آزمون نرم افزار

تمرین ۲

محمد جواد رنجبر

۸۱۰۱۰۱۱۷۳

سوال ۱

ابتدا برای Gedcom_Service.java تست‌هایی می‌نویسم تا از صحت این کلاس مطمئن شویم، این تست‌ها شامل موارد زیر می‌باشد:

ساختار و عملکرد

تنظیم و تخریب

`setUp()`: این متد با انوتیشن `@BeforeEach` مشخص شده و قبل از هر تست فراخوانی می‌شود. این متد `Gedcom_Service`را با فراخوانی `Gedcom_Service.Nulify()` مقداردهی اولیه می‌کند.

`tear_down()`: با انوتیشن `@AfterEach`مشخص شده و پس از هر تست برای بازنشانی وضعیت `Gedcom_Service` صد می‌شود.

متدهای تست

testBirthBeforeDeathValidFILEWithoutfile()

هدف :تست اینکه تاریخ تولد فرد قبل از تاریخ مرگ او باشد.

پیاده‌سازی :فردی با تاریخ تولد و مرگ معتبر ایجاد کرده و اطمینان حاصل می‌کند که متد `birthBeforeDeath` استثنایی پرتاب نمی‌کند.

testBirthBeforeDeathValidwithoutfiles()

هدف :مشابه تست قبلی اما بدون وابستگی به فایل‌های خارجی.

پیاده‌سازی :اطمینان حاصل می‌کند که همان عملکرد مستقیماً در تست موجود است.

testReadAndParseFile()

هدف :تست قابلیت‌های خواندن و تجزیه فایل.

پیاده‌سازی :یک فایل تست GEDCOM را خوانده و بررسی می‌کند که تعداد افراد و خانواده‌های درست تجزیه شوند.

testBirthBeforeDeath()

هدف :تست متد `birthBeforeDeath` با استفاده از یک فایل.

پیاده‌سازی :یک فایل GEDCOM را خوانده و سازگاری تاریخ تولد-مرگ را بررسی می‌کند.

TestParser()

هدف :اطمینان حاصل می‌کند که تجزیه‌کننده به درستی جزئیات فردی را پر می‌کند.

پیاده‌سازی :اطمینان حاصل می‌کند که فیلدهای مورد نیاز برای افراد تجزیه شده خالی یا null نیستند.

testMarriageBeforeDivorce()

هدف :تست اینکه ازدواج‌ها قبل از طلاق‌ها در داده‌ها ثبت شده‌اند.

پیاده‌سازی :یک فایل GEDCOM را خوانده و سازگاری ازدواج-قبل-از-طلاق را بررسی می‌کند.

testAuntsandUnclesname()

هدف :تست نام‌گذاری عموها و عمه‌ها در خانواده‌ها.

پیاده‌سازی :یک فایل GEDCOM را خوانده و نام عموها و عمه‌ها را بررسی می‌کند.

testBirthBeforeMarriageOfParent()

هدف :اطمینان حاصل می‌کند که تولد کودکان بعد از ازدواج والدینشان اتفاق افتاده است.

پیاده‌سازی :این رابطه را با استفاده از یک فایل تست GEDCOM بررسی می‌کند.

testMaleLastname()

هدف :تست اینکه اعضای مرد خانواده دارای نام خانوادگی یکسان هستند.

پیاده‌سازی :فایل‌های GEDCOM را خوانده و سازگاری نام خانوادگی مردان را بررسی می‌کند.

testGetMonth()

هدف :تست تبدیل اختصارات ماه به مقادیر عددی.

پیاده‌سازی :اطمینان حاصل می‌کند که تبدیل به درستی برای تمامی ماه‌ها انجام می‌شود.

حال به بررسی موارد خواسته شده در صورت سوال می‌پردازیم:

I. برخی جهش‌های باید نامعتبر باشند

این مورد در کد ما نشان داده نمی‌شود، اما شامل جهش‌هایی هست که دچار not coverage می‌شوند و شامل اینگونه جهش‌ها می‌شود، همچنین بعضی از جهش‌ها باعث ایجاد loop بینهایت می‌شوند که ارور TIMED_OUT خواهیم داشت.

```
52  1. removed call to edu/stevens/ssw555/Gedcom_Service::AuntsandUnclesname → NO_COVERAGE
54  1. removed call to edu/stevens/ssw555/Gedcom_Service::uniqueFamilynameBySpouses → NO_COVERAGE
57  1. removed call to java/io/PrintStream::println → NO_COVERAGE
58  1. removed call to edu/stevens/ssw555/Gedcom_Service::main → NO_COVERAGE
70  1. negated conditional → KILLED
74  1. negated conditional → KILLED
77  1. negated conditional → KILLED
    2. negated conditional → KILLED
79  1. negated conditional → TIMED_OUT
84  1. negated conditional → KILLED
85  1. removed call to edu/stevens/ssw555/Individual::setName → KILLED
    2. Replaced integer subtraction with addition → KILLED
86  1. negated conditional → KILLED
87  1. removed call to edu/stevens/ssw555/Individual::setSex → KILLED
88  1. negated conditional → KILLED
```

