دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

تمرین دوم

آزمون نرم‌افزار

نگارش
محمدرضا اخگری زیری
محمدعلی کشاورز
علی نظری

استاد درس
دکتر معصومه طارمی‌راد

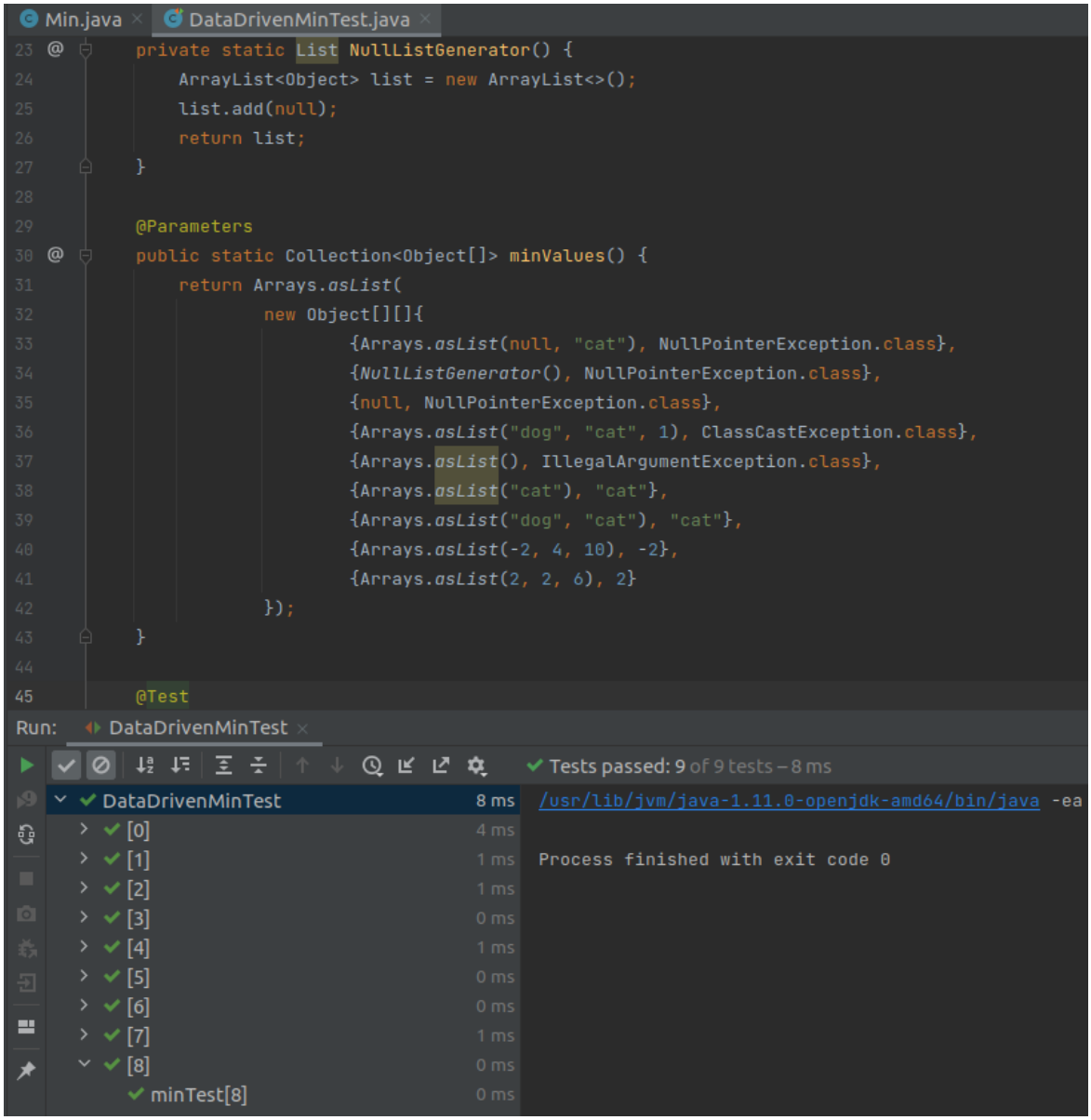
بهار ۱۴۰۰

فهرست مطالب

صفحه

سوال اول

در شـــکل ۱، نتایج اجرای این آزمون‌ها مشـــاهده می‌شـــود. موارد گفته شـــده در کلاس، در این آزمون‌ها پوشش داده شده است.



