

Test

Name: Mohammadali Rashidfarokhi

Email address: mr223jp@student.lnu.se

Link to my GitHub repo: <https://gitlab.lnu.se/>

Objective

The main purpose of this assignment is to test the provided code from the previous iteration which provided the player to choose random characters from a chosen word based on different levels.

What to test and How

I intended to test the methods that are suitable enough to be executed and some test to check its validity have been provided that run dynamic manual test-cases. At the beginning the TC1 will be evaluated. I also write automated unit tests for all player-methods in the class player test. However, as some methods cannot be used to write tests (e.g. the methods that will ask the player to provide some information and the ones that are not type void) in j unit all the methods in the class player will be used to write test cases for each one of them.

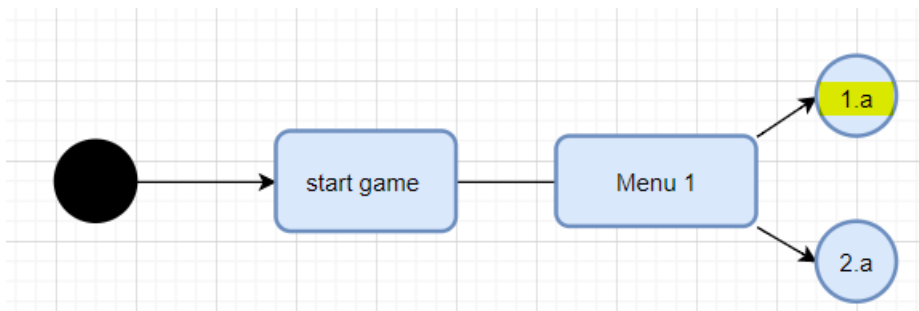
Time plan

Task	Estimated	Actual
Manual Tc	5 h	3 h
Unit tests	2 h	1:30 m
Running manual tests	1h	15 m
Code inspection	2 h	1 h
Test report	1 h	45 m

Manual Test-cases

Tc1.1 user choose to play the game

Scenario: the player chooses to continue (1.a) in Menu1.



The main scenario of this UC1.1 is tested where the user is faced with a menu (Menu 1) with two options and choose 1.a to continue.

Precondition: The player should not enter anything but 1.a

Test steps

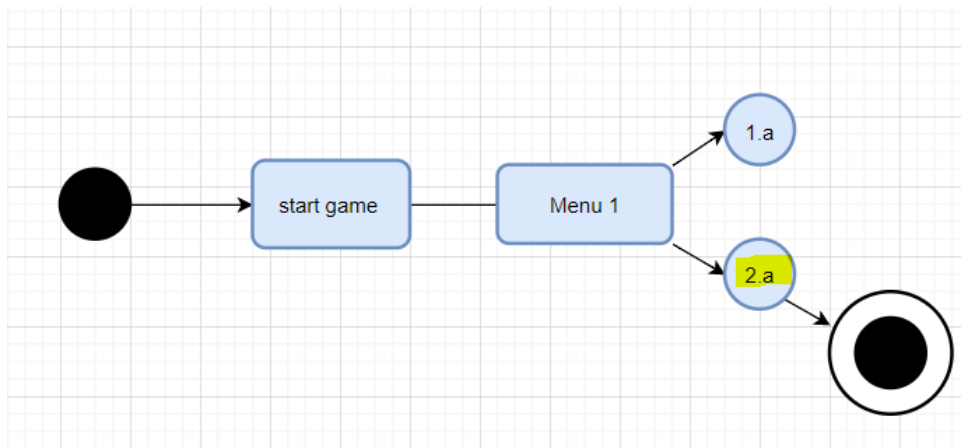
- The player will run the game in terminal.
- User press 1.a in the keyboard.
- User press enter in the keyboard.

Expected

- Menu to choose the level appear.
- The terminal shows the message “choose the level”.
- Press 1 for medium.
- Press 2 for hard

Tc1.2 User choose to leave the game.

Scenario: User will choose option 2.a (Exit game) in the menu1.



The main scenario of this UC1.1 is tested where the user is faced with a menu (Menu1) with two options and choose 2.a to Exit the game.

Precondition: The player should not enter anything but 2.a

Test steps

- The player will run the game in terminal.
- User press 2.a in the keyboard.

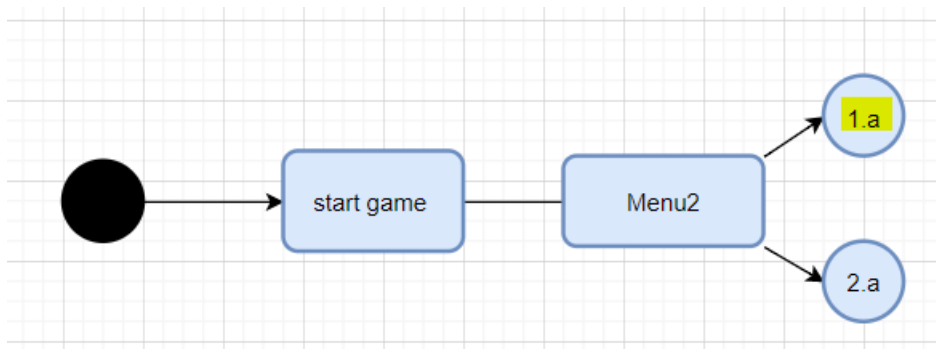
- User press enter in the keyboard.

Expected:

- The system will show a goodbye message.
- The program ends.

TC2.1 user choose mode moderate

Scenario: The player choose option 1.a (moderate mode) in the menu2.



The main scenario of this UC2.1 is tested where the user is faced with a menu (Menu2) with two options and choose 1.a to set the game mode as moderate.

Precondition: The player should not enter anything but 1.a

Steps

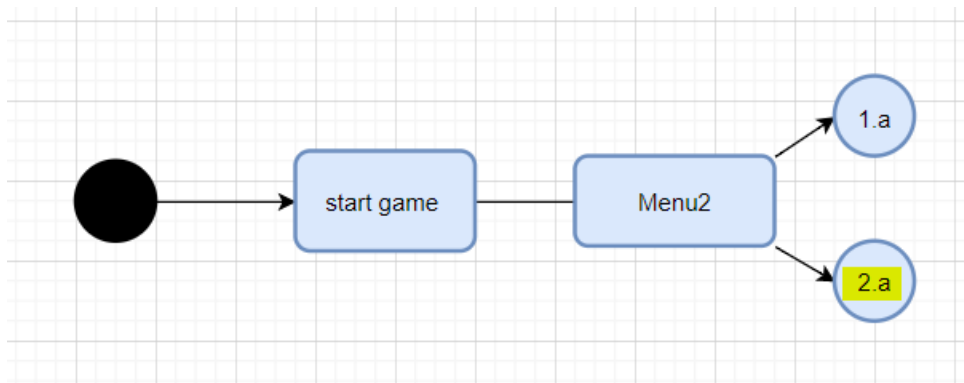
- The player will run the program.
- User enters in keyboard to continue the game.
- User will be directed to choose level menu (menu 2).
- User will choose option 1.a in keyboard for moderate mode.
- User press enter in keyboard and set the game mode as moderate.

Expected

- The player will be given 8 attempts to find the missing letters.

TC2.2 user choose mode Hard

Scenario: The player choose option 2.a (Hard mode) in Menu2.



The main scenario of this UC2.2 is tested where the user is faced with a menu (Menu2) with two options and choose 2.a to set the game mode as Hard.

Precondition: The player should not enter anything but number 2.

