گزارش کار آزمایش ششم آزمایشگاه مهندسی نرم‌افزار

لینک آزمایش: https://github.com/yasmansh/SE_LAB/tree/main/Exp6

اعضای گروه:

یاسمن شیخان 97101915

امیرحسین علی‌محمدی 97110166

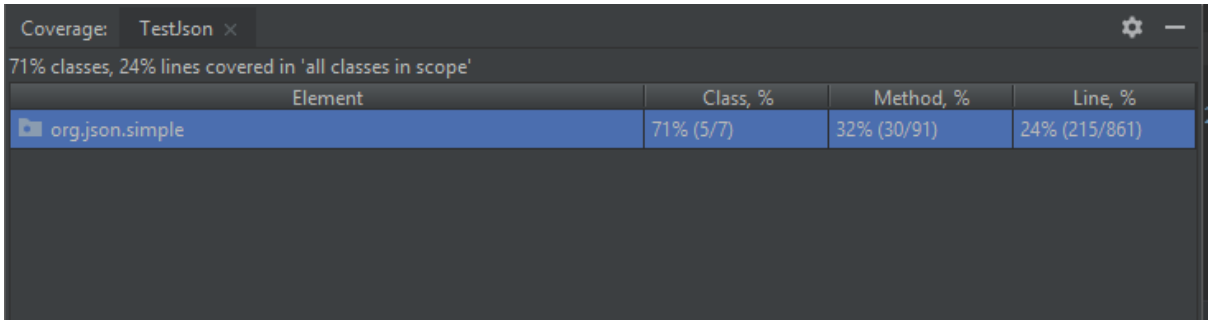
---

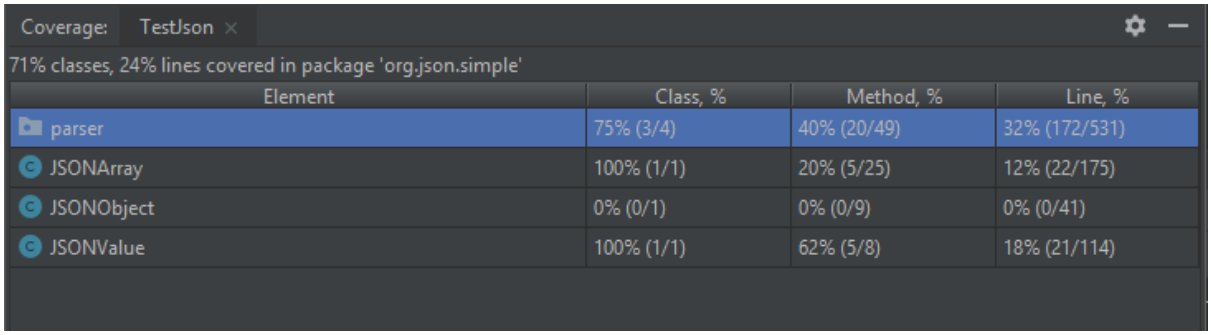
1. پروژه json-simple

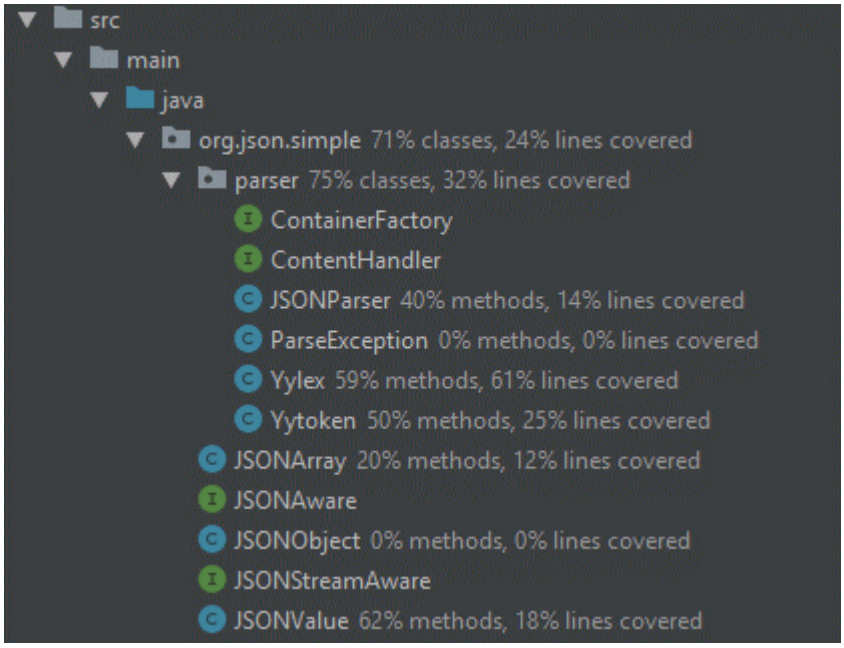
۱.۱ – ابتدا پروژه را در محیط intellij باز کرده و چون jUnit و SDK از قبل اضافه و نصب شده اند کلاس ها شناسایی میشوند.