شکل ۱- نتایج اجرای آزمون‌ها

سوال دوم

الف)

باید ببینیم مواردی که می‌خواهیم با هم مقایسه کنیم چه چیزی هستند و چه ویژگی‌هایی دارند و بعد تصمیم بگیریم که چگونه این تابع را پیاده‌سازی کنیم. در حقیقت پیاده‌سازی این تابع به اطلاعات ما از مساله دارد، یعنی مثلاً ما می‌دانیم که برابری دو نقطه، طبق اطلاعات ما، حالتی است که دو نقطه دارای مختصات یکسانی باشند. هرچند ممکن است به خاطر تعریف مساله این موضوع فرق کند ولی در اینجا باید دو نقطه را با هم مقایسه کنیم که دو ویژگی طول و عرض را دارد، پس برای پیاده‌سازی این تابع باید این مقادیر طول و عرض هر دو نقطه مدنظر را با هم مقایسه کنیم و نتیجه‌ی آن را به صورت خروجی تابع بیان کنیم.

ب)

پیاده‌سازی این تابع در قطعه کد زیر قابل مشاهده است.

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Point point = (Point) o;
    return x == point.x && y == point.y;
}
```

در صورت ارث‌بری از کلاس دیگری، باید در پیاده‌سازی این تابع، تابع equals از کلاس پدر را نیز فراخوانی کنیم و در صورتی که هر دو تابع equals در کلاس پدر و فرزند اعلام برابری کنند، ما هم برابری را به عنوان خروجی اعلام می‌کنیم (البته این مساله باز به تعریف مساله بستگی دارد). در زیر، بروزرسانی تابع equals را مشاهده می‌کنیم.

