

# آزمون نرم افزار

## تمرین عملی سری ۳

حوریه سلطانی-۴۰۰۱۹۱۷۳

- ۱

در این تمرین، هدف آزمون متدهای کلاس `BankAccountService` شامل `createAccount`، `deposit`، `withdraw`، `transfer`، `getBalance`، `exists` و `getAccount` بود. برای این منظور، کلاس `BankAccountServiceTest` در مسیر `test` تکمیل شد تا تمام حالات موفق و ناموفق برای هر تابع بررسی شود. همچنین باگ موجود در یکی از توابع کشف و اصلاح گردید.

### تکمیل توابع آزمون

برای هر یک از متدهای اصلی سرویس، تست‌های زیر نوشته شد:

- `depositTest()`: بررسی واریز صحیح، واریز مقدار منفی، واریز به حساب ناموجود.
- `withdrawTest()`: بررسی برداشت صحیح، برداشت بیش از موجودی، برداشت مقدار منفی، و برداشت از حساب ناموجود.
- `transferTest()`: بررسی انتقال موفق، انتقال بیش از موجودی، انتقال به/از حساب نامعتبر، و انتقال مقدار منفی.
- `getBalanceTest()`: بررسی دریافت موجودی برای حساب معتبر و نامعتبر.
- `existsAndGetAccountTest()`: بررسی وجود حساب و دریافت شیء `BankAccount` برای حساب موجود و ناموجود.

### کشف و اصلاح باگ

باگ موجود در متد `transfer` در کلاس `BankAccountService` کشف شد. در نسخه اولیه کد، موجودی حساب مقصد به اشتباه با استفاده از موجودی جدید حساب مبدأ محاسبه می‌شد:

```
to.setBalance(from.getBalance() + amount);
```

در این خط، چون موجودی `from` قبلاً کاهش یافته، مقدار محاسبه‌شده برای `to` اشتباه خواهد بود.