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

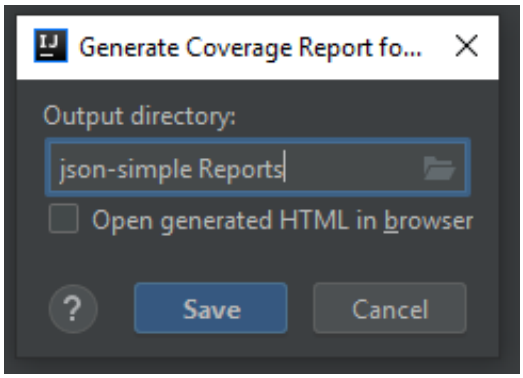
درصد کلاس ها ، متدها و خط های برنامه که موردتست قرار میگیرند را درخروجی مشاهده میکنیم:







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



[all classes]

Overall Coverage Summary

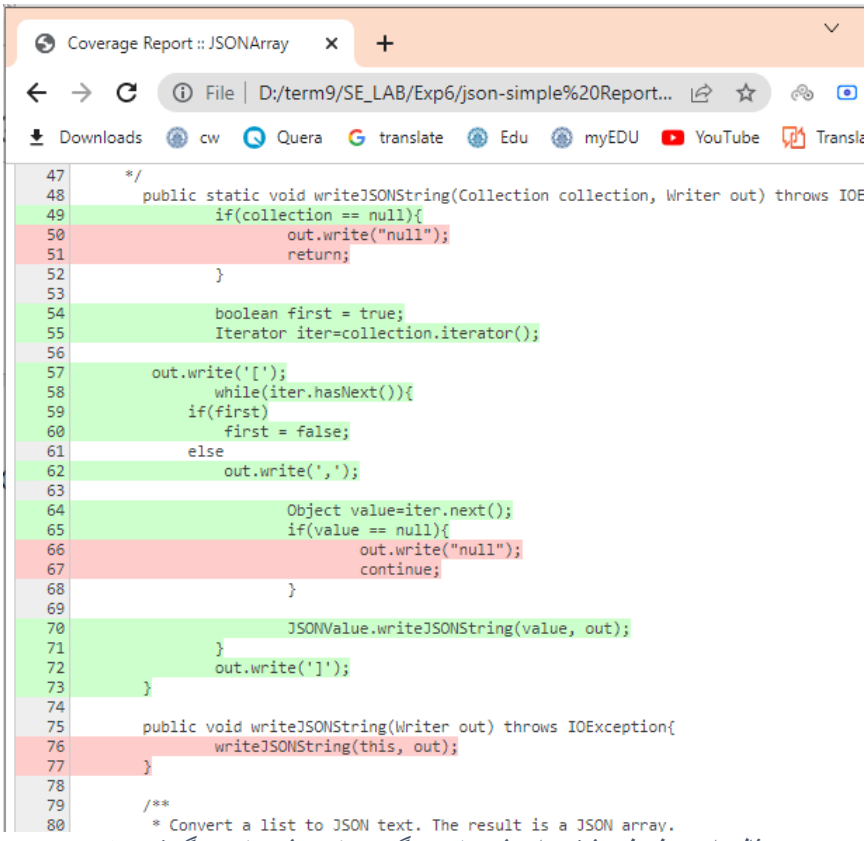
Package	Class, %	Method, %	Line, %
all classes	71.4% (5/ 7)	33.3% (31/ 93)	25.1% (215/ 857)