```
return x == point.x && y == point.y && super.equals(o);
```

ج)

در شکل ۲، چهار آزمون را مشاهده می‌کنیم که حالت‌های دو نقطه‌ی مساوی، دو نقطه‌ی نامساوی، مقایسه با نقطه‌ی Null و مقایسه دو جنس متفاوت را پوشش می‌دهد. نتیجه‌ی اجرای این آزمون‌ها نیز قابل مشاهده است.

د)

در شکل ۳، آزمون‌های قسمت ج را به صورت داده‌رانه نوشته‌ایم و نتایج اجرای آن‌ها نشان داده‌ایم.

ه)

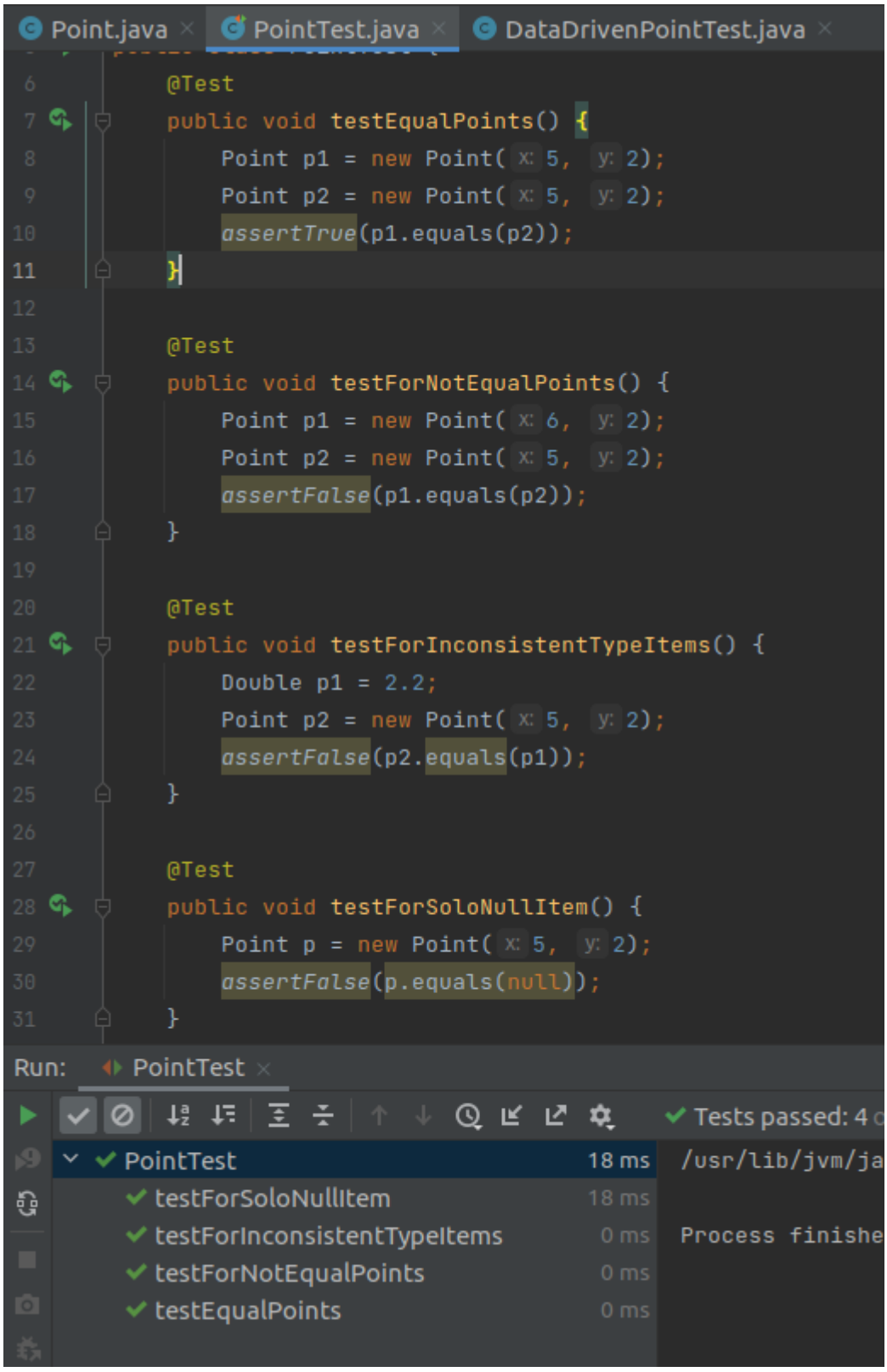
