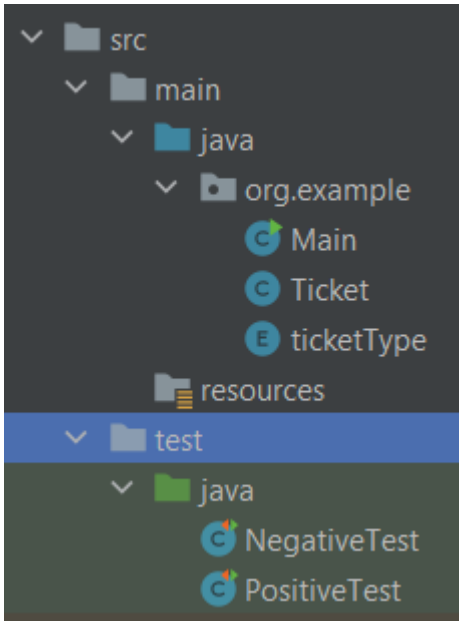
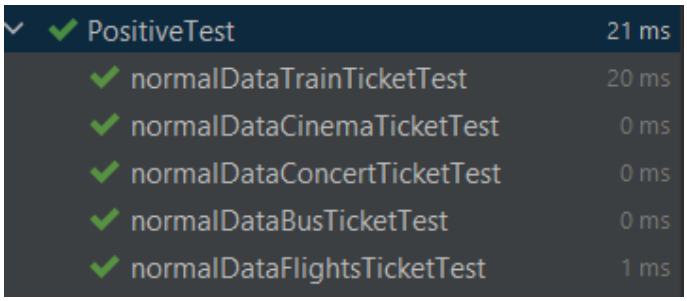
Мамонов Антон 3ИСиП-19-1



Программа состоит из двух пакетов main и test. Пакет main содержит в себе 3 класса для полноценной работы программы – Main.java, Ticket.java, ticketType.java. Пакет test содержит в себе два класса для тестирования программы PositiveTest.java и NegativeTest.java.



5 автоматизированных тестов с позитивным сценарием:

Тестирование суммы возврата в зависимости от типа билета без изменения даты покупки и окончания билета.

При типе билета = «BUS».

```
@Test
public void normalDataBusTicketTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "BUS"),
             description: "ticket on bus",
             price: 2000,
             ownerName: "Anton",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "1080.0", result);
}
```

При типе билета = «CINEMA»

```
@Test
public void normalDataCinemaTicketTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "CINEMA"),
            description: "ticket on cinema",
            price: 2000,
            ownerName: "Dima",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "1140.0",  result);
}
```

Ожидаемая сумма возврата совпала с полученной

При типе билета = «CONCERT»

```
@Test
public void normalDataConcertTicketTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "CONCERT"),
            description: "ticket on concert",
            price: 2000,
            ownerName: "Vlad",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "840.0",  result);
}
```

Ожидаемая сумма возврата совпала с полученной

При типе билета = «FLIGHTS»

```
@Test
public void normalDataFlightsTicketTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "FLIGHTS"),
            description: "ticket on flights",
            price: 2000,
            ownerName: "Misha",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "720.0",  result);
}
```

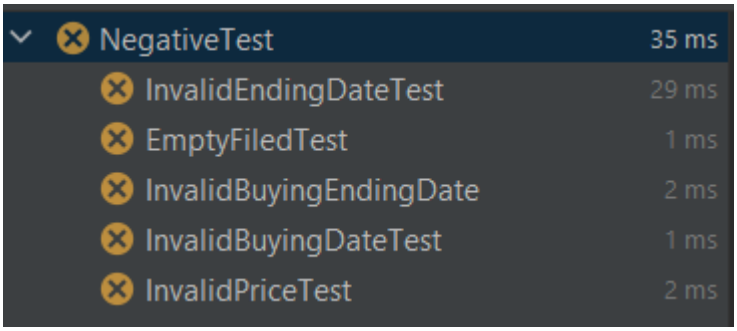
Ожидаемая сумма возврата совпала с полученной

При типе билета = «TRAIN»

```
@Test
public void normalDataTrainTicketTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "TRAIN"),
            description: "ticket on train",
            price: 2000,
            ownerName: "Denis",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "960.0",  result);
}
```

Ожидаемая сумма возврата совпала с полученной

5 автоматизированных тестов с негативным сценарием:



При вводе некорректной даты покупки билета должно быть выведено сообщение, сообщающее о неверно введенных данных

```
@Test
public void InvalidBuyingDateTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "BUS"),
            description: "ticket on bus",
            price: 2000,
            ownerName: "Anton",
            LocalDateTime.parse( text: "0001-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "Invalid buying date", result);
}
```

После выполнения была получена сумма возврата, не удовлетворяющая ожиданиям

```
org.junit.ComparisonFailure:
Expected :Invalid buying date
Actual   :1080.0
<Click to see difference>
```

При вводе некорректной даты окончания билета должно быть выведено сообщение, сообщающее о неверно введенных данных

```
@Test
public void InvalidEndingDateTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "BUS"),
            description: "ticket on bus",
            price: 2000,
            ownerName: "Anton",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "5000-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "Invalid ending date", result);
}
```

После выполнения была получена сумма возврата, не удовлетворяющая ожиданиям

```
org.junit.ComparisonFailure:
Expected :Invalid ending date
Actual   :1080.0
<Click to see difference>
```

При вводе даты окончания билета, которая меньше (ранее) даты покупки билета, должно быть выведено сообщение, сообщающее о неверно введенных данных

```
@Test
public void InvalidBuyingEndingDate() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "BUS"),
            description: "ticket on bus",
            price: 2000,
            ownerName: "Anton",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2021-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "Ending date cannot be less buying date", result);
}
```

После выполнения была получена сумма возврата = 0, не удовлетворяющая ожиданиям

```
org.junit.ComparisonFailure:
Expected :Ending date cannot be less buying date
Actual   :0.0
<Click to see difference>
```

При вводе некорректной цены билета должно быть выведено сообщение, сообщающее о неверно введенных данных

```
@Test
public void InvalidPriceTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "BUS"),
            description: "ticket on bus",
            price: -2000,
            ownerName: "Anton",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "Invalid price", result);
}
```

После выполнения была получена отрицательная сумма возврата, не удовлетворяющая ожиданиям

```
org.junit.ComparisonFailure:
Expected :Invalid price
Actual   :-1080.0
<Click to see difference>
```

При вводе пустых значений должно быть выведено сообщение, сообщающее о вводе пусты значений

```
@Test
public void EmptyFiledTest() {
    var actualResult :double = Main.createTicket(
            ticketType.valueOf( name: "BUS"),
            description: "",
            price: -2000,
            ownerName: "",
            LocalDateTime.parse( text: "2022-05-10 15:00", formatter),
            LocalDateTime.parse( text: "2022-05-27 15:00", formatter));
    String result = String.valueOf(actualResult);
    Assert.assertEquals( expected: "Invalid description and name", result);
}
```

После выполнения была получена отрицательная сумма возврата, не удовлетворяющая ожиданиям

```
org.junit.ComparisonFailure:
Expected :Invalid description and name
Actual   :-1080.0
<Click to see difference>
```

Принцип работы программы со стороны пользователя для написания ручных тестов:

```
Enter one of the ticket types:
[CONCERT, CINEMA, FLIGHTS, BUS, TRAIN]
BUS
Enter the ticket description
ticket on bus
Enter your full name
Anton
Enter the date of ticket purchase in the format: "yyyy-MM-dd HH:mm"
2022-05-10 10:00
Enter the date of ticket execution in the format: "yyyy-MM-dd HH:mm"
2022-05-31 18:00
Enter ticket price
400
288,00
```

Тест 1. При указании некорректного типа билета должно появиться сообщения с данной информацией

```
Enter one of the ticket types:
[CONCERT, CINEMA, FLIGHTS, BUS, TRAIN]
OPERA
Exception in thread "main" java.lang.IllegalArgumentException Create breakpoint : No enum constant org.example.ticketType.OPERA
    at java.base/java.lang.Enum.valueOf(Enum.java:273)
    at org.example.ticketType.valueOf(ticketType.java:3)
    at org.example.Main.startWork(Main.java:34)
    at org.example.Main.main(Main.java:21)
```

После ввода некорректного значения программа закрылась с ошибкой

Тест 2. При вводе данных (“CONCERT”, “bilet on concert”, “Denis”, “2022-04-24 14:10”, “2022-06-01 15:00”, 3000) мы должны получить результат суммы возврата = 1680,00