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

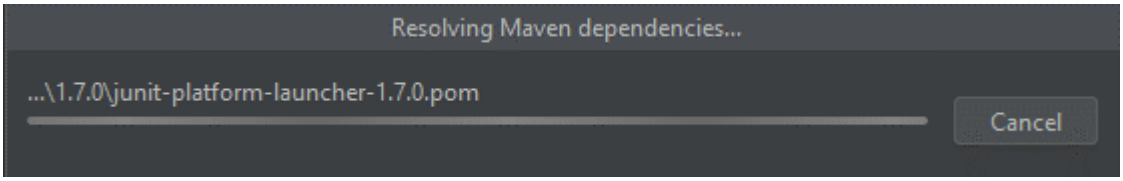
index.html



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

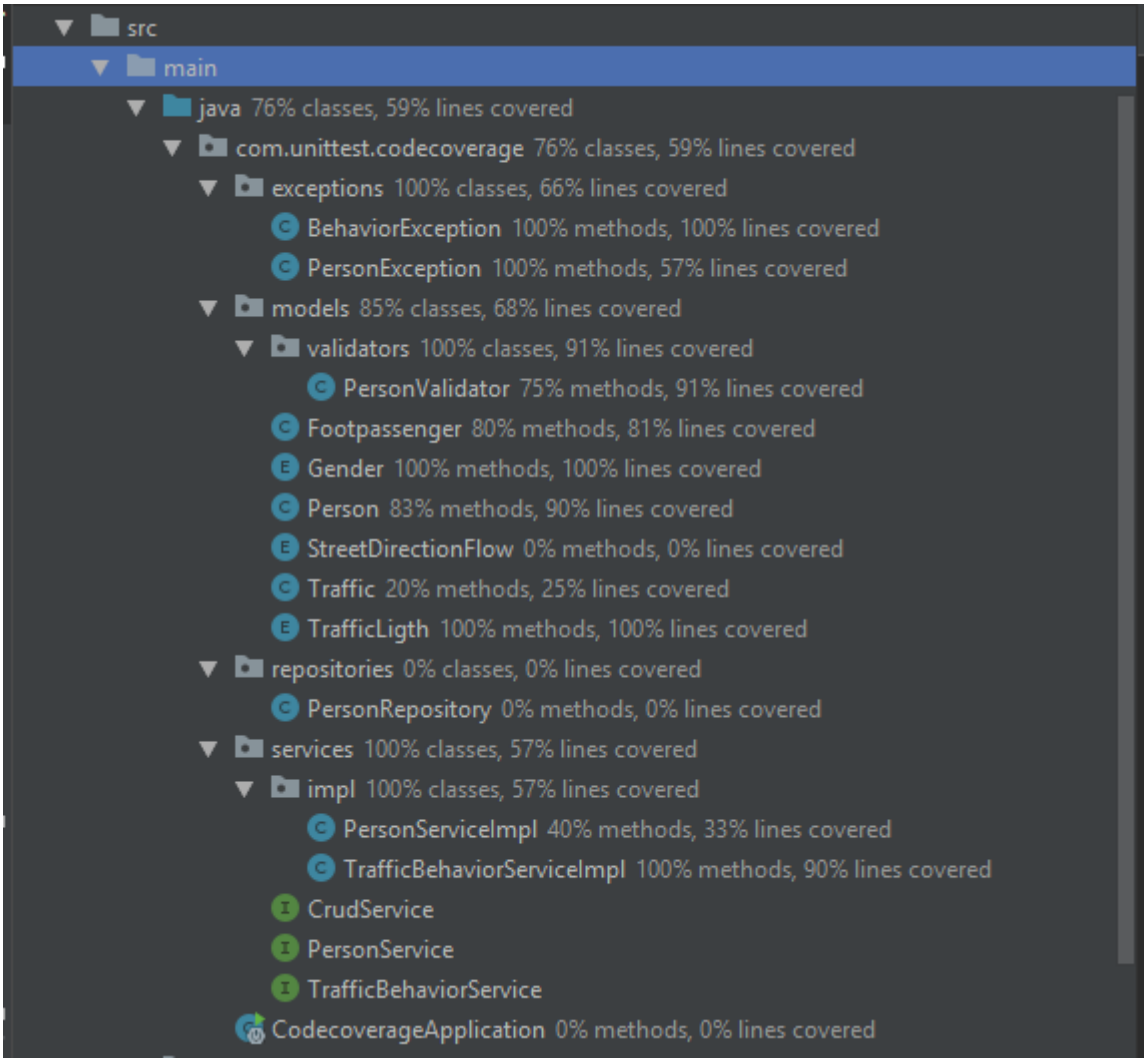
تمرین(CodeCoverageProject)

