

# گزارش کار آزمایش ششم آزمایشگاه مهندسی نرم افزار

لینک آزمایش: [https://github.com/yasmansh/SE\\_LAB/tree/main/Exp6](https://github.com/yasmansh/SE_LAB/tree/main/Exp6)

اعضای گروه:

یاسمن شیخان 97101915

امیرحسین علی محمدی 97110166

## 1. پروژه json-simple

۱.۱ - ابتدا پروژه را در محیط IntelliJ باز کرده و چون JUnit و SDK از قبل اضافه و نصب شده اند کلاس ها شناسایی میشوند.

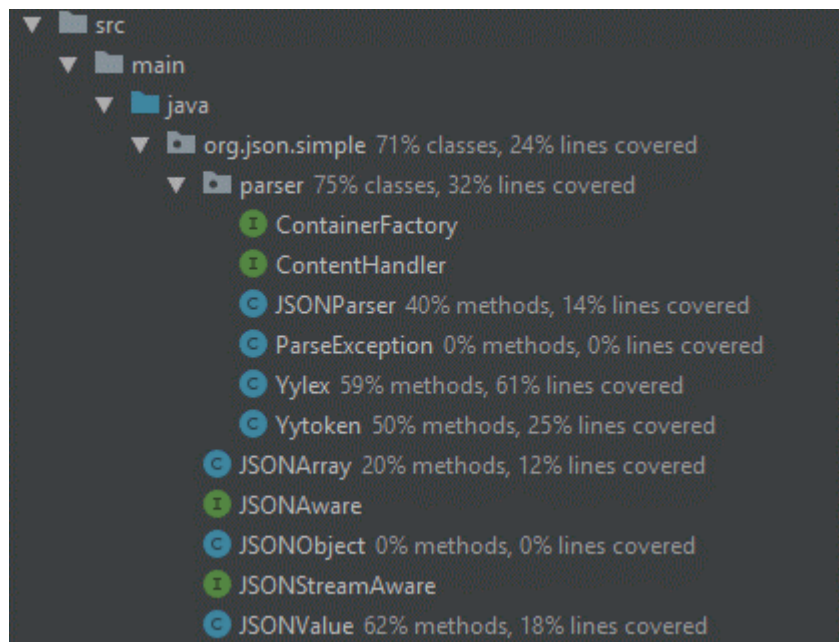
۱.۲ - سپس با کلیک بر روی کلاس TestJson گزینه run TestJson with coverage را انتخاب میکنیم.

درصد کلاس ها ، متدها و خط های برنامه که مورد تست قرار میگیرند را در خروجی مشاهده میکنیم:

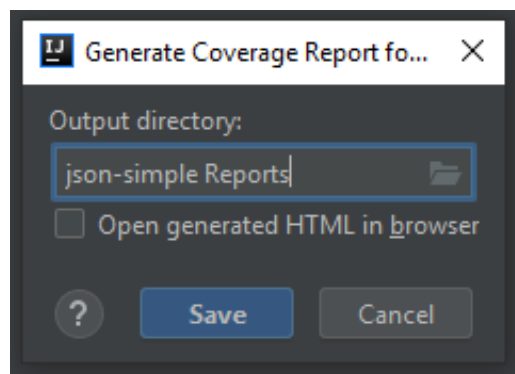
Coverage: TestJson ×			
71% classes, 24% lines covered in 'all classes in scope'			
Element	Class, %	Method, %	Line, %
org.json.simple	71% (5/7)	32% (30/91)	24% (215/861)

Coverage: TestJson ×			
71% classes, 24% lines covered in package 'org.json.simple'			
Element	Class, %	Method, %	Line, %
parser	75% (3/4)	40% (20/49)	32% (172/531)
JSONArray	100% (1/1)	20% (5/25)	12% (22/175)
JSONObject	0% (0/1)	0% (0/9)	0% (0/41)
JSONValue	100% (1/1)	62% (5/8)	18% (21/114)