رابطه‌ی تساوی ویژگی‌های «بازتابی»، «تقارنی» و «تعدی» را داراست. همچنین باید به Null نبودن مقادیر هم توجه کنیم. این ویژگی‌ها به صورت تئوری‌های Junit در شکل ۴ قابل مشاهده است. همانظورکه قابل ملاحظه است، ما برای هر ویژگی تابع آزمونی نوشته‌ایم و در هر مورد هم به Null نبودن متغیرها توجه کرده‌ایم.

و)

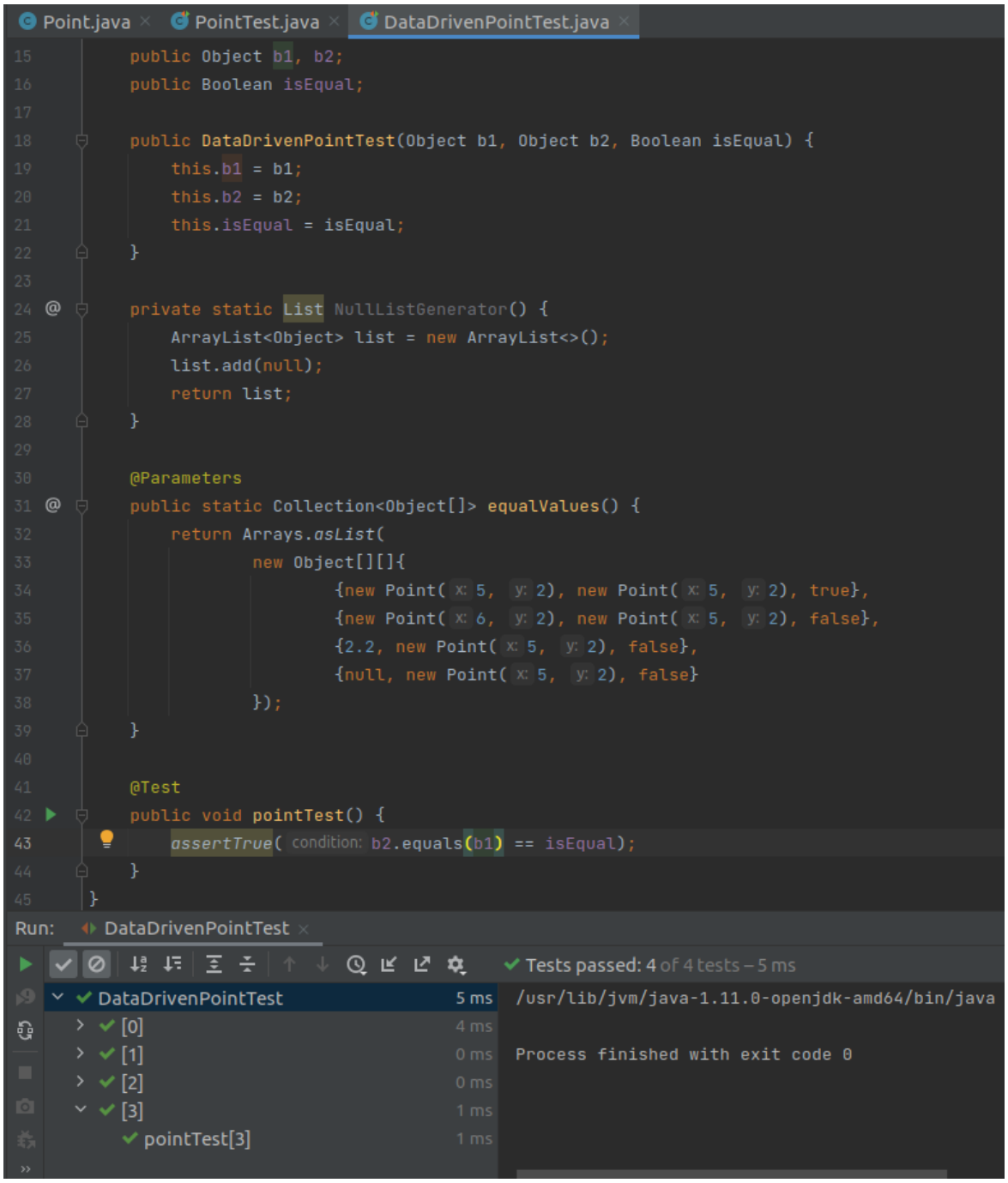
رابطه‌ی ریاضی بین این دو به این صورت است که اگر دو شی با هم برابر[1] باشند الزاما hashCodeهای یکسانی نیز دارند ولی برعکس این گزاره برقرار نیست یعنی اگر hashCodeها برابر باشند لزومی ندارد که