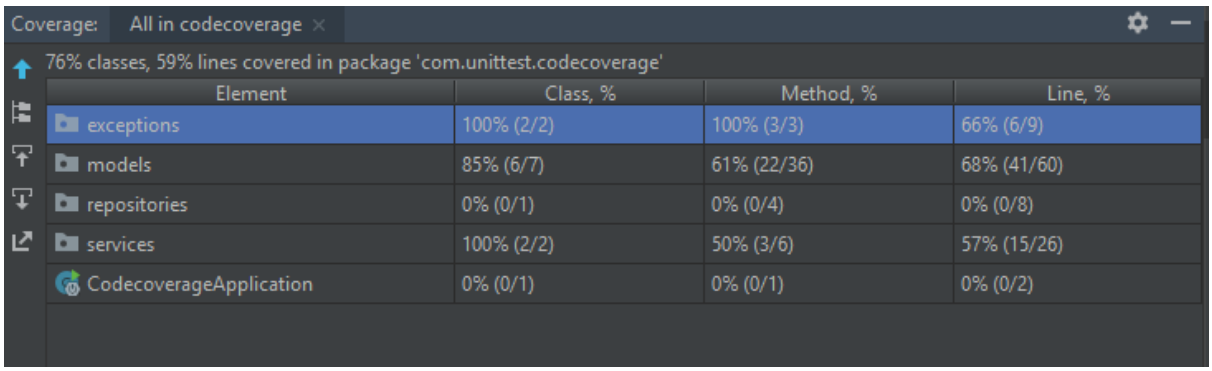
1. **مشابه بخش قبل، ابتدا ابتدا پروژه را در محیط intellij باز کرده و چون jUnit و SDK از قبل اضافه و نصب شده اند کلاس ها شناسایی میشوند اما برخی متدها شناسایی نمیشوند که تمامی آنها را نیز دانلود/ ایمپورت میکنیم.**