Steps

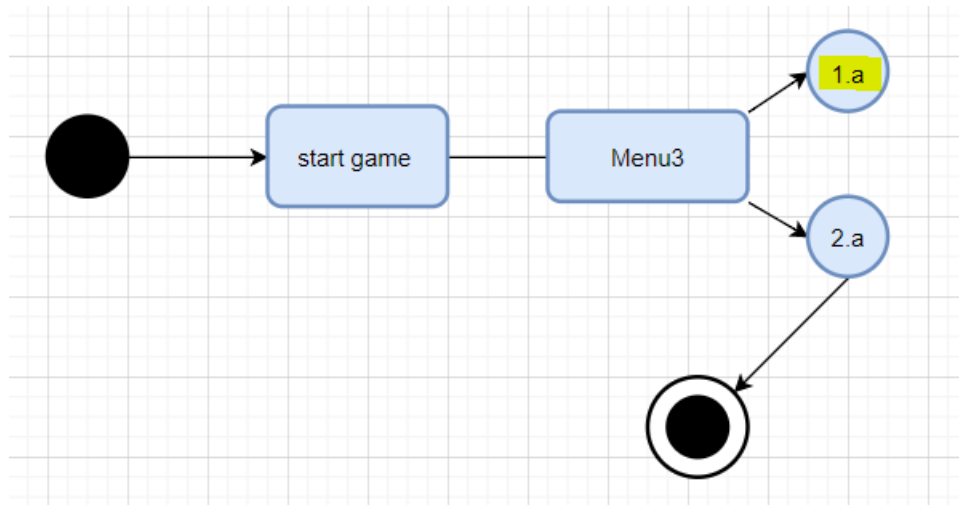
- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu 2).
- User will choose option 2.a in keyboard for hard mode.
- User press enter in keyboard and set the game mode as hard.

Expected

- The player will be given 6 attempts to find the missing letters.

TC3.1 user choose restart

Scenario: The player choose option 1.a (Restart game) in the Menu3.



The main scenario of this UC3.1 is tested where the user is faced with a menu3 with two options and choose 1.a to restart the game.

Precondition: The player should not enter anything but 1.a

Steps

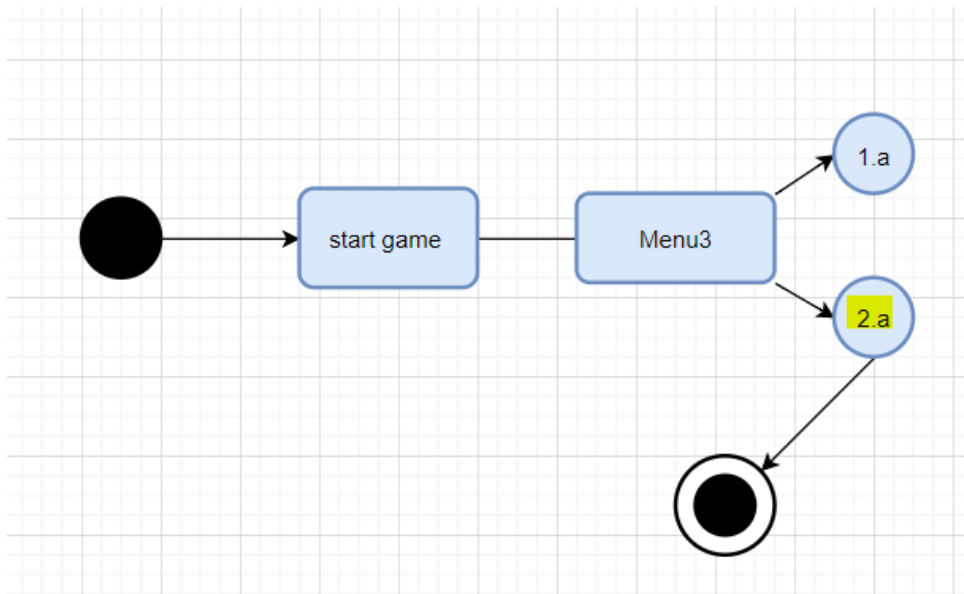
- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu2).
- User will choose options 1.a or 2.a in keyboard for hard and moderate mode.
- User press enter in keyboard and set the game mode as hard or moderate.
- User will guess the missing letters.
- User will be directed to menu3.
- User will enter 1.a in keyboard to restart the game.

Expected

- The System will direct the user to the game menu stage.

TC3.2 user Exit the game

Scenario: The player choose option 2.a (Exit game) in the game menu3.



The main scenario of this UC3.2 is tested where the user is faced with a menu (menu3) with two options and choose 2.a to exit the game.

Precondition: The player should not enter anything but 2.a

Steps

- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu2).
- User will choose options 1.a or 2.a in keyboard for hard and moderate mode.
- User press enter in keyboard and set the game mode as hard or moderate.
- User will guess the missing letters.
- User will be directed to the result menu (menu3).
- User will enter 2.a in keyboard to exit the game.

Expected

- The system will ask the user to enter some information.
- The system will provide user with few messages.
- The program ends.