نسخه صحیح به این شکل اصلاح شد:

```
to.setBalance(to.getBalance() + amount);
```

با این اصلاح، موجودی حساب مقصد به درستی افزایش می‌یابد.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS COMMENTS
%TESTC 6 v2
%TSTTREE2,com.iut.BankAccountServiceTest,true,6,false,1,BankAccountServiceTest,,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]
%TSTTREE3,transferTest(com.iut.BankAccountServiceTest),false,1,false,2,transferTest(),,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]/[method:transferTest()]
%TSTTREE4,getBalanceTest(com.iut.BankAccountServiceTest),false,1,false,2,getBalanceTest(),,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]/[method:getBalanceTest()]
%TSTTREE5,depositTest(com.iut.BankAccountServiceTest),false,1,false,2,depositTest(),,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]/[method:depositTest()]
%TSTTREE6,existsAndGetAccountTest(com.iut.BankAccountServiceTest),false,1,false,2,existsAndGetAccountTest(),,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]/[method:existsAndGetAccountTest()]
%TSTTREE7,createAccountTest(com.iut.BankAccountServiceTest),false,1,false,2,createAccountTest(),,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]/[method:createAccountTest()]
%TSTTREE9,withdrawTest(com.iut.BankAccountServiceTest),false,1,false,2,withdrawTest(),,[engine:junit-jupiter]/[class:com.iut.BankAccountServiceTest]/[method:withdrawTest()]
%TESTS 3,transferTest(com.iut.BankAccountServiceTest)
%TESTE 3,transferTest(com.iut.BankAccountServiceTest)
%TESTS 4,getBalanceTest(com.iut.BankAccountServiceTest)
%TESTE 4,getBalanceTest(com.iut.BankAccountServiceTest)
%TESTS 5,depositTest(com.iut.BankAccountServiceTest)
%TESTE 5,depositTest(com.iut.BankAccountServiceTest)
%TESTS 6,existsAndGetAccountTest(com.iut.BankAccountServiceTest)
%TESTE 6,existsAndGetAccountTest(com.iut.BankAccountServiceTest)
```

## توضیح خروجی اجرای تست‌ها

پس از تکمیل کلاس آزمون `BankAccountServiceTest` و رفع باگ در متد `transfer`، تست‌ها با استفاده از `Maven` اجرا شدند. همه تست‌ها با موفقیت اجرا شدند و هیچ خطا (error) یا شکست (failure) در خروجی دیده نشد. این موضوع در خطوط `%TESTS` و `%TESTE` مربوط به هر تست قابل مشاهده است که نشان‌دهنده شروع و پایان موفق هر تابع تست می‌باشد.

## ۲-

در متد `averageTest` یکی از ورودی‌ها شامل مقادیر اعشاری (0.1 و 0.2) است. در زبان `Java`، محاسبه اعداد اعشاری با دقت مطلق انجام نمی‌شود و ممکن است اختلاف بسیار کوچکی نسبت به مقدار مورد انتظار وجود داشته باشد. به همین دلیل، مقایسه مستقیم با استفاده از `assertEquals(a, b)` منجر به شکست تست می‌شود. برای حل این مشکل، باید از نسخه‌ای از `assertEquals` استفاده شود که یک مقدار خطای مجاز (delta) تعیین می‌کند. با افزودن `1e-9` به عنوان دقت، تست به درستی پاس می‌شود و این مشکل برطرف می‌گردد.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS COMMENTS
%TESTC 0 v2
%TSTTREE2,com.iut.MathUtilsTest,true,6,false,1,MathUtilsTest,,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]
%TSTTREE3,averageTest(com.iut.MathUtilsTest),true,0,false,2,averageTest(double[],double),[D],double,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:averageTest(%SBD\,double)]
%TSTTREE4,isPrimeTest(com.iut.MathUtilsTest),true,0,false,2,isPrimeTest(int\,boolean),int\,boolean,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:isPrimeTest(int\,boolean)]
%TSTTREE5,gcdTest(com.iut.MathUtilsTest),true,0,false,2,gcdTest(int\,int\,int),int\,int\,int,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:gcdTest(int\,int\,int)]
%TSTTREE6,factorialTest(com.iut.MathUtilsTest),true,0,false,2,factorialTest(int\,long),int\,long,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:factorialTest(int\,long)]
%TSTTREE7,powerTest(com.iut.MathUtilsTest),true,0,false,2,powerTest(double\,int\,double),double\,int\,double,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:powerTest(double\,int\,double)]
%TSTTREE8,fibonacciTest(com.iut.MathUtilsTest),true,0,false,2,fibonacciTest(int\,int),int\,int,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:fibonacciTest(int\,int)]
%TSTTREE9,averageTest(com.iut.MathUtilsTest),false,1,true,3,[1] [1.0\, 2.0\, 3.0\, 4.0]\, 2.5,[D],double,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:averageTest(%SBD\,double)]/[test-template-invocation:#1]
%TESTS 9,averageTest(com.iut.MathUtilsTest)
%TESTE 9,averageTest(com.iut.MathUtilsTest)
%TSTTREE10,averageTest(com.iut.MathUtilsTest),false,1,true,3,[2] [10.0\, 10.0\, 10.0]\, 10.0,[D],double,[engine:junit-jupiter]/[class:com.iut.MathUtilsTest]/[test-template:averageTest(%SBD\,double)]/[test-template-invocation:#2]
%TESTS 10,averageTest(com.iut.MathUtilsTest)
%TESTE 10,averageTest(com.iut.MathUtilsTest)

Test Runner for Java
[1] [1.0, 2.0, 3.0, 4.0], 2.5
[2] [10.0, 10.0, 10.0], 10.0
[3] [0.0, 0.0, 0.0, 0.0], 0.0
[4] [56.0, 11.0, 25.0, 159.0, 153.0], 80.8
[5] [0.1, 0.2], 0.15
[1] 1, false
[2] 2, true
[3] 10, false
[4] 101, true
[5] 1000, false
[1] 54, 24, 6
[2] 100, 10, 10
[3] 49, 7, 7
[4] 17, 13, 1
[5] 0, 5, 5
```

در این تمرین، برای کلاس `MathUtils` تست‌هایی با استفاده از `@ParameterizedTest` نوشته شد که عملکرد متدهای `factorial`، `power`، `gcd`، `isPrime` و `fibonacci` را در شرایط مختلف بررسی می‌کنند. داده‌های ورودی شامل مقادیر مثبت، صفر، منفی (در صورت نیاز)، و مقادیر اعشاری بودند. پس از اجرای تست‌ها، خروجی نشان داد که تمامی تست‌ها بدون خطا و شکست اجرا شده‌اند و عملکرد متدها مطابق انتظار بوده است. این نتیجه تأیید می‌کند که پیاده‌سازی متدهای کلاس `MathUtils` از نظر منطقی درست و قابل اعتماد است.