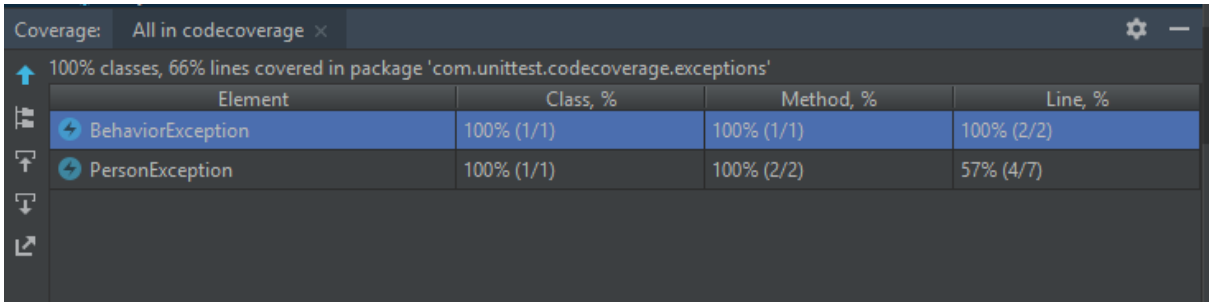


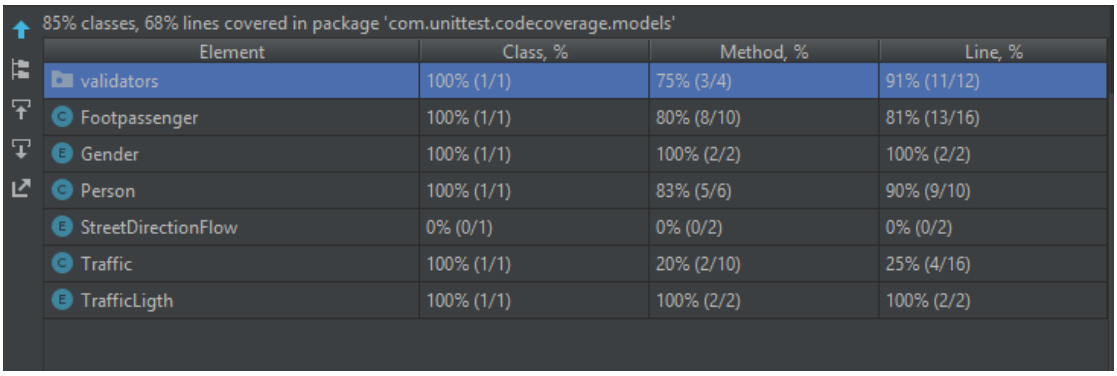
سپس با کلیک بر روی پکیج java در مسیر test، گزینه run all tests with coverage را انتخاب میکنیم.

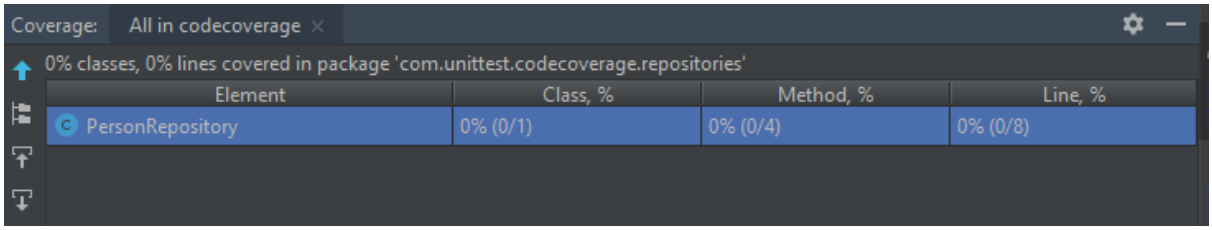
درصد و نتایج پوشش آزمون به این صورت است:

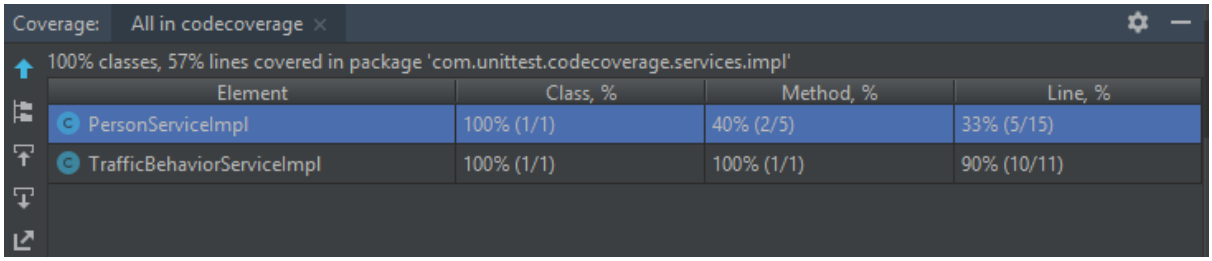












گزارش به فرمت html را نیز در فولدر CodeCoverageProject - Before Reports دریافت میکنیم.