Test reports

Test	UC1
TC1.1	1/OK
TC1.1	1/OK
Coverage & Success	2/OK

Test	UC2
TC2.1	1/OK
TC2.2	1/OK
Coverage & Success	2/OK

Test	UC3
TC3.1	1/OK
TC3.2	1/OK
Coverage & Success	2/OK

Test plan and unit test cases

Link to my code: <https://gitlab.lnu.se/1dv600/student/mr223jp/assignment-3/-/blob/master/PlayerTest.java>

```
1 package Assingment3;
2
3 import org.junit.Test;
4 import org.junit.jupiter.api.BeforeEach;
5 import static org.junit.Assert.assertEquals;
6 import static org.junit.jupiter.api.Assertions.*;
7
8 public class PlayerTest {
9     private Player testing = new Player();
10
11
12     @Test
13     public void setName() {
14         String expected = "Ali", expected2 = "Marcos", expected3= "John", expected4 = "Emilia", expected5 = "Lana";
15         String actual;
16
17         testing.setName(expected);
18         actual = testing.getName();
19         assertEquals(expected, actual);
20
21         testing.setName(expected2);
22         actual = testing.getName();
23         assertEquals(expected2, actual);
24
25         testing.setName(expected3);
26         actual = testing.getName();
27         assertEquals(expected3, actual);
28
29         testing.setName(expected4);
30         actual = testing.getName();
31         assertEquals(expected4, actual);
32
33         testing.setName(expected5);
34         actual = testing.getName();
35     }
36 }
```

```
PlayerTest.java x Player.java x GameMain.java x GameBrain.java x
33 testing.setName(expected5);
34 actual = testing.getName();
35 assertEquals(expected5, actual);
36
37 }
38
39 @Test
40 public void setAge() {
41     int expected = 20, expected2 = 30, expected3 = 40, expected4 = 50, expected5 = 60;
42
43     testing.setAge(expected);
44     int actual = testing.getAge();
45     assertEquals(actual, expected);
46
47     testing.setAge(expected2);
48     actual = testing.getAge();
49     assertEquals(actual, expected2);
50
51     testing.setAge(expected3);
52     actual = testing.getAge();
53     assertEquals(actual, expected3);
54
55     testing.setAge(expected4);
56     actual = testing.getAge();
57     assertEquals(actual, expected4);
58
59     testing.setAge(expected5);
60     actual = testing.getAge();
61     assertEquals(actual, expected5);
62
63 }
64
65 @Test
```

```
66 @Test
67 public void setDate() {
68     try {
69         int expected = 2020;
70         int expected2 = 11;
71         int expected3 = 06;
72         testing.setDate(expected, expected2, expected3);
73
74         int expectedYear = 1999;
75         int expectedMonth = 06;
76         int expectedDay = 01;
77         testing.setDate(expectedYear, expectedMonth, expectedDay);
78
79         int expectedYear1 = 2020;
80         int expectedMonth1 = 10;
81         int expectedDay1 = 15;
82         testing.setDate(expectedYear1, expectedMonth1, expectedDay1);
83
84         int expectedYear2 = 1985;
85         int expectedMonth2 = 05;
86         int expectedDay2 = 29;
87         testing.setDate(expectedYear2, expectedMonth2, expectedDay2);
88
89         int expectedYear3 = 2019;
90         int expectedMonth3 = 3;
91         int expectedDay3 = 25;
92         testing.setDate(expectedYear3, expectedMonth3, expectedDay3);
93
94     } catch (Exception e) {
95         System.out.println(e.getMessage());
96     }
97 }
98
99 go)
```



```
PlayerTest.java x Player.java x GameMain.java x GameBrain.java x
96
97     } catch (Exception e) {
98         System.out.println(e.getMessage());
99     }
100 }
101
102 @Test
103 public void setSurvey() {
104     String expected = "I love the game", expected2 = "It was hard for me ", expected3= "It was amazing, loved it",
105         expected4 = "Maybe its better to add some hints", expected5 = "Wow, it was absolutely a fun experience";
106
107     testing.setSurvey(expected);
108     String actual = testing.getSurvey();
109     assertEquals(expected, actual);
110
111     testing.setSurvey(expected2);
112     actual = testing.getSurvey();
113     assertEquals(expected2, actual);
114
115     testing.setSurvey(expected3);
116     actual = testing.getSurvey();
117     assertEquals(expected3, actual);
118
119     testing.setSurvey(expected4);
120     actual = testing.getSurvey();
121     assertEquals(expected4, actual);
122
123     testing.setSurvey(expected5);
124     actual = testing.getSurvey();
125     assertEquals(expected5, actual);
126 }
127
128 @Test
129 public void getName() {
130
131     String expected = "Johnny", expected2 = "Victor", expected3= "Bryan",expected4 = "John snow", expected5 = "Mother of dragons";
132
133     testing.setName(expected);
134     String actual = testing.getName();
135     assertEquals(expected, actual);
136
137     testing.setName(expected2);
138     actual = testing.getName();
139     assertEquals(expected2, actual);
140
141     testing.setName(expected3);
142     actual = testing.getName();
143     assertEquals(expected3, actual);
144
145     testing.setName(expected4);
146     actual = testing.getName();
147     assertEquals(expected4, actual);
148
149     testing.setName(expected5);
150     actual = testing.getName();
151     assertEquals(expected5, actual);
152 }
153
154 @Test
155 public void getAge() {
156     int expected = 12,expected2 = 35, expected3= 25,expected4 = 65, expected5 =90;
157
158     testing.setAge(expected);
159     int actual = testing.getAge();
160     assertEquals(expected, actual);
161
162     testing.setAge(expected2);
163     actual = testing.getAge();
164     assertEquals(expected2, actual);
165
166     testing.setAge(expected3);
167     actual = testing.getAge();
168     assertEquals(expected3, actual);
169
170     testing.setAge(expected4);
171     actual = testing.getAge();
172     assertEquals(expected4, actual);
173
174     testing.setAge(expected5);
175     actual = testing.getAge();
176     assertEquals(expected5, actual);
177 }
178
179 @Test
180 public void getGame() {
181     String expected = "Game of Thrones", expected2 = "Game of Thrones", expected3= "Game of Thrones",
182         expected4 = "Game of Thrones", expected5 = "Game of Thrones";
183
184     testing.setGame(expected);
185     String actual = testing.getGame();
186     assertEquals(expected, actual);
187
188     testing.setGame(expected2);
189     actual = testing.getGame();
190     assertEquals(expected2, actual);
191
192     testing.setGame(expected3);
193     actual = testing.getGame();
194     assertEquals(expected3, actual);
195
196     testing.setGame(expected4);
197     actual = testing.getGame();
198     assertEquals(expected4, actual);
199
200     testing.setGame(expected5);
201     actual = testing.getGame();
202     assertEquals(expected5, actual);
203 }
204
205 @Test
206 public void getCharacter() {
207     String expected = "Jon Snow", expected2 = "Jon Snow", expected3= "Jon Snow",
208         expected4 = "Jon Snow", expected5 = "Jon Snow";
209
210     testing.setCharacter(expected);
211     String actual = testing.getCharacter();
212     assertEquals(expected, actual);
213
214     testing.setCharacter(expected2);
215     actual = testing.getCharacter();
216     assertEquals(expected2, actual);
217
218     testing.setCharacter(expected3);
219     actual = testing.getCharacter();
220     assertEquals(expected3, actual);
221
222     testing.setCharacter(expected4);
223     actual = testing.getCharacter();
224     assertEquals(expected4, actual);
225
226     testing.setCharacter(expected5);
227     actual = testing.getCharacter();
228     assertEquals(expected5, actual);
229 }
230
231 @Test
232 public void getWeapon() {
233     String expected = "Long sword", expected2 = "Long sword", expected3= "Long sword",
234         expected4 = "Long sword", expected5 = "Long sword";
235
236     testing.setWeapon(expected);
237     String actual = testing.getWeapon();
238     assertEquals(expected, actual);
239
240     testing.setWeapon(expected2);
241     actual = testing.getWeapon();
242     assertEquals(expected2, actual);
243
244     testing.setWeapon(expected3);
245     actual = testing.getWeapon();
246     assertEquals(expected3, actual);
247
248     testing.setWeapon(expected4);
249     actual = testing.getWeapon();
250     assertEquals(expected4, actual);
251
252     testing.setWeapon(expected5);
253     actual = testing.getWeapon();
254     assertEquals(expected5, actual);
255 }
256
257 @Test
258 public void getPower() {
259     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
260         expected4 = "Fire", expected5 = "Fire";
261
262     testing.setPower(expected);
263     String actual = testing.getPower();
264     assertEquals(expected, actual);
265
266     testing.setPower(expected2);
267     actual = testing.getPower();
268     assertEquals(expected2, actual);
269
270     testing.setPower(expected3);
271     actual = testing.getPower();
272     assertEquals(expected3, actual);
273
274     testing.setPower(expected4);
275     actual = testing.getPower();
276     assertEquals(expected4, actual);
277
278     testing.setPower(expected5);
279     actual = testing.getPower();
280     assertEquals(expected5, actual);
281 }
282
283 @Test
284 public void getSkill() {
285     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
286         expected4 = "Fire", expected5 = "Fire";
287
288     testing.setSkill(expected);
289     String actual = testing.getSkill();
290     assertEquals(expected, actual);
291
292     testing.setSkill(expected2);
293     actual = testing.getSkill();
294     assertEquals(expected2, actual);
295
296     testing.setSkill(expected3);
297     actual = testing.getSkill();
298     assertEquals(expected3, actual);
299
300     testing.setSkill(expected4);
301     actual = testing.getSkill();
302     assertEquals(expected4, actual);
303
304     testing.setSkill(expected5);
305     actual = testing.getSkill();
306     assertEquals(expected5, actual);
307 }
308
309 @Test
310 public void getDefence() {
311     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
312         expected4 = "Fire", expected5 = "Fire";
313
314     testing.setDefence(expected);
315     String actual = testing.getDefence();
316     assertEquals(expected, actual);
317
318     testing.setDefence(expected2);
319     actual = testing.getDefence();
320     assertEquals(expected2, actual);
321
322     testing.setDefence(expected3);