TestJson (org.json.simple)	119 ms	"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" -ea -javaagent:C:\Users\ASuS\IntelliJ IDEA201
testJSONArrayCollection	33 ms	---- IntelliJ IDEA coverage runner ----
testDecode	86 ms	sampling ...
		include patterns:
		org.json.simple\.*
		exclude patterns:=====decode=====
		=====the 2nd element of array=====
		10
		Class transformation time: 0.1162815s for 462 classes or 2.516915584415584E-4s per class
		Process finished with exit code 0



سپس گزینه **Generate coverage report** را انتخاب و گزارش به فرمت **html** را دریافت میکنیم.



[ all classes ]

#### Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/ 7)	33.3% (31/ 93)	25.1% (215/ 857)

#### Coverage Breakdown

Package ^	Class, %	Method, %	Line, %
org.json.simple	66.7% (2/ 3)	23.3% (10/ 43)	13.2% (43/ 326)
org.json.simple.parser	75% (3/ 4)	42% (21/ 50)	32.4% (172/ 531)

generated on 2022-12-15 01:07

*index.html*

```

47  */
48  public static void writeJSONString(Collection collection, Writer out) throws IOE
49      if(collection == null){
50          out.write("null");
51          return;
52      }
53
54      boolean first = true;
55      Iterator iter=collection.iterator();
56
57      out.write('[');
58      while(iter.hasNext()){
59          if(first)
60              first = false;
61          else
62              out.write(',');
63
64          Object value=iter.next();
65          if(value == null){
66              out.write("null");
67              continue;
68          }
69
70          JSONValue.writeJSONString(value, out);
71      }
72      out.write(']');
73  }
74
75  public void writeJSONString(Writer out) throws IOException{
76      writeJSONString(this, out);
77  }
78
79  /**
80   * Convert a list to JSON text. The result is a JSON array.

```

مثالی از خطوط پوشش داده شده (به رنگ سبز) و نشده (به رنگ قرمز)

## تمرین (CodeCoverageProject)

1. مشابه بخش قبل، ابتدا ابتدا پروژه را در محیط IntelliJ باز کرده و چون JUnit و SDK

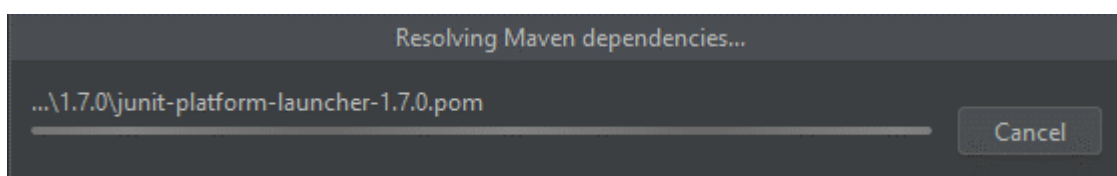
از قبل اضافه و نصب شده اند کلاس ها شناسایی میشوند اما برخی متدها شناسایی

