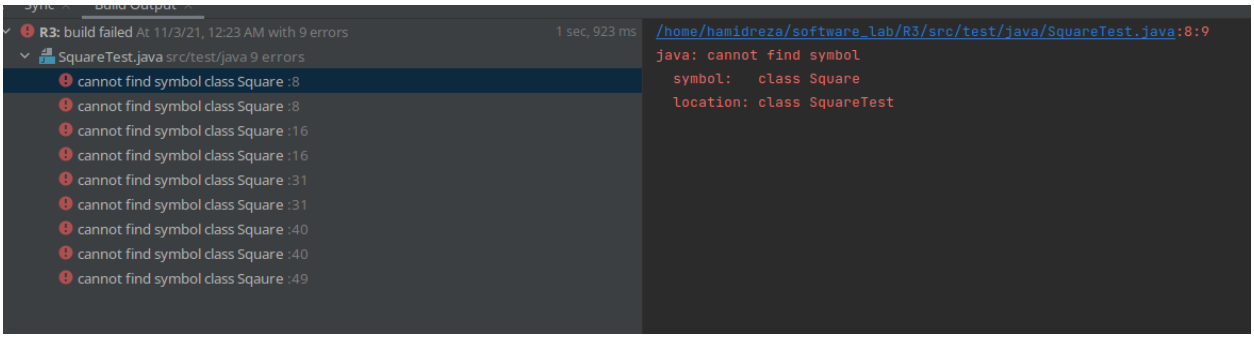
۴- ساپورت کردن شکل مربع:

- **پیاده‌سازی تست‌ها:**

مانند مستطیل دو تست مختلف valid برای محاسبه مساحت و یک تست با طول منفی و همچنین یک تست برای set کردن طول مربع و یک تست برای set کردن invalid و سرجمع ۵ تست مختلف پیاده‌سازی شد.

- **اجرای تست‌ها:**

به دلیل پیاده‌سازی نشدن class مربع در زمان کامپایل مشکل خوردیم:



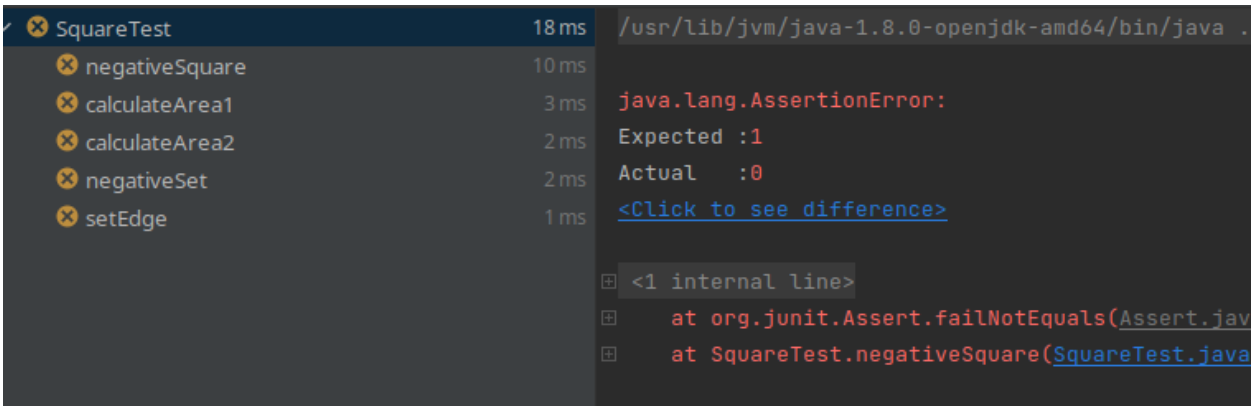
- **برطرف کردن مشکلات کامپایل:**

با پیاده‌سازی مربع مانند مستطیل و ارثبری از shape و پیاده‌سازی areaCalculation تلاش کردیم که مشکلات کامپایل را حل کنیم:

```java
public class Square extends Shape implements areaCalculation {
    private int edge;
    public Square(int edge) throws Exception{
    }

    @Override
    public int calculateArea() {
        return 0;
    }

    public void setEdge(int edge) throws Exception{
    }

    public int getEdge() {
        return 0;
    }
}
```