323     actual = testing.getDefence();
324     assertEquals(expected3, actual);
325
326     testing.setDefence(expected4);
327     actual = testing.getDefence();
328     assertEquals(expected4, actual);
329
330     testing.setDefence(expected5);
331     actual = testing.getDefence();
332     assertEquals(expected5, actual);
333 }
334
335 @Test
336 public void getAttack() {
337     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
338         expected4 = "Fire", expected5 = "Fire";
339
340     testing.setAttack(expected);
341     String actual = testing.getAttack();
342     assertEquals(expected, actual);
343
344     testing.setAttack(expected2);
345     actual = testing.getAttack();
346     assertEquals(expected2, actual);
347
348     testing.setAttack(expected3);
349     actual = testing.getAttack();
350     assertEquals(expected3, actual);
351
352     testing.setAttack(expected4);
353     actual = testing.getAttack();
354     assertEquals(expected4, actual);
355
356     testing.setAttack(expected5);
357     actual = testing.getAttack();
358     assertEquals(expected5, actual);
359 }
360
361 @Test
362 public void getHealth() {
363     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
364         expected4 = "Fire", expected5 = "Fire";
365
366     testing.setHealth(expected);
367     String actual = testing.getHealth();
368     assertEquals(expected, actual);
369
370     testing.setHealth(expected2);
371     actual = testing.getHealth();
372     assertEquals(expected2, actual);
373
374     testing.setHealth(expected3);
375     actual = testing.getHealth();
376     assertEquals(expected3, actual);
377
378     testing.setHealth(expected4);
379     actual = testing.getHealth();
380     assertEquals(expected4, actual);
381
382     testing.setHealth(expected5);
383     actual = testing.getHealth();
384     assertEquals(expected5, actual);
385 }
386
387 @Test
388 public void getStrength() {
389     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
390         expected4 = "Fire", expected5 = "Fire";
391
392     testing.setStrength(expected);
393     String actual = testing.getStrength();
394     assertEquals(expected, actual);
395
396     testing.setStrength(expected2);
397     actual = testing.getStrength();
398     assertEquals(expected2, actual);
399
400     testing.setStrength(expected3);
401     actual = testing.getStrength();
402     assertEquals(expected3, actual);
403
404     testing.setStrength(expected4);
405     actual = testing.getStrength();
406     assertEquals(expected4, actual);
407
408     testing.setStrength(expected5);
409     actual = testing.getStrength();
410     assertEquals(expected5, actual);
411 }
412
413 @Test
414 public void getSpeed() {
415     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
416         expected4 = "Fire", expected5 = "Fire";
417
418     testing.setSpeed(expected);
419     String actual = testing.getSpeed();
420     assertEquals(expected, actual);
421
422     testing.setSpeed(expected2);
423     actual = testing.getSpeed();
424     assertEquals(expected2, actual);
425
426     testing.setSpeed(expected3);
427     actual = testing.getSpeed();
428     assertEquals(expected3, actual);
429
430     testing.setSpeed(expected4);
431     actual = testing.getSpeed();
432     assertEquals(expected4, actual);
433
434     testing.setSpeed(expected5);
435     actual = testing.getSpeed();
436     assertEquals(expected5, actual);
437 }
438
439 @Test
440 public void getAgility() {
441     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
442         expected4 = "Fire", expected5 = "Fire";
443
444     testing.setAgility(expected);
445     String actual = testing.getAgility();
446     assertEquals(expected, actual);
447
448     testing.setAgility(expected2);
449     actual = testing.getAgility();
450     assertEquals(expected2, actual);
451
452     testing.setAgility(expected3);
453     actual = testing.getAgility();
454     assertEquals(expected3, actual);
455
456     testing.setAgility(expected4);
457     actual = testing.getAgility();
458     assertEquals(expected4, actual);
459
460     testing.setAgility(expected5);
461     actual = testing.getAgility();
462     assertEquals(expected5, actual);
463 }
464
465 @Test
466 public void getEndurance() {
467     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
468         expected4 = "Fire", expected5 = "Fire";
469
470     testing.setEndurance(expected);
471     String actual = testing.getEndurance();
472     assertEquals(expected, actual);
473
474     testing.setEndurance(expected2);
475     actual = testing.getEndurance();
476     assertEquals(expected2, actual);
477
478     testing.setEndurance(expected3);
479     actual = testing.getEndurance();
480     assertEquals(expected3, actual);
481
482     testing.setEndurance(expected4);
483     actual = testing.getEndurance();
484     assertEquals(expected4, actual);
485
486     testing.setEndurance(expected5);
487     actual = testing.getEndurance();
488     assertEquals(expected5, actual);
489 }
490
491 @Test
492 public void getIntelligence() {
493     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
494         expected4 = "Fire", expected5 = "Fire";
495
496     testing.setIntelligence(expected);
497     String actual = testing.getIntelligence();
498     assertEquals(expected, actual);
499
500     testing.setIntelligence(expected2);
501     actual = testing.getIntelligence();
502     assertEquals(expected2, actual);
503
504     testing.setIntelligence(expected3);
505     actual = testing.getIntelligence();
506     assertEquals(expected3, actual);
507
508     testing.setIntelligence(expected4);
509     actual = testing.getIntelligence();
510     assertEquals(expected4, actual);
511
512     testing.setIntelligence(expected5);
513     actual = testing.getIntelligence();
514     assertEquals(expected5, actual);
515 }
516
517 @Test
518 public void getWisdom() {
519     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
520         expected4 = "Fire", expected5 = "Fire";
521
522     testing.setWisdom(expected);
523     String actual = testing.getWisdom();
524     assertEquals(expected, actual);
525
526     testing.setWisdom(expected2);
527     actual = testing.getWisdom();
528     assertEquals(expected2, actual);
529
530     testing.setWisdom(expected3);
531     actual = testing.getWisdom();
532     assertEquals(expected3, actual);
533
534     testing.setWisdom(expected4);
535     actual = testing.getWisdom();
536     assertEquals(expected4, actual);
537
538     testing.setWisdom(expected5);
539     actual = testing.getWisdom();
540     assertEquals(expected5, actual);
541 }
542
543 @Test
544 public void getCourage() {
545     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
546         expected4 = "Fire", expected5 = "Fire";
547
548     testing.setCourage(expected);
549     String actual = testing.getCourage();
550     assertEquals(expected, actual);
551
552     testing.setCourage(expected2);
553     actual = testing.getCourage();
554     assertEquals(expected2, actual);
555
556     testing.setCourage(expected3);
557     actual = testing.getCourage();
558     assertEquals(expected3, actual);
559
560     testing.setCourage(expected4);
561     actual = testing.getCourage();
562     assertEquals(expected4, actual);
563
564     testing.setCourage(expected5);
565     actual = testing.getCourage();
566     assertEquals(expected5, actual);
567 }
568
569 @Test
570 public void getKindness() {
571     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
572         expected4 = "Fire", expected5 = "Fire";
573
574     testing.setKindness(expected);
575     String actual = testing.getKindness();
576     assertEquals(expected, actual);
577
578     testing.setKindness(expected2);
579     actual = testing.getKindness();
580     assertEquals(expected2, actual);
581
582     testing.setKindness(expected3);
583     actual = testing.getKindness();
584     assertEquals(expected3, actual);
585
586     testing.setKindness(expected4);
587     actual = testing.getKindness();
588     assertEquals(expected4, actual);
589
590     testing.setKindness(expected5);
591     actual = testing.getKindness();
592     assertEquals(expected5, actual);
593 }
594
595 @Test
596 public void getGenerosity() {
597     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
598         expected4 = "Fire", expected5 = "Fire";
599
600     testing.setGenerosity(expected);
601     String actual = testing.getGenerosity();
602     assertEquals(expected, actual);
603
604     testing.setGenerosity(expected2);
605     actual = testing.getGenerosity();
606     assertEquals(expected2, actual);
607
608     testing.setGenerosity(expected3);
609     actual = testing.getGenerosity();
610     assertEquals(expected3, actual);
611
612     testing.setGenerosity(expected4);
613     actual = testing.getGenerosity();
614     assertEquals(expected4, actual);
615
616     testing.setGenerosity(expected5);
617     actual = testing.getGenerosity();
618     assertEquals(expected5, actual);
619 }
620
621 @Test
622 public void getHonesty() {
623     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
624         expected4 = "Fire", expected5 = "Fire";
625
626     testing.setHonesty(expected);
627     String actual = testing.getHonesty();
628     assertEquals(expected, actual);
629
630     testing.setHonesty(expected2);
631     actual = testing.getHonesty();
632     assertEquals(expected2, actual);
633
634     testing.setHonesty(expected3);
635     actual = testing.getHonesty();
636     assertEquals(expected3, actual);
637
638     testing.setHonesty(expected4);
639     actual = testing.getHonesty();
640     assertEquals(expected4, actual);
641
642     testing.setHonesty(expected5);
643     actual = testing.getHonesty();
644     assertEquals(expected5, actual);
645 }
646
647 @Test
648 public void getModesty() {
649     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
650         expected4 = "Fire", expected5 = "Fire";
651
652     testing.setModesty(expected);
653     String actual = testing.getModesty();
654     assertEquals(expected, actual);
655
656     testing.setModesty(expected2);
657     actual = testing.getModesty();
658     assertEquals(expected2, actual);
659
660     testing.setModesty(expected3);