نمیشوند که تمامی آنها را نیز دانلود / ایمپورت میکنیم.

```

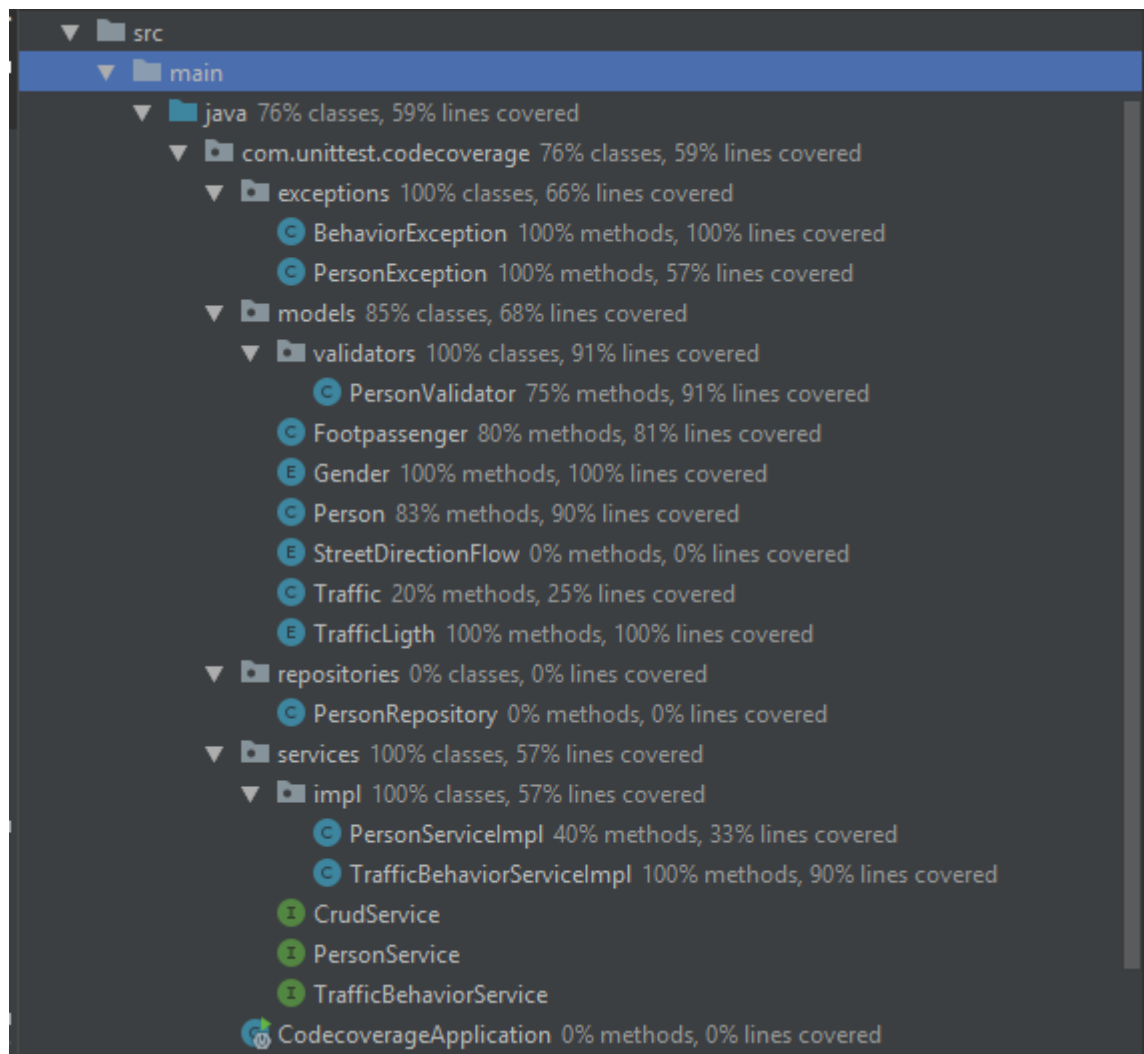
codecoverage [test] x
Downloaded from central: https://repo.maven.apache.org/maven2/org/eclipse/ee4j/project/1.0.6/project-1.0.6.pom (13 kB)
Downloaded from central: https://repo.maven.apache.org/maven2/org/eclipse/jetty/jetty-bom/9.4.35.v20201120/jetty-bom-
Downloaded from central: https://repo.maven.apache.org/maven2/org/eclipse/jetty/jetty-bom/9.4.35.v20201120/jetty-bom-
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.7.0/junit-bom-5.7.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.7.0/junit-bom-5.7.0.pom (5.1 kB a
Downloaded from central: https://repo.maven.apache.org/maven2/org/jetbrains/kotlin/kotlin-bom/1.4.21/kotlin-bom-1.4.2
Downloaded from central: https://repo.maven.apache.org/maven2/org/jetbrains/kotlin/kotlin-bom/1.4.21/kotlin-bom-1.4.2
Downloaded from central: https://repo.maven.apache.org/maven2/org/jetbrains/kotlinx/kotlinx-coroutines-bom/1.4.2/kotl
Downloaded from central: https://repo.maven.apache.org/maven2/org/jetbrains/kotlinx/kotlinx-coroutines-bom/1.4.2/kotl
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/logging/log4j/log4j-bom/2.13.3/log4j-bom-2.1
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/logging/log4j/log4j-bom/2.13.3/log4j-bom-2.1
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/logging/log4j/log4j-bom/2.13.3/log4j-bom-2.1
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/logging/logging-parent/1/logging-parent-1.p
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/logging/logging-parent/1/logging-parent-1.p
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/18/apache-18.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/apache/18/apache-18.pom (16 kB at 37 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/io/micrometer/micrometer-bom/1.6.2/micrometer-bom-1.6.2
Downloaded from central: https://repo.maven.apache.org/maven2/io/micrometer/micrometer-bom/1.6.2/micrometer-bom-1.6.2
Downloaded from central: https://repo.maven.apache.org/maven2/io/netty/netty-bom/4.1.55.Final/netty-bom-4.1.55.Final
Downloaded from central: https://repo.maven.apache.org/maven2/io/netty/netty-bom/4.1.55.Final/netty-bom-4.1.55.Final
Downloaded from central: https://repo.maven.apache.org/maven2/com/oracle/database/jdbc/ojdbc-bom/19.8.0.0/ojdbc-bom-