در این تمرین، کلاس `CsvManager` که وظیفه‌ی ایجاد، حذف و مدیریت فایل‌های CSV را بر عهده دارد، با استفاده از تست واحد ارزیابی شد. برای این منظور، کلاس `CsvManagerTest` به گونه‌ای طراحی شد که توابع آن به ترتیب خاصی اجرا شوند تا وابستگی بین داده‌ها در طول تست حفظ گردد.

جهت اطمینان از ترتیب اجرای تست‌ها، از annotation های زیر استفاده شد:

- متد `generateCsv()` با `BeforeAll@` برای ایجاد فایل CSV پیش از اجرای تست‌ها
- متد `deleteCsv()` با `AfterAll@` برای حذف فایل پس از اتمام همه‌ی تست‌ها
- متدهای تست اصلی مانند `insertDataTest`، `findDataByRowTest`، `replaceRowTest` و `testFindDataByRow` با `Test@` و `Order(n@)` شماره‌گذاری و مرتب شدند تا ابتدا درج داده، سپس جستجو و جایگزینی آن‌ها انجام گیرد.






```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.iut.CsvManagerTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.075 s - in com.iut.CsvManagerTest
[INFO] Running com.iut.BankAccountServiceTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.017 s - in com.iut.BankAccountServiceTest
[INFO] Running com.iut.MathUtilsTest
[INFO] Tests run: 32, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.206 s - in com.iut.MathUtilsTest
[INFO] Results:
[INFO] Tests run: 42, Failures: 0, Errors: 0, Skipped: 0
[INFO] --- jacoco:0.8.11:report (report) @ software-testing ---
[INFO] Loading execution data file /Users/houri/Downloads/ex2/target/jacoco.exec
[INFO] Analyzed bundle 'software-testing' with 5 classes
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:37 min
[INFO] Finished at: 2025-05-09T22:02:59+03:30
[INFO] -----
```

پس از اجرای موفق تست‌ها با دستور `mvn test`، خروجی نشان داد که هر ۴ تست مربوط به `CsvManagerTest` بدون خطا و با موفقیت اجرا شدند. همچنین هیچ تستی از کلاس‌های `BankAccountServiceTest` و `MathUtilsTest` نیز با شکست مواجه نشد و مجموعاً ۴۲ تست اجرا و پاس شدند.

برای اندازه‌گیری میزان پوشش تست‌ها، پلاگین `jacoco` به فایل `pom.xml` اضافه شده بود. پس از اجرای تست‌ها، گزارش پوشش تست در مسیر `target/site/jacoco/index.html` تولید شد. بررسی این گزارش نشان داد:

- پوشش دستورات (Instruction Coverage): ۹۱٪
- پوشش شاخه‌ها (Branch Coverage): ۸۲٪

## software-testing

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 <a href="#">com.iut.csv</a>	<div><div></div></div>	86%	<div><div></div></div>	65%	9	20	14	53	0	7	0	1
 <a href="#">com.iut.math</a>	<div><div></div></div>	88%	<div><div></div></div>	87%	5	23	3	33	1	7	0	1
 <a href="#">com.iut.bank.model</a>	<div><div></div></div>	67%	<div><div></div></div>	50%	2	6	2	11	1	5	0	1
 <a href="#">com.iut.bank.repo</a>	<div><div></div></div>	97%	<div><div></div></div>	87%	1	10	1	20	0	6	0	1
 <a href="#">com.iut.bank.service</a>	<div><div></div></div>	100%	<div><div></div></div>	95%	1	19	0	42	0	8	0	1
Total	55 of 643	91%	16 of 90	82%	18	78	20	159	2	33	0	5

که هر دو مقدار بالاتر از حد مطلوب ۸۰٪ هستند و نشان می‌دهند که تست‌های نوشته‌شده بخش عمده‌ای از منطق کد را پوشش داده‌اند. به طور خاص، کلاس CsvManager نیز با ۸۶٪ پوشش دستوری و ۶۵٪ پوشش شاخه‌ای، در وضعیت خوبی قرار دارد.

---