661     actual = testing.getModesty();
662     assertEquals(expected3, actual);
663
664     testing.setModesty(expected4);
665     actual = testing.getModesty();
666     assertEquals(expected4, actual);
667
668     testing.setModesty(expected5);
669     actual = testing.getModesty();
670     assertEquals(expected5, actual);
671 }
672
673 @Test
674 public void getHumility() {
675     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
676         expected4 = "Fire", expected5 = "Fire";
677
678     testing.setHumility(expected);
679     String actual = testing.getHumility();
680     assertEquals(expected, actual);
681
682     testing.setHumility(expected2);
683     actual = testing.getHumility();
684     assertEquals(expected2, actual);
685
686     testing.setHumility(expected3);
687     actual = testing.getHumility();
688     assertEquals(expected3, actual);
689
690     testing.setHumility(expected4);
691     actual = testing.getHumility();
692     assertEquals(expected4, actual);
693
694     testing.setHumility(expected5);
695     actual = testing.getHumility();
696     assertEquals(expected5, actual);
697 }
698
699 @Test
700 public void getPatience() {
701     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
702         expected4 = "Fire", expected5 = "Fire";
703
704     testing.setPatience(expected);
705     String actual = testing.getPatience();
706     assertEquals(expected, actual);
707
708     testing.setPatience(expected2);
709     actual = testing.getPatience();
710     assertEquals(expected2, actual);
711
712     testing.setPatience(expected3);
713     actual = testing.getPatience();
714     assertEquals(expected3, actual);
715
716     testing.setPatience(expected4);
717     actual = testing.getPatience();
718     assertEquals(expected4, actual);
719
720     testing.setPatience(expected5);
721     actual = testing.getPatience();
722     assertEquals(expected5, actual);
723 }
724
725 @Test
726 public void getCompassion() {
727     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
728         expected4 = "Fire", expected5 = "Fire";
729
730     testing.setCompassion(expected);
731     String actual = testing.getCompassion();
732     assertEquals(expected, actual);
733
734     testing.setCompassion(expected2);
735     actual = testing.getCompassion();
736     assertEquals(expected2, actual);
737
738     testing.setCompassion(expected3);
739     actual = testing.getCompassion();
740     assertEquals(expected3, actual);
741
742     testing.setCompassion(expected4);
743     actual = testing.getCompassion();
744     assertEquals(expected4, actual);
745
746     testing.setCompassion(expected5);
747     actual = testing.getCompassion();
748     assertEquals(expected5, actual);
749 }
750
751 @Test
752 public void getGratitude() {
753     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
754         expected4 = "Fire", expected5 = "Fire";
755
756     testing.setGratitude(expected);
757     String actual = testing.getGratitude();
758     assertEquals(expected, actual);
759
760     testing.setGratitude(expected2);
761     actual = testing.getGratitude();
762     assertEquals(expected2, actual);
763
764     testing.setGratitude(expected3);
765     actual = testing.getGratitude();
766     assertEquals(expected3, actual);
767
768     testing.setGratitude(expected4);
769     actual = testing.getGratitude();
770     assertEquals(expected4, actual);
771
772     testing.setGratitude(expected5);
773     actual = testing.getGratitude();
774     assertEquals(expected5, actual);
775 }
776
777 @Test
778 public void getForgiveness() {
779     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
780         expected4 = "Fire", expected5 = "Fire";
781
782     testing.setForgiveness(expected);
783     String actual = testing.getForgiveness();
784     assertEquals(expected, actual);
785
786     testing.setForgiveness(expected2);
787     actual = testing.getForgiveness();
788     assertEquals(expected2, actual);
789
790     testing.setForgiveness(expected3);
791     actual = testing.getForgiveness();
792     assertEquals(expected3, actual);
793
794     testing.setForgiveness(expected4);
795     actual = testing.getForgiveness();
796     assertEquals(expected4, actual);
797
798     testing.setForgiveness(expected5);
799     actual = testing.getForgiveness();
800     assertEquals(expected5, actual);
801 }
802
803 @Test
804 public void getGenerosity() {
805     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
806         expected4 = "Fire", expected5 = "Fire";
807
808     testing.setGenerosity(expected);
809     String actual = testing.getGenerosity();
810     assertEquals(expected, actual);
811
812     testing.setGenerosity(expected2);
813     actual = testing.getGenerosity();
814     assertEquals(expected2, actual);
815
816     testing.setGenerosity(expected3);
817     actual = testing.getGenerosity();
818     assertEquals(expected3, actual);
819
820     testing.setGenerosity(expected4);
821     actual = testing.getGenerosity();
822     assertEquals(expected4, actual);
823
824     testing.setGenerosity(expected5);
825     actual = testing.getGenerosity();
826     assertEquals(expected5, actual);
827 }
828
829 @Test
830 public void getHonesty() {
831     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
832         expected4 = "Fire", expected5 = "Fire";
833
834     testing.setHonesty(expected);
835     String actual = testing.getHonesty();
836     assertEquals(expected, actual);
837
838     testing.setHonesty(expected2);
839     actual = testing.getHonesty();
840     assertEquals(expected2, actual);
841
842     testing.setHonesty(expected3);
843     actual = testing.getHonesty();
844     assertEquals(expected3, actual);
845
846     testing.setHonesty(expected4);
847     actual = testing.getHonesty();
848     assertEquals(expected4, actual);
849
850     testing.setHonesty(expected5);
851     actual = testing.getHonesty();
852     assertEquals(expected5, actual);
853 }
854
855 @Test
856 public void getModesty() {
857     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
858         expected4 = "Fire", expected5 = "Fire";
859
860     testing.setModesty(expected);
861     String actual = testing.getModesty();
862     assertEquals(expected, actual);
863
864     testing.setModesty(expected2);
865     actual = testing.getModesty();
866     assertEquals(expected2, actual);
867
868     testing.setModesty(expected3);
869     actual = testing.getModesty();
870     assertEquals(expected3, actual);
871
872     testing.setModesty(expected4);
873     actual = testing.getModesty();
874     assertEquals(expected4, actual);
875
876     testing.setModesty(expected5);
877     actual = testing.getModesty();
878     assertEquals(expected5, actual);
879 }
880
881 @Test
882 public void getHumility() {
883     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
884         expected4 = "Fire", expected5 = "Fire";
885
886     testing.setHumility(expected);
887     String actual = testing.getHumility();
888     assertEquals(expected, actual);
889
890     testing.setHumility(expected2);
891     actual = testing.getHumility();
892     assertEquals(expected2, actual);
893
894     testing.setHumility(expected3);
895     actual = testing.getHumility();
896     assertEquals(expected3, actual);
897
898     testing.setHumility(expected4);
899     actual = testing.getHumility();
900     assertEquals(expected4, actual);
901
902     testing.setHumility(expected5);
903     actual = testing.getHumility();
904     assertEquals(expected5, actual);
905 }
906
907 @Test
908 public void getPatience() {
909     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
910         expected4 = "Fire", expected5 = "Fire";
911
912     testing.setPatience(expected);
913     String actual = testing.getPatience();
914     assertEquals(expected, actual);
915
916     testing.setPatience(expected2);
917     actual = testing.getPatience();
918     assertEquals(expected2, actual);
919
920     testing.setPatience(expected3);
921     actual = testing.getPatience();
922     assertEquals(expected3, actual);
923
924     testing.setPatience(expected4);
925     actual = testing.getPatience();
926     assertEquals(expected4, actual);
927
928     testing.setPatience(expected5);
929     actual = testing.getPatience();
930     assertEquals(expected5, actual);
931 }
932
933 @Test
934 public void getCompassion() {
935     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
936         expected4 = "Fire", expected5 = "Fire";
937
938     testing.setCompassion(expected);
939     String actual = testing.getCompassion();
940     assertEquals(expected, actual);
941
942     testing.setCompassion(expected2);
943     actual = testing.getCompassion();
944     assertEquals(expected2, actual);
945
946     testing.setCompassion(expected3);
947     actual = testing.getCompassion();
948     assertEquals(expected3, actual);
949
950     testing.setCompassion(expected4);
951     actual = testing.getCompassion();
952     assertEquals(expected4, actual);
953
954     testing.setCompassion(expected5);
955     actual = testing.getCompassion();
956     assertEquals(expected5, actual);
957 }
958
959 @Test
960 public void getGratitude() {
961     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
962         expected4 = "Fire", expected5 = "Fire";
963
964     testing.setGratitude(expected);
965     String actual = testing.getGratitude();
966     assertEquals(expected, actual);
967
968     testing.setGratitude(expected2);
969     actual = testing.getGratitude();
970     assertEquals(expected2, actual);
971
972     testing.setGratitude(expected3);
973     actual = testing.getGratitude();
974     assertEquals(expected3, actual);
975
976     testing.setGratitude(expected4);
977     actual = testing.getGratitude();
978     assertEquals(expected4, actual);
979
980     testing.setGratitude(expected5);
981     actual = testing.getGratitude();
982     assertEquals(expected5, actual);
983 }
984
985 @Test
986 public void getForgiveness() {
987     String expected = "Fire", expected2 = "Fire", expected3= "Fire",
988         expected4 = "Fire", expected5 = "Fire";
989
990     testing.setForgiveness(expected);
991     String actual = testing.getForgiveness();
992     assertEquals(expected, actual);
993