شکل ۱ نامعتبر

II. برخی جهش‌ها معادل با کد اصلی باشند.

این جهش‌ها شامل، جهش‌های not coverage می‌شوند که که تست‌های ما اصلا اجازه‌ی بررسی یک بخش‌هایی را نمیده و باید تست‌های بیشتری برای آن‌ها تعریف کنیم. در این تابع main هیچوقت مورد بررسی کد قرار نمی‌گیرد و بنابراین جهش معادل با کد اصلی خواهد بود.

```
35          public static void main(String[] args) throws IOException, ParseException {
36  1               System.out.println("Please Enter the Input File Path with filename: ");
37                  try {
38                          BufferedReader bufferRead = new BufferedReader(new InputStreamReader(System.in));
39                          String fileName = bufferRead.readLine();
40  1                       createOutputFile();
41  1                       readAndParseFile(fileName);
42                          //printMaps();
43                          //UuserStory 3
44  1                       birthBeforeDeath(individuals);
45                          //UserStory 4
46  1                       Marriagebeforedivorce(individuals, families);
47                          //UserStory 8
48  1                       birthbeforemarriageofparent(individuals, families);
49                          //UserStory 16
50  1                       Malelastname(families);
51                          //UserStory 20
52  1                       AuntsandUnclesname(families);
53                          //UserStory 24
54  1                       uniqueFamilynameBySpouses(individuals, families);
55
56                  } catch (FileNotFoundException ex) {
57  1                       System.out.println("File Not Found. Please reenter path");
58  1                       main(null);
59                  }
60          }
```

شکل ۲ بدون تغییر

III. چند مورد باید معتبر اما غیر مفید باشد.

این تست‌ها شامل جهش‌هایی می‌شود که توسط همه‌ی تست‌ها مشخص می‌شوند، مثلا در بخش خواندن فایل، جهش توسط همه‌ی تست‌ها قابل فهمیدن می‌باشد. در این بخش جهش‌هایی که روی خط ۷۰ اتفاق می‌افتد، توسط هر تستی که این خط را صدا می‌زند، یافت می‌شود و این جهش را غیر مفید می‌کند.

```
61
62          public static void readAndParseFile(String fileName) throws IOException {
63
64                  String[] validTags = { "INDI", "NAME", "SEX", "BIRT", "DEAT", "FAMC", "FAMS", "FAM", "MARR", "HUSB", "WIFE"
65                                  "CHIL", "DIV", "DATE", "HEAD", "TRLR", "NOTE" };
66                  BufferedReader br = new BufferedReader(new FileReader(fileName));
67                  String line = br.readLine();
68
69
70  1               while (line != null) {
```

شکل ۳ معتبر غیر مفید

IV. دو یا سه مورد جهش باید مفید باشد.

این جهش‌ها شامل، جهش‌هایی مربوط به هرکدام از تست‌هایی است که نوشتیم و kill شده‌اند.

```
74  1                   if (Arrays.asList(validTags).contains(parts[1])) {
75                              line = br.readLine();
76                      } else {
77  2                           if (parts[0].equals("0") && Arrays.asList(validTags).contains(parts[2])) {
78
79  1                                   if (parts[2].equals("INDI")) {
80                                              Individual indi = new Individual(parts[1]);
81                                              String individualParts = br.readLine();
82                                              do {
83                                                      String[] indParts = individualParts.split(" ");
84  1                                                   if (indParts[1].equals("NAME"))
85  2                                                           indi.setName(indParts[2] + " " + indParts[3].substring(1, indParts[3].length() - 1));
86  1                                                   if (indParts[1].equals("SEX"))
87  1                                                           indi.setSex(indParts[2]);
88  1                                                   if (indParts[1].equals("FAMS"))
```

شکل ٤ مفید

V. بررسی جهش‌های زنده و نوشتن تست برای هدف قرار دادن آن‌ها

این کار در حین روند انجام این پروژه برای تست‌های مختلف انجام شده است که از survived به kill تبدیل شده‌اند.