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 درنتیجه درصد پوشش آزمون کلاس مربوطه بیافزاییم. داریم:

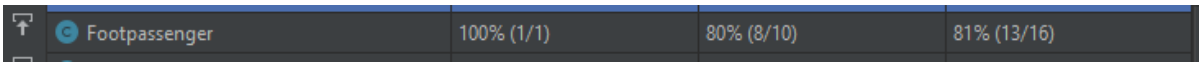
- کلاس **Footpassenger**

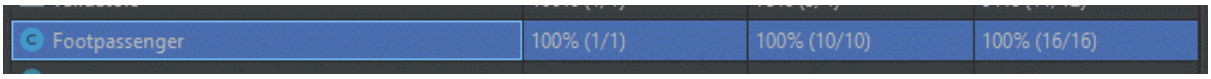
Coverage Summary for Class: Footpassenger (com.unittest.codecoverage.models)

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         }
```

مقایسه بهبود درصد پوشش:





تست کیس مربوطه در TraficBehaviorServiceTest:

```
@Test
@DisplayName("Everything is ok when The footpassenger successfully and carefully crosses the road")
public void testFootpassengerCrossTheStreet_shouldEverythingIsOkWhenFootpassengerCrossesTheRoadSuccessfully() {

    Traffic currentTrafic = new Traffic();
    currentTrafic.setIntenseCarTraffic(true);
    Footpassenger currentFootpassengerBehavior = new Footpassenger();
    currentFootpassengerBehavior.setCrossedTheCrosswalk(true);
    currentFootpassengerBehavior.setLookedToTheRight(true);
    currentFootpassengerBehavior.setLookedToTheLeft(true);
    currentFootpassengerBehavior.setCrossedTrafficLigth(TrafficLigth.GREEN);

    assertTrue(currentFootpassengerBehavior.crossedTheCrosswalk());
    assertAll(() -> trafficBehaviorService.footpassengerCrossTheStreet(currentTrafic, currentFootpassengerBehavior));
}
```

عابرپیاده 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;
```

مقایسه بهبود درصد پوشش:

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:

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_shouldThrowBehaviorExceptionWhenFootpassengerCrossesTheRoadDuringHeavyTraff

    Traffic currentTrafic = new Traffic();
    currentTrafic.setIntenseCarTraffic(true);
    currentTrafic.setStreetDirectionFlow(StreetDirectionFlow.ONE_WAY);

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

    assertEquals(currentTrafic.getStreetDirectionFlow(), StreetDirectionFlow.ONE_WAY);
    Assertions.assertThatThrownBy(() -> trafficBehaviorService.footpassengerCrossTheStreet(currentTrafic, currentFootp
            .isInstanceOf(BehaviorException.class)
            .hasMessageContaining("You should'nt do that, reckless person");

}
```

- کلاس **Trafic**

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

مقایسه بهبود درصد پوشش:

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


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

- 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 آبجکت Person و یک آبجکت 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_shouldInsertAndUpdateAndGetAndDeleteOfPer
    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)

```java
package com.unittest.codecoverage.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.unittest.codecoverage.exceptions.PersonException;
import com.unittest.codecoverage.models.Person;
import com.unittest.codecoverage.models.validators.PersonValidator;
import com.unittest.codecoverage.repositories.PersonRepository;
import com.unittest.codecoverage.services.PersonService;

@Service
public class PersonServiceImpl implements PersonService {

	private PersonValidator validator;
	@Autowired
	private PersonRepository repository;

	@Autowired
	public PersonServiceImpl() {
		this.validator = new PersonValidator();
	}

	@Override
	public Person insert(Person person) {
		validator.validate(person);
		return repository.insert(person);
	}

	@Override
	public void update(Person person) {
		validator.validate(person);
		repository.update(person);
	}

	@Override
	public Person get(String name) {
		if(validator.requiredName(name)) {
			throw new PersonException("Name is required");
		}
		return repository.get(name);
	}

	@Override
	public void delete(String name) {
		if(validator.requiredName(name)) {
			throw new PersonException("Name is required");
		}
		repository.delete(name);
```

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):

```java
@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)**