994     testing.setForgiveness(expected2);
995     actual = testing.getForgiveness();
996     assertEquals(expected2, actual);
997
998     testing.setForgiveness(expected3);
999     actual = testing.getForgiveness();
1000    assertEquals(expected3, actual);
1001
1002    testing.setForgiveness(expected4);
1003    actual = testing.getForgiveness();
1004    assertEquals(expected4, actual);
1005
1006    testing.setForgiveness(expected5);
1007    actual = testing.getForgiveness();
1008    assertEquals(expected5, actual);
1009
1010    testing.setForgiveness(expected6);
1011    actual = testing.getForgiveness();
1012    assertEquals(expected6, actual);
1013
1014    testing.setForgiveness(expected7);
1015    actual = testing.getForgiveness();
1016    assertEquals(expected7, actual);
1017
1018    testing.setForgiveness(expected8);
1019    actual = testing.getForgiveness();
1020    assertEquals(expected8, actual);
1021
1022    testing.setForgiveness(expected9);
1023    actual = testing.getForgiveness();
1024    assertEquals(expected9, actual);
1025
1026    testing.setForgiveness(expected10);
1027    actual = testing.getForgiveness();
1028    assertEquals(expected10, actual);
1029
1030    testing.setForgiveness(expected11);
1031    actual = testing.getForgiveness();
1032    assertEquals(expected11, actual);
1033
1034    testing.setForgiveness(expected12);
1035    actual = testing.getForgiveness();
1036    assertEquals(expected12, actual);
1037
1038    testing.setForgiveness(expected13);
1039    actual = testing.getForgiveness();
1040    assertEquals(expected13, actual);
1041
1042    testing.setForgiveness(expected14);
1043    actual = testing.getForgiveness();
1044    assertEquals(expected14, actual);
1045
1046    testing.setForgiveness(expected15);
1047    actual = testing.getForgiveness();
1048    assertEquals(expected15, actual);
1049
1050    testing.setForgiveness(expected16);
1051    actual = testing.getForgiveness();
1052    assertEquals(expected16, actual);
1053
1054    testing.setForgiveness(expected17);
1055    actual = testing.getForgiveness();
1056    assertEquals(expected17, actual);
1057
1058    testing.setForgiveness(expected18);
1059    actual = testing.getForgiveness();
1060    assertEquals(expected18, actual);
1061
1062    testing.setForgiveness(expected19);
1063    actual = testing.getForgiveness();
1064    assertEquals(expected19, actual);
1065
1066    testing.setForgiveness(expected20);
1067    actual = testing.getForgiveness();
1068    assertEquals(expected20, actual);
1069
1070    testing.setForgiveness(expected21);
1071    actual = testing.getForgiveness();
1072    assertEquals(expected21, actual);
1073
1074    testing.setForgiveness(expected22);
1075    actual = testing.getForgiveness();
1076    assertEquals(expected22, actual);
1077
1078    testing.setForgiveness(expected23);
1079    actual = testing.getForgiveness();
1080    assertEquals(expected23, actual);
1081
1082    testing.setForgiveness(expected24);
1083    actual = testing.getForgiveness();
1084    assertEquals(expected24, actual);
1085
1086    testing.setForgiveness(expected25);
1087    actual = testing.getForgiveness();
1088    assertEquals(expected25, actual);
1089
1090    testing.setForgiveness(expected26);
1091    actual = testing.getForgiveness();
1092    assertEquals(expected26, actual);
1093
1094    testing.setForgiveness(expected27);
1095    actual = testing.getForgiveness();
1096    assertEquals(expected27, actual);
1097
1098    testing.setForgiveness(expected28);
1099    actual = testing.getForgiveness();
1100    assertEquals(expected28, actual);
1101
1102    testing.setForgiveness(expected29);
1103    actual = testing.getForgiveness();
1104    assertEquals(expected29, actual);
1105
1106    testing.setForgiveness(expected30);
1107    actual = testing.getForgiveness();
1108    assertEquals(expected30, actual);
1109
1110    testing.setForgiveness(expected31);
1111    actual = testing.getForgiveness();
1112    assertEquals(expected31, actual);
1113
1114    testing.setForgiveness(expected32);
1115    actual = testing.getForgiveness();
1116    assertEquals(expected32, actual);
1117
1118    testing.setForgiveness(expected33);
1119    actual = testing.getForgiveness();
1120    assertEquals(expected33, actual);
1121
1122    testing.setForgiveness(expected34);
1123    actual = testing.getForgiveness();
1124    assertEquals(expected34, actual);
1125
1126    testing.setForgiveness(expected35);
1127    actual = testing.getForgiveness();
1128    assertEquals(expected35, actual);
1129
1130    testing.setForgiveness(expected36);
1131    actual = testing.getForgiveness();
1132    assertEquals(expected36, actual);
1133
1134    testing.setForgiveness(expected37);
1135    actual = testing.getForgiveness();
1136    assertEquals(expected37, actual);
1137
1138    testing.setForgiveness(expected38);
1139    actual = testing.getForgiveness();
1140    assertEquals(expected38, actual);
1141
1142    testing.setForgiveness(expected39);
1143    actual = testing.getForgiveness();
1144    assertEquals(expected39, actual);
1145
1146    testing.setForgiveness(expected40);
1147    actual = testing.getForgiveness();
1148    assertEquals(expected40, actual);
1149
1150    testing.setForgiveness(expected41);
1151    actual = testing.getForgiveness();
1152    assertEquals(expected41, actual);
1153
1154    testing.setForgiveness(expected42);
1155    actual = testing.getForgiveness();
1156    assertEquals(expected42, actual);
1157
1158    testing.setForgiveness(expected43);
1159    actual = testing.getForgiveness();
1160    assertEquals(expected43, actual);
1161
1162    testing.setForgiveness(expected44);
1163    actual = testing.getForgiveness();
1164    assertEquals(expected44, actual);
1165
1166    testing.setForgiveness(expected45);
1167    actual = testing.getForgiveness();
1168    assertEquals(expected45, actual);
1169
1170    testing.setForgiveness(expected46);
1171    actual = testing.getForgiveness();
1172    assertEquals(expected46, actual);
1173
1174    testing.setForgiveness(expected47);
1175    actual = testing.getForgiveness();
1176    assertEquals(expected47, actual);
1177
1178    testing.setForgiveness(expected48);
1179    actual = testing.getForgiveness();
1180    assertEquals(expected48, actual);
1181
1182    testing.setForgiveness(expected49);
1183    actual = testing.getForgiveness();
1184    assertEquals(expected49, actual);
1185
1186    testing.setForgiveness(expected50);
1187    actual = testing.getForgiveness();
1188    assertEquals(expected50, actual);
1189
1190    testing.setForgiveness(expected51);
1191    actual = testing.getForgiveness();
1192    assertEquals(expected51, actual);
1193
1194    testing.setForgiveness(expected52);
1195    actual = testing.getForgiveness();
1196    assertEquals(expected52, actual);
1197
1198    testing.setForgiveness(expected53);
1199    actual = testing.getForgiveness();
1200    assertEquals(expected53, actual);
1201
1202    testing.setForgiveness(expected54);
1203    actual = testing.getForgiveness();
1204    assertEquals(expected54, actual);
1205
1206    testing.setForgiveness(expected55);
1207    actual = testing.getForgiveness();
1208    assertEquals(expected55, actual);
1209
1210    testing.setForgiveness(expected56);
1211    actual = testing.getForgiveness();
1212    assertEquals(expected56, actual);
1213
1214    testing.setForgiveness(expected57);
1215    actual = testing.getForgiveness();
1216    assertEquals(expected57, actual);
1217
1218    testing.setForgiveness(expected58);
1219    actual = testing.getForgiveness();
1220    assertEquals(expected58, actual);
1221
1222    testing.setForgiveness(expected59);
1223    actual = testing.getForgiveness();
1224    assertEquals(expected59, actual);
1225
1226    testing.setForgiveness(expected60);
1227    actual = testing.getForgiveness();
1228    assertEquals(expected60, actual);
1229
1230    testing.setForgiveness(expected61);
1231    actual = testing.getForgiveness();
1232    assertEquals(expected61, actual);
1233
1234    testing.setForgiveness(expected62);
1235    actual = testing.getForgiveness();
1236    assertEquals(expected62, actual);
1237
1238    testing.setForgiveness(expected63);
1239    actual = testing.getForgiveness();
1240    assertEquals(expected63, actual);
1241
1242    testing.setForgiveness(expected64);
1243    actual = testing.getForgiveness();
1244    assertEquals(expected64, actual);
1245
1246    testing.setForgiveness(expected65);
1247    actual = testing.getForgiveness();
1248    assertEquals(expected65, actual);
1249
1250    testing.setForgiveness(expected66);
1251    actual = testing.getForgiveness();
1252    assertEquals(expected66, actual);
1253
1254    testing.setForgiveness(expected67);
1255    actual = testing.getForgiveness();
1256    assertEquals(expected67, actual);
1257
1258    testing.setForgiveness(expected68);
1259    actual = testing.getForgiveness();
1260    assertEquals(expected68, actual);
1261
1262    testing.setForgiveness(expected69);
1263    actual = testing.getForgiveness();
1264    assertEquals(expected69, actual);
1265
1266    testing.setForgiveness(expected70);
1267    actual = testing.getForgiveness();
1268    assertEquals(expected70, actual);
1269
1270    testing.setForgiveness(expected71);
1271    actual = testing.getForgiveness();
1272    assertEquals(expected71, actual);
1273
1274    testing.setForgiveness(expected72);
1275    actual = testing.getForgiveness();
1276    assertEquals(expected72, actual);
1277
1278    testing.setForgiveness(expected73);
1279    actual = testing.getForgiveness();
1280    assertEquals(expected73, actual);
1281
1282    testing.setForgiveness(expected74);
1283    actual = testing.getForgiveness();
1284    assertEquals(expected74, actual);
1285
1286    testing.setForgiveness(expected75);
1287    actual = testing.getForgiveness();
1288    assertEquals(expected75, actual);
1289
1290    testing.setForgiveness(expected76);
1291    actual = testing.getForgiveness();
1292    assertEquals(expected76, actual);
1293
1294    testing.setForgiveness(expected77);
1295    actual = testing.getForgiveness();
1296    assertEquals(expected77, actual);
1297
1298    testing.setForgiveness(expected78);
1299    actual = testing.getForgiveness();
1300    assertEquals(expected78, actual);