VI. جهش‌ها تکراری یکدیگیر باشند.

این نیز شامل جهش‌هایی می‌شود که بر یک قسمت کد اعمال می‌شوند ولی باعث می‌شوند رفتار کد تغییر نکند، برای مثال در کد زیر تغییر مقدار if با remove کردن دستور write یک تغییر یکسان در کد ایجاد می‌کنند و فرقی با یکدیگر ندارند.

```
327 1                                   if (fam.getDivorce() != null) {
328                                             divorceDate = sdf.parse(fam.getDivorce());
329 1                                           if (birthDate.after(divorceDate)) {
330 1                                                   writeToFile(
331                                                                     "ERROR: User Story US08: Birth After Divorce Date\nFamily ID: "
332                                                                             + fam.getId() + "\nIndividual: " + indi.getId() + ": " + indi.getName
333                                                                             + " Has been born after parents' divorce\nDOB: " + indi.getBirth()
334                                                                             + " Parents Divorce Date: " + fam.getDivorce()
335                                                                             + "\n\n");
336                                             }
```

شکل ٥ تکراری

سوال ۲

الف)

برای بررسی تمام مسیرهای اجرای تابع h با استفاده از روش اجرای نمادی، ابتدا باید مسیرهای مختلف اجرای تابع را تعیین کنیم و شرایط ورودی را برای هر مسیر استخراج کنیم. این تابع به صورت زیر عمل می‌کند:

- اگر x کمتر یا مساوی با 31*6 باشد ($x \leq 6 * 31$) یعنی روز در نیمه اول سال است. در این صورت، ماه و روز را محاسبه کرده و چاپ می‌کند.
- اگر x بزرگتر از 31*6 باشد ($x > 6 * 31$) یعنی روز در نیمه دوم سال است. حال داریم:
  - اگر x کمتر یا مساوی با 30*5 باشد ($x \leq 5 * 30$) ماه و روز را برای نیمه دوم سال محاسبه کرده و چاپ می‌کند.
  - اگر x بزرگتر از 30*5 باشد ($x > 5 * 30$) وارد ماه آخر سال (اسفند) می‌شود.
    - اگر سال کبیسه باشد و اگر x کمتر یا مساوی با 30 باشد ($x \leq 30$) روز در اسفند را چاپ می‌کند.
    - اگر سال کبیسه باشد و اگر x کمتر یا مساوی با 30 باشد ($x \leq 30$) روز در اسفند را چاپ می‌کند.
    - در غیر این صورت، خطا رخ می‌دهد.

حال در حالت کلی آزمایه‌ها به شرح زیر خواهند بود:

برای اجرای نمادین، مسیرهای مختلف به صورت زیر هستند:

- مسیر اول $x \leq 6 * 31$
  - آزمایه: h(x, y) که در آن $0 \leq x \leq 6 * 31$
- مسیر دوم $6 * 31 < x \leq 6 * 31 + 5 * 30$
  - آزمایه: h(x, y) که در آن $6 * 31 \leq x \leq 6 * 31 + 5 * 30$
- مسیر سوم $6 * 31 + 5 * 30 < x \leq 365$ و سال کبیسه باشد
  - آزمایه: h(x, leap_year) و $6 * 31 + 5 * 30 < x \leq 365$
- مسیر چهارم $6 * 31 + 5 * 30 < x \leq 365$ و سال کبیسه نباشد
  - آزمایه: h(x, not_leap_year) و $6 * 31 + 5 * 30 < x \leq 365$
- مسیر پنجم $x > 365$
  - آزمایه: h(x, y) که در آن $365 < x$

حال با توجه به اینکه نیازی به سال کبیسه برای دیدن همه‌ی مسیرها نداریم کافیست تست‌کیس‌های زیر را تولید کنیم:

۱. $x \leq 6 * 31$ مثلا $x = 5$، خطوط دیده شده: ۱، ۲، ۳ و ۴
۲. $6 * 31 < x \leq 6 * 31 + 5 * 30$ مثلا x=190 خطوط دیده شده: ۱، ۲، ۳، ۵، ۶، ۷ و ۸
۳. $6 * 31 + 5 * 30 < x \leq 365$ مثلا x=363 خطوط دیده شده: ۱، ۲، ۳، ۵، ۶، ۷، ۹، ۱۰، ۱۱، ۱۲ و ۱۳
۴. $365 < x$ مثلا x=900 خطوط دیده شده: ۱، ۲، ۳، ۵، ۶، ۷، ۹، ۱۰، ۱۱، ۱۲، ۱۴ و ۱۵

ب) برای حل این مشکل آن بخش کد را به صورت زیر بازنویسی می‌کنیم:

```
boolean leap = isLeapYear(y);
if (leap) {
    if (x <= 30) {
        System.out.println(12 + " " + x);
    } else {
        throw new RuntimeException();
    }
} else {
    if (x <= 29) {
```

```
        System.out.println(12 + " " + x);
    } else {
        throw new RuntimeException();
    }
                                                                    }
```