```



سپس با کلیک بر روی پکیج java در مسیر test، گزینه run all tests with coverage را انتخاب میکنیم.

درصد و نتایج پوشش آزمون به این صورت است:



Coverage: All in codecoverage x			
76% classes, 59% lines covered in package 'com.unittest.codecoverage'			
Element	Class, %	Method, %	Line, %
exceptions	100% (2/2)	100% (3/3)	66% (6/9)
models	85% (6/7)	61% (22/36)	68% (41/60)
repositories	0% (0/1)	0% (0/4)	0% (0/8)
services	100% (2/2)	50% (3/6)	57% (15/26)
CodecoverageApplication	0% (0/1)	0% (0/1)	0% (0/2)

Coverage: All in codecoverage ×

100% classes, 66% lines covered in package 'com.unittest.codecoverage.exceptions'

Element	Class, %	Method, %	Line, %
⚡ BehaviorException	100% (1/1)	100% (1/1)	100% (2/2)
⚡ PersonException	100% (1/1)	100% (2/2)	57% (4/7)

85% classes, 68% lines covered in package 'com.unittest.codecoverage.models'

Element	Class, %	Method, %	Line, %
📁 validators	100% (1/1)	75% (3/4)	91% (11/12)
⊙ Footpassenger	100% (1/1)	80% (8/10)	81% (13/16)
⊙ Gender	100% (1/1)	100% (2/2)	100% (2/2)
⊙ Person	100% (1/1)	83% (5/6)	90% (9/10)
⊙ StreetDirectionFlow	0% (0/1)	0% (0/2)	0% (0/2)
⊙ Traffic	100% (1/1)	20% (2/10)	25% (4/16)
⊙ TrafficLigth	100% (1/1)	100% (2/2)	100% (2/2)

Coverage: All in codecoverage ×

0% classes, 0% lines covered in package 'com.unittest.codecoverage.repositories'

Element	Class, %	Method, %	Line, %
⊙ PersonRepository	0% (0/1)	0% (0/4)	0% (0/8)

Coverage: All in codecoverage ×

100% classes, 57% lines covered in package 'com.unittest.codecoverage.services.impl'

Element	Class, %	Method, %	Line, %
⊙ PersonServiceImpl	100% (1/1)	40% (2/5)	33% (5/15)
⊙ TrafficBehaviorServiceImpl	100% (1/1)	100% (1/1)	90% (10/11)

گزارش به فرمت html را نیز در فولدر **CodeCoverageProject - Before Reports** دریافت میکنیم.