- **اجرای تست‌ها:**

مشکلات کامپایل حل شد ولی نتایج تست‌ها مورد انتظار ما نبود:



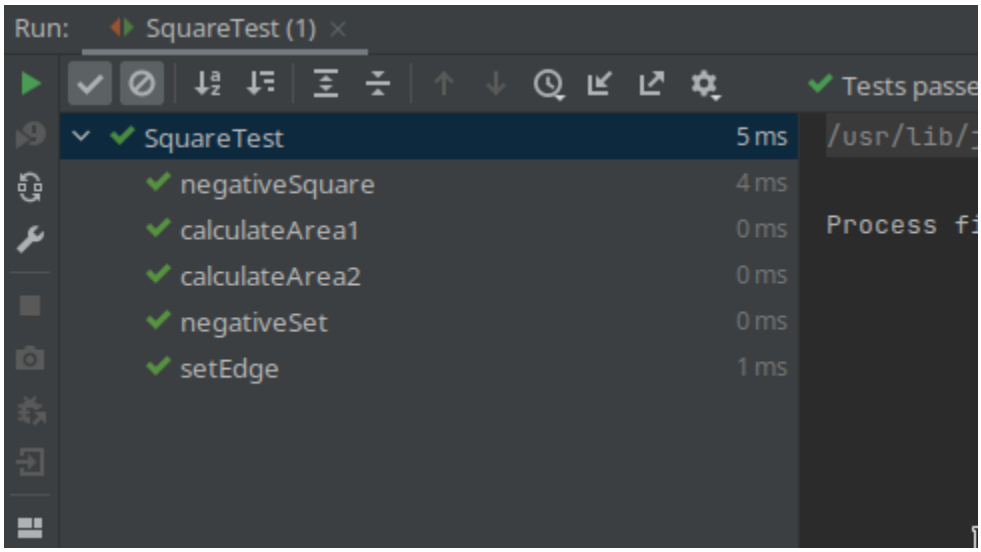
- **برطرف کردن مشکلات زمان اجرا:**

با پیاده‌سازی توابع مورد نیاز تلاش به حل مشکلات کردیم:

```java
public class Square extends Shape implements areaCalculation {
    private int edge;
    public Square(int edge) throws Exception{
        if (edge <= 0)
            throw new Exception("Negative Square");
        this.edge = edge;
    }

    @Override
    public int calculateArea() {
        return this.edge * this.edge;
    }

    public void setEdge(int edge) throws Exception{
        if (edge <= 0)
            throw new Exception("Negative Square");
        this.edge = edge;
    }

    public int getEdge() {
        return this.edge;
    }
}
```

- **اجرای تست‌ها:**

طبق انتظارمان توانستیم تست‌ها را pass کنیم و برنامه را به پایان برسانیم:



۵- جزئیات نحوه پیاده‌سازی:

در این پروژه کلاس مربع و مستطیل از کلاس Shape ارث‌بری می‌کنند و اینترفیس areaCalculation را پیاده‌سازی می‌کنند لذا می‌توانیم قواعد SOLID را در این پروژه بررسی کنیم:

- **DIP:**

از آن جا که ارث‌بری تا حد ممکن رخ داده و abstraction وجود دارد ولی در این پروژه نیاز مبرمی به وجود این اصل وجود نداشت.

- **ISP:**

مربع و مستطیل از Shape به صورت جداگانه ارث‌بری می‌کنند که یک رابط چندمنظوره حساب می‌شود.

- **LSP:**

از یک کلاس والد Shape استفاده شده است.

- **OCP:**

برای توسعه به سادگی می‌توان Shape را تغییر داد و ویژگی‌ای به مستطیل و مربع اضافه کرد یا اینکه مستقلا ویژگی‌های خاصی به هر کدام از این کلاس‌ها افزود که باعث می‌شود هر تغییر باعث conflictی در کلاس‌های دیگر نشود.

- **SRP:**

همانطور که از کد مشخص است این اصل به خوبی رعایت شده است و هر کلاس تنها یک دلیل برای تغییر دارد. (تغییر اندازه)

تمامی کدهای اشاره شده از طریق https://github.com/alirezadizaji/software_lab/tree/main/R3 قابل دسترسی هستند!