این دو شی با هم equal باشند. برای مثال اگر ما تابع هش را به صورت زیر تعریف کنیم، دو نقطه‌ی (۲و۷) و (۷و۲) دارای هش برابر خواهند شد در حالی که دو نقطه برابر نیستند.

$$\text{hash}_p = p.x + p.y$$

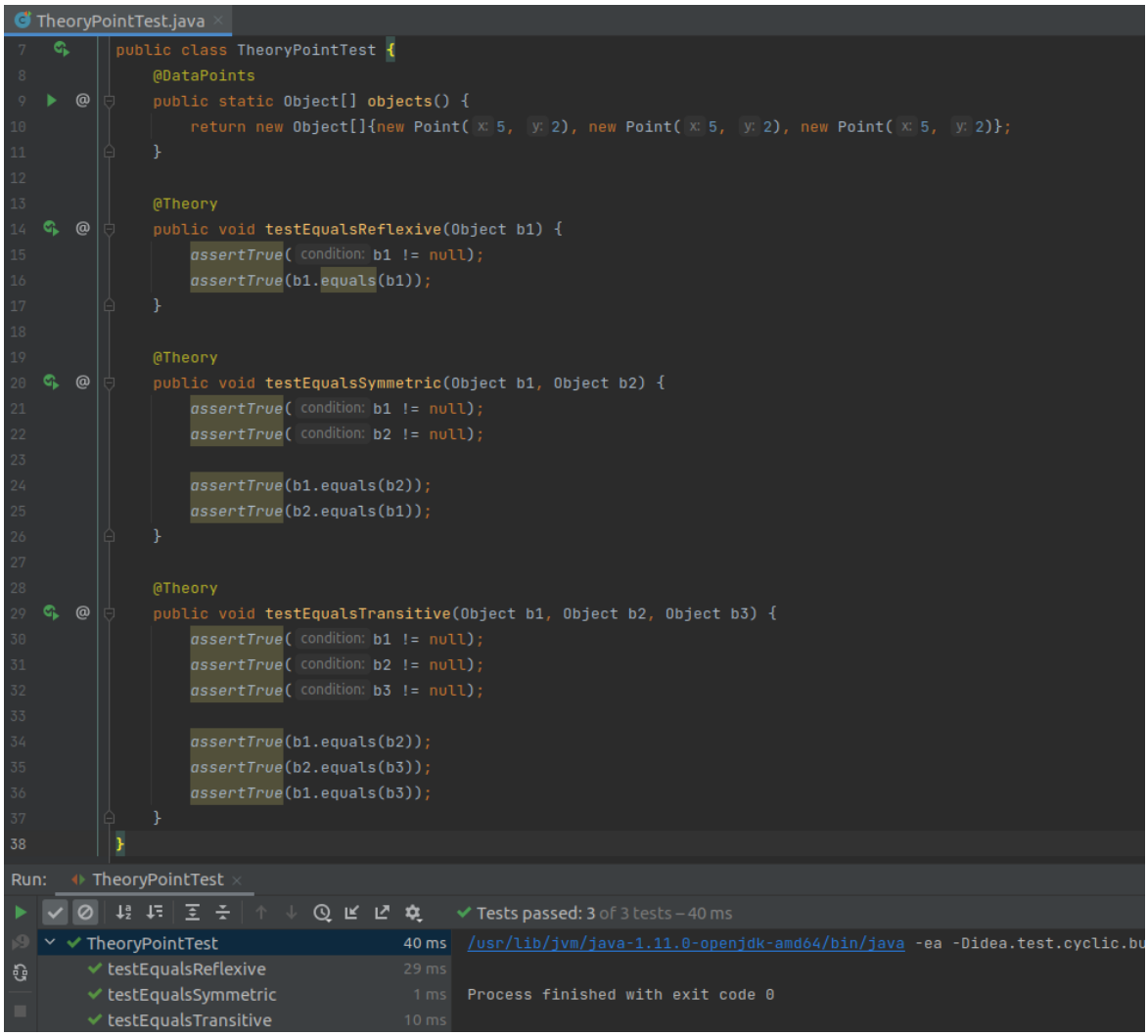
[1] equal



شکل ۲- چهار آزمون برای کلاس Point به همراه نتیجه‌ی اجرای آن‌ها



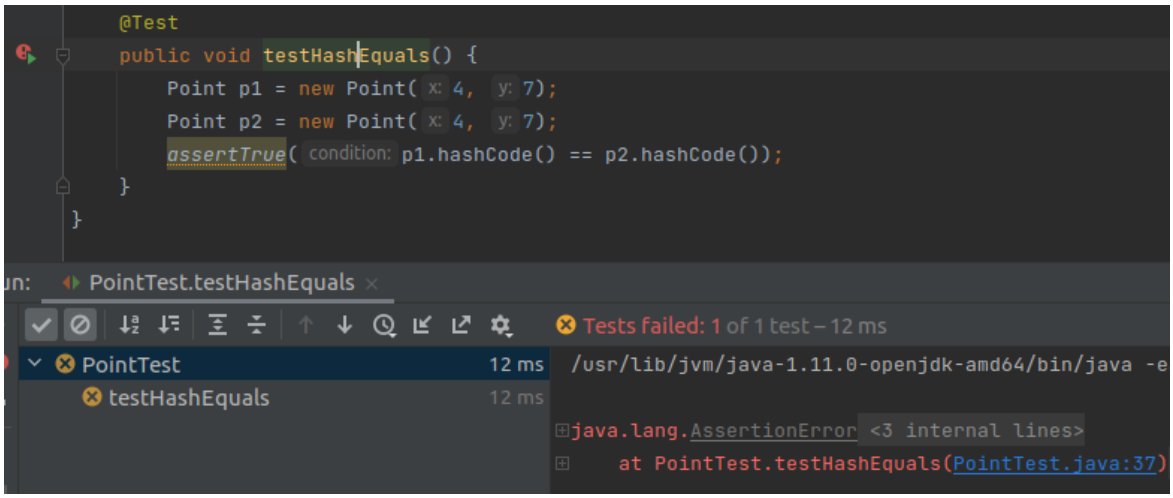
شکل ۳- اجرای آزمون‌ها به صورت داده‌رانه



شکل ۴- نمایش رابطه‌ی تساوی به صورت تئوری در junit

ز)