2. در این بخش سعی میکنیم با افزودن بخش هایی به کد تست، درصد پوشش آزمون کلاس ها را بهبود دهیم. از گزارش بخش قبل از تغییرات استفاده کرده تا بتوان خطوط قرمز(که مورد پوشش قرار نگرفته اند) را شناسایی و به کد تست آنها و در نتیجه درصد پوشش آزمون کلاس مربوطه بیافزاییم. داریم:

## • کلاس Footpassenger

Coverage Summary for Class: Footpassenger (com.unittest.codecoverage.models)

Class	Class, %	Method, %	Line, %
Footpassenger	100% (1/ 1)	81.8% (9/ 11)	81.2% (13/ 16)

```

1 package com.unittest.codecoverage.models;
2
3 public class Footpassenger {
4
5     private boolean crossedTheCrosswalk;
6     private TrafficLigth crossedTrafficLigth;
7     private boolean lookedToTheRight;
8     private boolean lookedToTheLeft;
9     private boolean crossedTheRoad;
10
11     public boolean crossedTheCrosswalk() {
12         return crossedTheCrosswalk;
13     }
14     public void setCrossedTheCrosswalk(boolean crossedTheCrosswalk) {
15         this.crossedTheCrosswalk = crossedTheCrosswalk;
16     }
17     public TrafficLigth getCrossedTrafficLigth() {
18         return crossedTrafficLigth;
19     }
20     public void setCrossedTrafficLigth(TrafficLigth crossedSignaling) {
21         this.crossedTrafficLigth = crossedSignaling;
22     }
23     public boolean lookedToTheRight() {
24         return lookedToTheRight;
25     }
26     public void setLookedToTheRight(boolean lookedToTheRight) {
27         this.lookedToTheRight = lookedToTheRight;
28     }
29 }

```

مقایسه بهبود درصد پوشش:

Class	Class, %	Method, %	Line, %
Footpassenger	100% (1/1)	80% (8/10)	81% (13/16)

Class	Class, %	Method, %	Line, %
Footpassenger	100% (1/1)	100% (10/10)	100% (16/16)

## تست کیس مربوطه در TrafficBehaviorServiceTest:

```

@Test
@DisplayName("Everything is ok when The footpassenger successfully and carefully crosses the road")
public void testFootpassengerCrossTheStreet_shouldEverythingIsOkWhenFootpassengerCrossesTheRoadSuccessfully() {
    Traffic currentTraffic = new Traffic();
    currentTraffic.setIntenseCarTraffic(true);
    Footpassenger currentFootpassengerBehavior = new Footpassenger();
    currentFootpassengerBehavior.setCrossedTheCrosswalk(true);
    currentFootpassengerBehavior.setLookedToTheRight(true);
    currentFootpassengerBehavior.setLookedToTheLeft(true);
    currentFootpassengerBehavior.setCrossedTrafficLigth(TrafficLigth.GREEN);

    assertTrue(currentFootpassengerBehavior.crossedTheCrosswalk());
    assertEquals("TrafficBehaviorService.crossTheStreet() should return true when footpassenger crosses the road successfully",
        true, TrafficBehaviorService.crossTheStreet(currentTraffic, currentFootpassengerBehavior));
}

```

عابر پیاده موقع سبز بودن چراغ با توجه به طرفین، با موفقیت رد بشود.