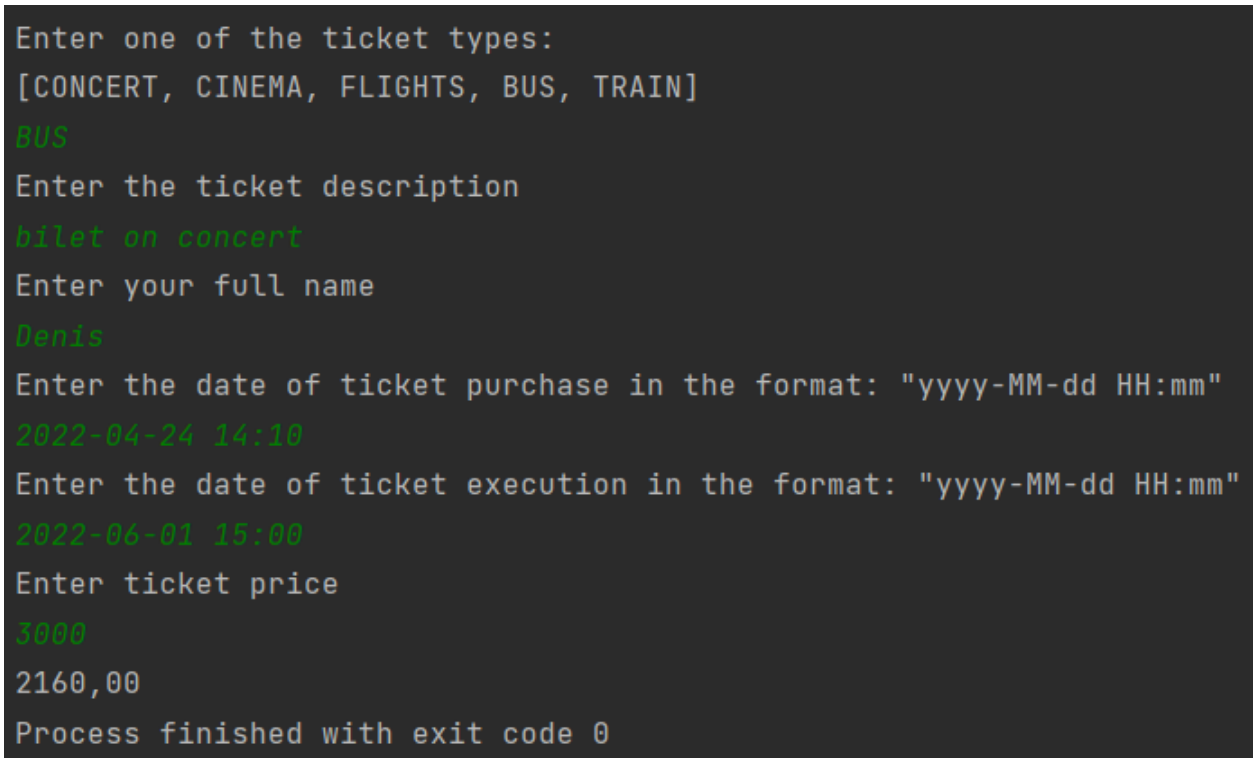
```
Enter one of the ticket types:
[CONCERT, CINEMA, FLIGHTS, BUS, TRAIN]
CONCERT
Enter the ticket description
bilet on concert
Enter your full name
Denis
Enter the date of ticket purchase in the format: "yyyy-MM-dd HH:mm"
2022-04-24 14:10
Enter the date of ticket execution in the format: "yyyy-MM-dd HH:mm"
2022-06-01 15:00
Enter ticket price
3000
1680,00
Process finished with exit code 0
```

При вводе корректных данных мы получили ожидаемый результат

Тест 3. При вводе какого-либо пустого значения должно появиться сообщение с информацией для пользователя

При оставлении пустыми полей description и full name программа продолжила работу. При оставлении пустым поля buying date программа закончила работу с ошибкой

Тест 4. При вводе аналогичных данных Тесту 2, изменив тип билета на “BUS”, получим иную сумму возврата = 2160,00



После выполнения программа вывела ожидаемый результат

Тест 5. При вводе отрицательной стоимости билета должно появиться сообщение о вводе неверной информации

```
Enter one of the ticket types:
[CONCERT, CINEMA, FLIGHTS, BUS, TRAIN]
BUS
Enter the ticket description
bilet on bus
Enter your full name
Denis
Enter the date of ticket purchase in the format: "yyyy-MM-dd HH:mm"
2022-04-10 15:00
Enter the date of ticket execution in the format: "yyyy-MM-dd HH:mm"
2022-05-31 15:00
Enter ticket price
-10000
-7200,00
```

После выполнения программы была выведена отрицательная сумма возврата