در شکل ۵، مشاهده می‌شود که چون تابع equals را عوض کردیم ولی تابع hashCode تغییری نکرده است، این رابطه برقرار نیست.



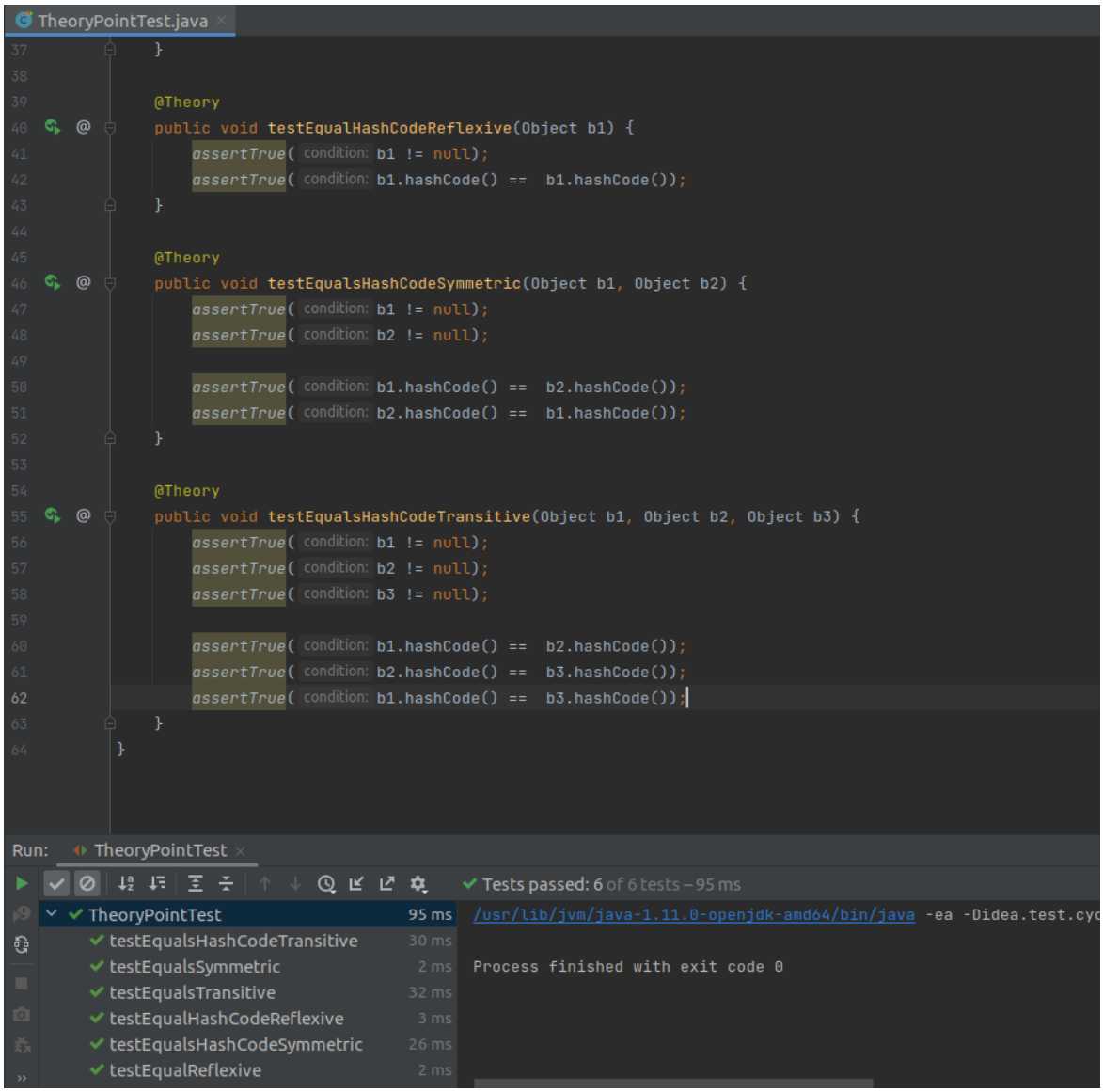
شکل ۵- نتیجه‌ی اجرای تست برابری مقدار hashCode

ح)

برای رفع این مشکل کافیست که تابع hashCode را بروز کنیم. در قطعه کد زیر این بروزرسانی قابل مشاهده است.

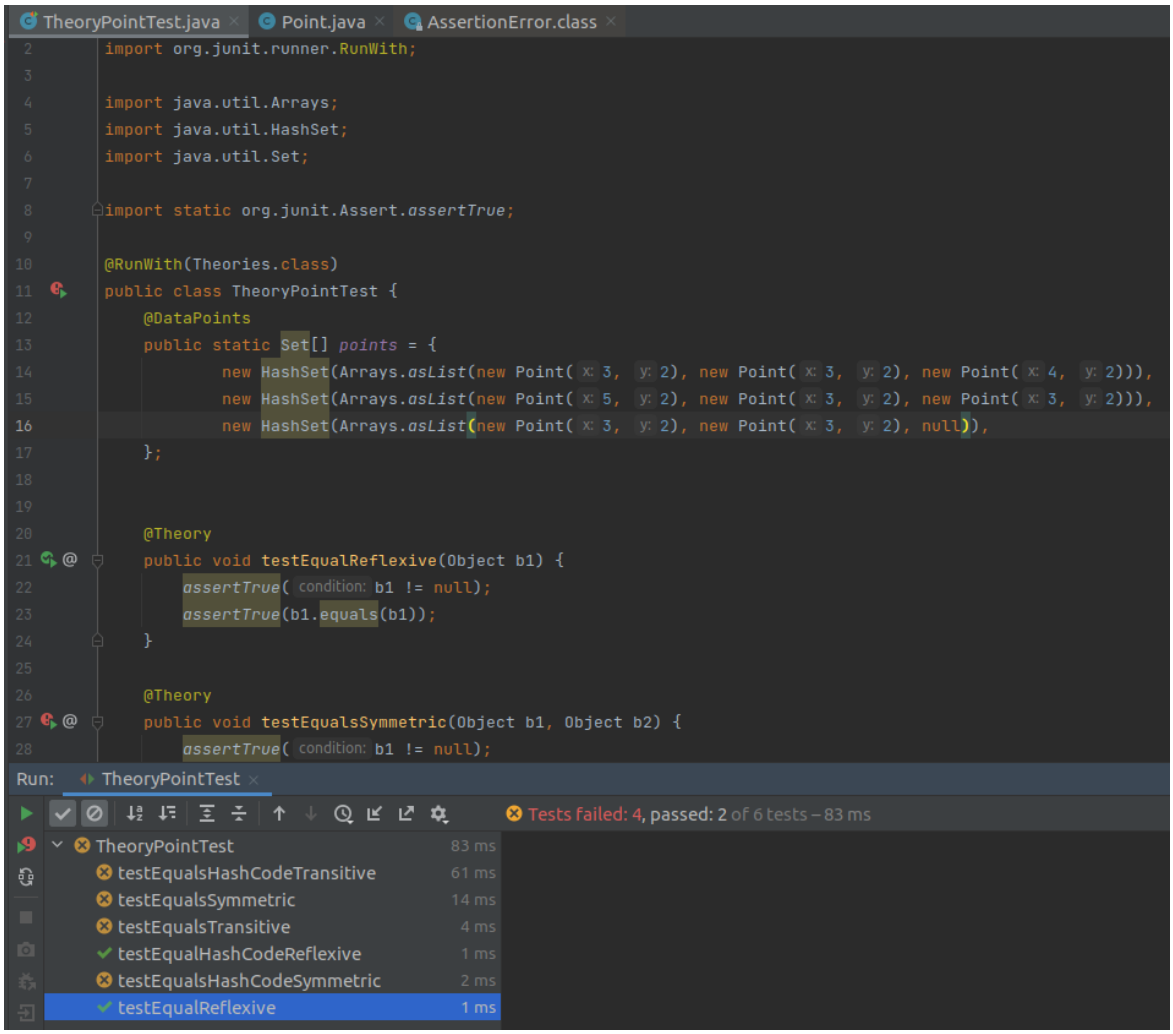
```
public class Point {
    private int x, y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Point point = (Point) o;