## • کلاس Person

Coverage Summary for Class: Person (com.unittest.codecoverage.models)

Class	Class, %	Method, %	Line, %
Person	100% (1/ 1)	85.7% (6/ 7)	90% (9/ 10)

```

1 package com.unittest.codecoverage.models;
2
3 import java.io.Serializable;
4
5 public class Person implements Serializable {
6
7     private static final long serialVersionUID = 1L;
8
9     private String name;
10    private int age;
11    private Gender gender;
12
13    public String getName() {
14        return name;
15    }
16    public void setName(String name) {
17        this.name = name;
18    }
19    public int getAge() {
20        return age;
21    }
22    public void setAge(int age) {
23        this.age = age;
24    }
25    public Gender getGender() {
26        return gender;
27    }
28    public void setGender(Gender gender) {
29        this.gender = gender;
30    }
31 }

```

مقایسه بهبود درصد پوشش:

Person	100% (1/1)	83% (5/6)	90% (9/10)
--------	------------	-----------	------------

Person	100% (1/1)	100% (6/6)	100% (10/10)
--------	------------	------------	--------------

تست کیس مربوطه در **PersonServiceTest**:

در یکی از تست کیس های موجود در این کلاس از **setAge** استفاده شده است. همانجا برابری **person.getAge** را با مقدار موردانتظار چک میکنیم:

```

@Test
public void testInsert_shouldInsertPersonWithSuccessWhenAllPersonsInf
    Person person = new Person();
    person.setName("Name");
    person.setAge(21);
    person.setGender(Gender.M);

    assertEquals(person.getAge(), actual: 21);
    when(repository.insert(any(Person.class))).thenReturn(person);

    service.insert(person);
}

```

## • کلاس StreetDirectionFlow

Coverage Summary for Class: StreetDirectionFlow (com.unittest.codecoverage.models)

Class	Class, %	Method, %	Line, %
StreetDirectionFlow	0% (0/ 1)	0% (0/ 2)	0% (0/ 2)

```
1 package com.unittest.codecoverage.models;
2
3 public enum StreetDirectionFlow {
4
5     ONE_WAY, TWO_WAY;
6
7 }
```

generated on 2022-12-15 01:36

مقایسه بهبود درصد پوشش:

E StreetDirectionFlow	0% (0/1)	0% (0/2)	0% (0/2)
-----------------------	----------	----------	----------

E StreetDirectionFlow	100% (1/1)	100% (2/2)	100% (2/2)
-----------------------	------------	------------	------------

تست کیس مربوطه:

به یکی از تست کیس ها که در آن چراغ قرمز و ترافیک سنگین است، جهت یک طرفه را نیز اضافه و چک میکنیم.

```
@Test
public void testFootpassengerCrossTheStreet_shouldThrowBehaviorExceptionWhenFootpassengerCrossesTheRoadDuringHeavyTraffic() {
    Traffic currentTraffic = new Traffic();
    currentTraffic.setIntenseCarTraffic(true);
    currentTraffic.setStreetDirectionFlow(StreetDirectionFlow.ONE_WAY);

    Footpassenger currentFootpassengerBehavior = new Footpassenger();
    currentFootpassengerBehavior.setCrossedTheRoad(true);
    currentFootpassengerBehavior.setCrossedTrafficLigth(TrafficLigth.RED);

    assertEquals(currentTraffic.getStreetDirectionFlow(), StreetDirectionFlow.ONE_WAY);
    Assertions.assertThatThrownBy(() -> trafficBehaviorService.footpassengerCrossTheStreet(currentTraffic, currentFootpassengerBehavior))
        .isInstanceOf(BehaviorException.class)
        .hasMessageContaining("You shouldn't do that, reckless person!");
}
```