```

```

151     }
152
153     @Test
154     public void getAge() {
155         int expected = 12, expected2 = 35, expected3 = 25, expected4 = 65, expected5 = 98;
156
157         testing.setAge(expected);
158         int actual = testing.getAge();
159         assertEquals(expected, actual);
160
161         testing.setAge(expected2);
162         actual = testing.getAge();
163         assertEquals(expected2, actual);
164
165         testing.setAge(expected3);
166         actual = testing.getAge();
167         assertEquals(expected3, actual);
168
169         testing.setAge(expected4);
170         actual = testing.getAge();
171         assertEquals(expected4, actual);
172
173         testing.setAge(expected5);
174         actual = testing.getAge();
175         assertEquals(expected5, actual);
176     }
177

```

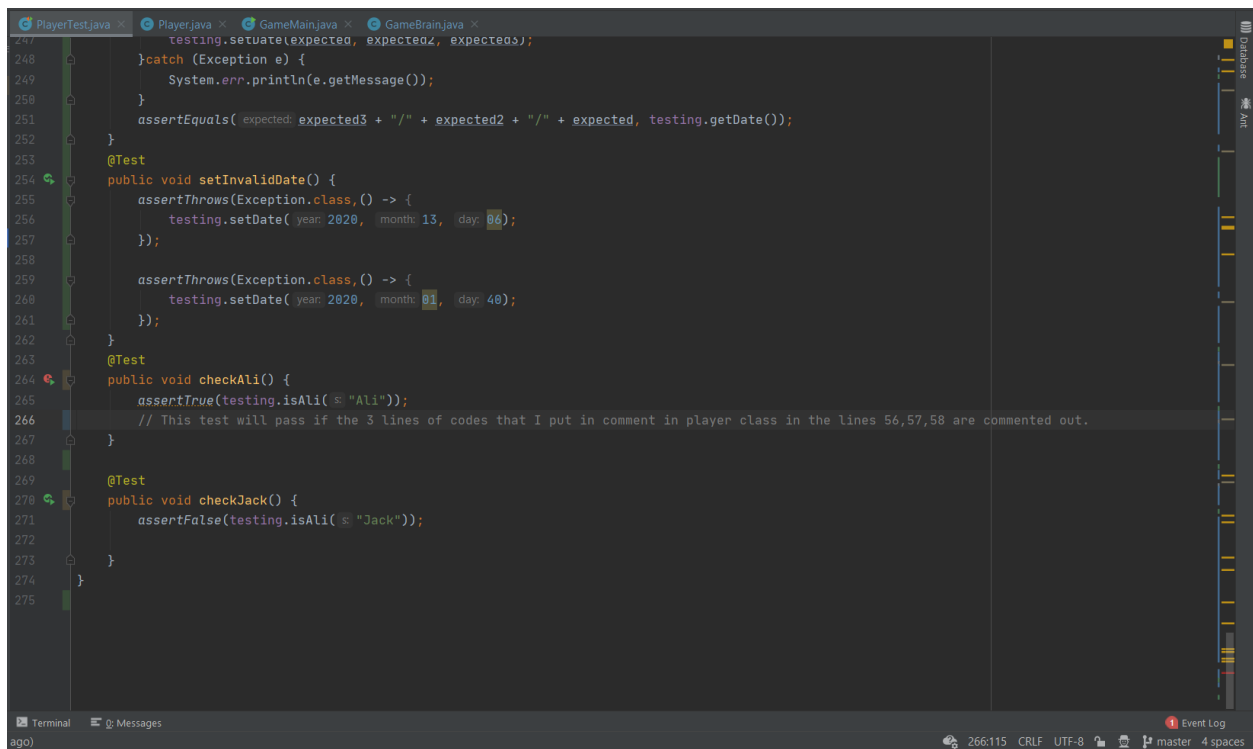
```

176     }
177
178     @Test
179     public void getSurvey() {
180         String expected = "was nice", expected2 = "SoSo", expected3 = "Such an amazing game",
181             expected4 = "Didn't like the game", expected5 = "Boring";
182
183
184         testing.setSurvey(expected);
185         String actual = testing.getSurvey();
186         assertEquals(actual, expected);
187
188         testing.setSurvey(expected2);
189         actual = testing.getSurvey();
190         assertEquals(actual, expected2);
191
192         testing.setSurvey(expected3);
193         actual = testing.getSurvey();
194         assertEquals(actual, expected3);
195     }
196

```

```
PlayerTest.java x Player.java x GameMain.java x GameBrain.java x
197 @Test
198 public void getDate() throws Exception {
199     try {
200
201         int year = 2020, month = 05, day = 06;
202         testing.setDate(year, month, day);
203         String actual = testing.getDate();
204         assertEquals( expected: day + "/" + month + "/" + year, actual);
205
206         int year1 = 2019, month1 = 10, day1 = 13;
207         testing.setDate(year1, month1, day1);
208         actual = testing.getDate();
209         assertEquals( expected: day1 + "/" + month1 + "/" + year1, actual);
210
211         int year2 = 2018, month2 = 12, day2 = 28;
212         testing.setDate(year2, month2, day2);
213         actual = testing.getDate();
214         assertEquals( expected: day2 + "/" + month2 + "/" + year2, actual);
215
216         int year3 = 1995, month3 = 03, day3 = 01;
217         testing.setDate(year3, month3, day3);
218         actual = testing.getDate();
219         assertEquals( expected: day3 + "/" + month3 + "/" + year3, actual);
220
221         int year4 = 1999, month4 = 07, day4 = 30;
222         testing.setDate(year4, month4, day4);
223         actual = testing.getDate();
224         assertEquals( expected: day4 + "/" + month4 + "/" + year4, actual);
225
226     }catch (Exception e){
227         System.out.println(e.getMessage());
228     }
229 }
230
Terminal Messages
ago) 266:115 CRLF UTF-8 master 4 spaces
```

```
PlayerTest.java x Player.java x GameMain.java x GameBrain.java x
230
231 @Test
232 public void setValidDate() {
233     int expected = 2020;
234     int expected2 = 11;
235     int expected3 = 06;
236     try {
237         testing.setDate(expected, expected2, expected3);
238     }catch (Exception e) {
239         System.err.println(e.getMessage());
240     }
241     assertEquals( expected: expected3 + "/" + expected2 + "/" + expected, testing.getDate());
242
243     expected = 1378;
244     expected2 = 01;
245     expected3 = 4;
246     try {
247         testing.setDate(expected, expected2, expected3);
248     }catch (Exception e) {
249         System.err.println(e.getMessage());
250     }
251     assertEquals( expected: expected3 + "/" + expected2 + "/" + expected, testing.getDate());
252 }
253 @Test
254 public void setInvalidDate() {
255     assertThrows(Exception.class, () -> {
256         testing.setDate( year: 2020, month: 13, day: 06);
257     });
258
259     assertThrows(Exception.class, () -> {
260         testing.setDate( year: 2020, month: 01, day: 40);
261     });
262 }
263 @Test
Terminal Messages
ago) 266:115 CRLF UTF-8 master 4 spaces
```