        return x == point.x && y == point.y;
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }
}
```

ط) همانطوریکه در قسمت‌های قبلی گفته شد، لازم است که رابطه‌ی تساوی در hashCodeها نیز برقرار باشد، برای همین تئوری‌های نوشته شده را برای hashCodeها نیز می‌نویسیم. در شکل ۶ تئوری‌های اضافه شده به تئوری‌های موجود را مشاهده می‌کنید.



شکل ۶- تئوری‌های اضافه شده برای کامل کردن رابطه‌ی تساوی

ی) در شکل ۷، تئوری‌های نوشته شده را با datapointهای مناسب ارزیابی کرده‌ایم. چون این نقاط تئوری‌های مشخص شده را ارضا نمی‌کنند، آزمون‌ها با شکست روبه رو شده‌اند (در شکل ۶ این تئوری‌ها را برای نقاط مساوی اجرا کرده‌ایم و همه‌ی آزمون‌ها قبول شدند).



شکل ۷- نمایش اجرای تئوری‌ها برای datapointهای درون شکل

گزارش انجام تمرین به صورت گروهی:

برای اجرای گروهی این تمرین، سه جلسه در google meat به مدت مجموعاً ۵ ساعت تشکیل شد و تمامی سوال‌ها با همفکری و همچنین جستوجو در اینترنت انجام شد. نهایتاً نسخه‌ی نهایی نوشته شد و به تایید تمامی اعضای گروه رسید.