## • کلاس Traffic



Class	Class, %	Method, %	Line, %
Traffic	100% (1/ 1)	27.3% (3/ 11)	25% (4/ 16)

```

1 package com.unittest.codecoverage.models;
2
3 public class Traffic {
4     private TrafficLigth currentTrafficLight;
5     private boolean intenseCarTraffic;
6     private short maxSpeedAllowed;
7     private short minSpeedAllowed;
8     private StreetDirectionFlow streetDirectionFlow;
9
10    public TrafficLigth getCurrentTrafficLight() {
11        return currentTrafficLight;
12    }
13    public void setCurrentTrafficLight(TrafficLigth currentSignaling) {
14        this.currentTrafficLight = currentSignaling;
15    }
16    public boolean intenseCarTraffic() {
17        return intenseCarTraffic;
18    }
19    public void setIntenseCarTraffic(boolean intenseCarTraffic) {
20        this.intenseCarTraffic = intenseCarTraffic;
21    }
22    public short getMaxSpeedAllowed() {
23        return maxSpeedAllowed;
24    }
25    public void setMaxSpeedAllowed(short maxSpeedAllowed) {
26        this.maxSpeedAllowed = maxSpeedAllowed;
27    }
28    public short getMinSpeedAllowed() {
29        return minSpeedAllowed;
30    }
31    public void setMinSpeedAllowed(short minSpeedAllowed) {
32        this.minSpeedAllowed = minSpeedAllowed;
33    }
34    public StreetDirectionFlow getStreetDirectionFlow() {
35        return streetDirectionFlow;
36    }
37    public void setStreetDirectionFlow(StreetDirectionFlow streetDirectionFlow) {
38        this.streetDirectionFlow = streetDirectionFlow;
39    }
40 }

```

مقایسه بهبود درصد پوشش:

 Traffic	100% (1/1)	20% (2/10)	25% (4/16)
---	------------	------------	------------

 Traffic	100% (1/1)	60% (6/10)	68% (11/16)
---	------------	------------	-------------

تست کیس مربوطه:

به تستی که برای کلاس **Footpassenger** اضافه کردیم، حداقل سرعت 30 و حداکثر 45 را اضافه و آن را چک میکنیم. (ترافیک سبک، چراغ سبز، محدوده سرعت کم => بدون اکسپشن)

```

@Test
@DisplayName("Everything is ok when The footpassenger successfully and carefully cro
public void testFootpassengerCrossTheStreet_shouldEverythingIsOkWhenFootpassengerCro

    Traffic currentTraffic = new Traffic();
    currentTraffic.setIntenseCarTraffic(true);
    Footpassenger currentFootpassengerBehavior = new Footpassenger();
    currentFootpassengerBehavior.setCrossedTheCrosswalk(true);
    currentFootpassengerBehavior.setLookedToTheRight(true);
    currentFootpassengerBehavior.setLookedToTheLeft(true);
    currentFootpassengerBehavior.setCrossedTrafficLigth(TrafficLigth.GREEN);
    currentTraffic.setMaxSpeedAllowed((short)45);
    currentTraffic.setMinSpeedAllowed((short)30);

    assertTrue(currentFootpassengerBehavior.crossedTheCrosswalk());
    assertAll(() -> trafficBehaviorService.footpassengerCrossTheStreet(currentTraffic
}

```

## • کلاس PersonRepository

total	0% (0/ 5)	0% (0/ 9)
-------	-----------	-----------

```

1 package com.unittest.codecoverage.repositories;
2
3 import java.util.Objects;
4
5 import com.unittest.codecoverage.models.Person;
6
7 public class PersonRepository {
8
9     public Person insert(Person person) {
10         Objects.requireNonNull(person, "person can't be null");
11         return person;
12     }
13
14     public void update(Person person) {
15         Objects.requireNonNull(person, "person can't be null");
16     }
17
18     public void delete(String name) {
19         Objects.requireNonNull(name, "name can't be null");
20     }
21
22     public Person get(String name) {
23         Objects.requireNonNull(name, "name can't be null");
24         return null;
25     }
26
27 }

```

مقایسه بهبود درصد پوشش:

0% classes, 0% lines covered in package 'com.unittest.codecoverage.repositories'			
Element	Class, %	Method, %	Line, %
PersonRepository	0% (0/1)	0% (0/4)	0% (0/8)