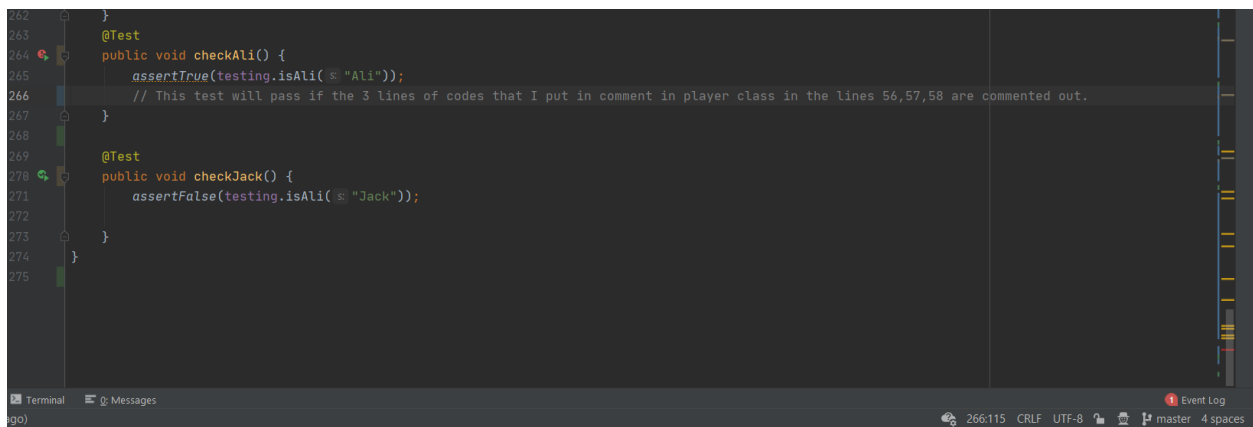


```
247 testing.setUate(expected, expected2, expected3);
248 }catch (Exception e) {
249     System.err.println(e.getMessage());
250 }
251 assertEquals( expected: expected3 + "/" + expected2 + "/" + expected, testing.getDate());
252 }
253 @Test
254 public void setInvalidDate() {
255     assertThrows(Exception.class,() -> {
256         testing.setDate( year: 2020, month: 13, day: 06);
257     });
258
259     assertThrows(Exception.class,() -> {
260         testing.setDate( year: 2020, month: 01, day: 40);
261     });
262 }
263 @Test
264 public void checkAli() {
265     assertTrue(testing.isAli( "Ali"));
266     // This test will pass if the 3 lines of codes that I put in comment in player class in the lines 56,57,58 are commented out.
267 }
268
269 @Test
270 public void checkJack() {
271     assertFalse(testing.isAli( "Jack"));
272 }
273 }
274 }
275 }
```

Automated unit test code screenshots

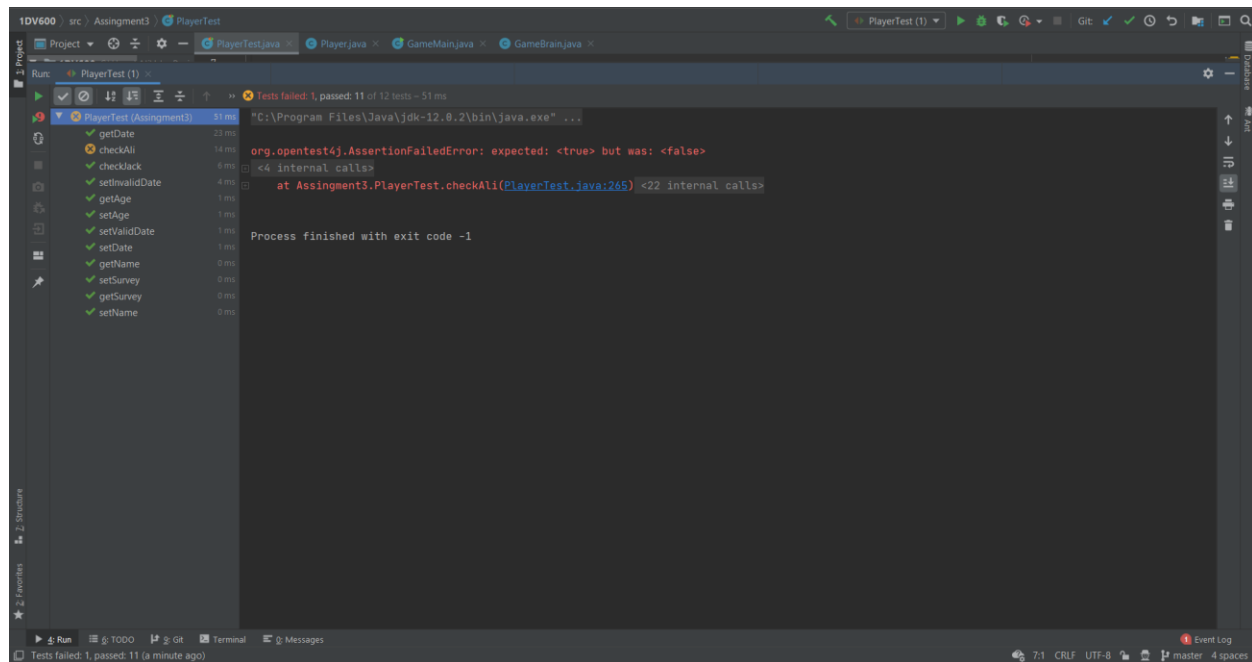
The below pictures illustrate two test codes like each other but one of them will pass and the other will fail when executing at the same time.

The test method called checkAli(), this Test can pass if the codes in lines 56,57,58 in class player are commented out.



```
262 }
263 @Test
264 public void checkAli() {
265     assertTrue(testing.isAli( "Ali"));
266     // This test will pass if the 3 lines of codes that I put in comment in player class in the lines 56,57,58 are commented out.
267 }
268
269 @Test
270 public void checkJack() {
271     assertFalse(testing.isAli( "Jack"));
272 }
273 }
274 }
275 }
```

As I have mentioned before the Test checkAli() will fail and the Test checkJack() will pass.



Comments

All the tests passed, but more test for data user case can be written to test if it works or not. However, two methods called checkAli() and checkJack() are the prominent examples that will demonstrate one test will fail on purpose and the other will pass (checkJack).

Reflection

In this iteration, 3 use cases have been used to write a manual test case, the results were promising since the tests passed. However, more use cases can be written to make sure if they can pass or fail. An example would be writing a use case along with 2 test cases for the data stage in my code.

Regarding the automated test case in my program, there are 10 tests in total which all will pass except one of them (fail on purpose).

A noticeable point about the test that fails (checkAli) is that there is another test called checkJack() which is similar to the checkAli() test, but one will fail and the other will pass.

I have spent more than four hours to come up with the test that fails on purpose. In the beginning, my idea about this test was to make some changes to the date method in the player. But I came with the idea of creating two methods that fail and pass at the same time. Consequently, I spent more time on creating these two methods.

Thanks to this assignment I have improved my skills at writing tests in junit and from this assignment, a developer can make sure either the methods are going to work or not. But it would be better there was more time to spend on this project. Therefore, more precise tests would have been written.