100% classes, 100% lines covered in package 'com.unittest.codecoverage.repositories'			
Element	Class, %	Method, %	Line, %
PersonRepository	100% (1/1)	100% (4/4)	100% (9/9)

تست کیس مربوطه:

یک آجکت **Person** و یک آجکت **PersonRepository** ساخته و متدهای ریپازیتوری را استفاده میکنیم. برای معنادار بودن، در دو حالت **null** و غیر **null** آن ها را بررسی میکنیم.

```

@Test
public void testInsertUpdateGetDeleteOfRepository_shouldInsertAndUpdateAndGetAndDeleteOfPerson() {
    Person person = new Person();
    person.setName("Name");
    person.setAge(50);
    person.setGender(Gender.F);
    PersonRepository newRepo = new PersonRepository();

    Person newPerson = newRepo.insert(person);
    assertEquals(newPerson, person);

    newRepo.update(person);
    assertThatThrownBy(() -> newRepo.update(person: null))
        .hasMessage("person can't be null");

    newRepo.get(person.getName());
    assertThatThrownBy(() -> newRepo.get(null))
        .hasMessage("name can't be null");

    newRepo.delete(person.getName());
}

```

## • کلاس PersonServiceImpl

Class	Class, %	Method, %	Line, %
PersonServiceImpl	100% (1/ 1)	40% (2/ 5)	33.3% (5/ 15)

```

1 package con.unittest.codecoverage.services.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Service;
5
6 import con.unittest.codecoverage.exceptions.PersonException;
7 import con.unittest.codecoverage.models.Person;
8 import con.unittest.codecoverage.models.validators.PersonValidator;
9 import con.unittest.codecoverage.repositories.PersonRepository;
10 import con.unittest.codecoverage.services.PersonService;
11
12 @Service
13 public class PersonServiceImpl implements PersonService {
14
15     private PersonValidator validator;
16     @Autowired
17     private PersonRepository repository;
18
19     @Autowired
20     public PersonServiceImpl() {
21         this.validator = new PersonValidator();
22     }
23
24     @Override
25     public Person insert(Person person) {
26         validator.validate(person);
27         return repository.insert(person);
28     }
29
30     @Override
31     public void update(Person person) {
32         validator.validate(person);
33         repository.update(person);
34     }
35
36     @Override
37     public Person get(String name) {
38         if(validator.requiredName(name)) {
39             throw new PersonException("Name is required");
40         }
41         return repository.get(name);
42     }
43
44     @Override
45     public void delete(String name) {
46         if(validator.requiredName(name)) {
47             throw new PersonException("Name is required");
48         }
49         repository.delete(name);
50     }
51 }

```

مقایسه بهبود درصد پوشش:

Element	Class, %	Method, %	Line, %
PersonServiceImpl	100% (1/1)	40% (2/5)	33% (5/15)

PersonServiceImpl	100% (1/1)	100% (5/5)	80% (12/15)
-------------------	------------	------------	-------------

تست کیس مربوطه:

با کمک service، اطلاعات کاربری را به روزسانی، دریافت و حذف میکنیم (خطوط قرمز):

```

@Test
public void testPersonServiceImpl_shouldUpdateAndGetAndDeleteOfPersonsInfoInPersonServiceImpl() {

    Person person = new Person();
    person.setName("Name");
    person.setAge(50);
    person.setGender(Gender.F);

    service.update(person);
    assertThatThrownBy(() -> service.update( object: null))
        .isInstanceOf(PersonException.class)
        .hasMessage("Name is required;Gender is required");

    assertThatThrownBy(() -> service.get("Name"))
        .isInstanceOf(PersonException.class)
        .hasMessage("Name is required");

    assertThatThrownBy(() -> service.delete( id: "Name"))
        .isInstanceOf(PersonException.class)
        .hasMessage("Name is required");
}

```

- درصد بهبود کلی ( تمام کلاس ها بهبود داشته اند)

