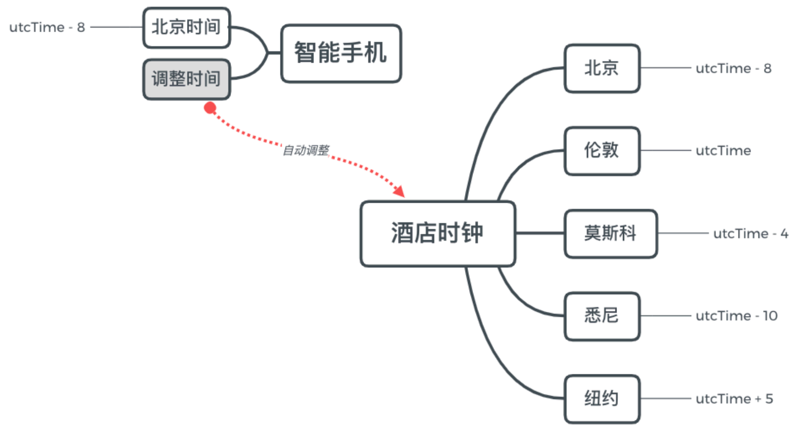
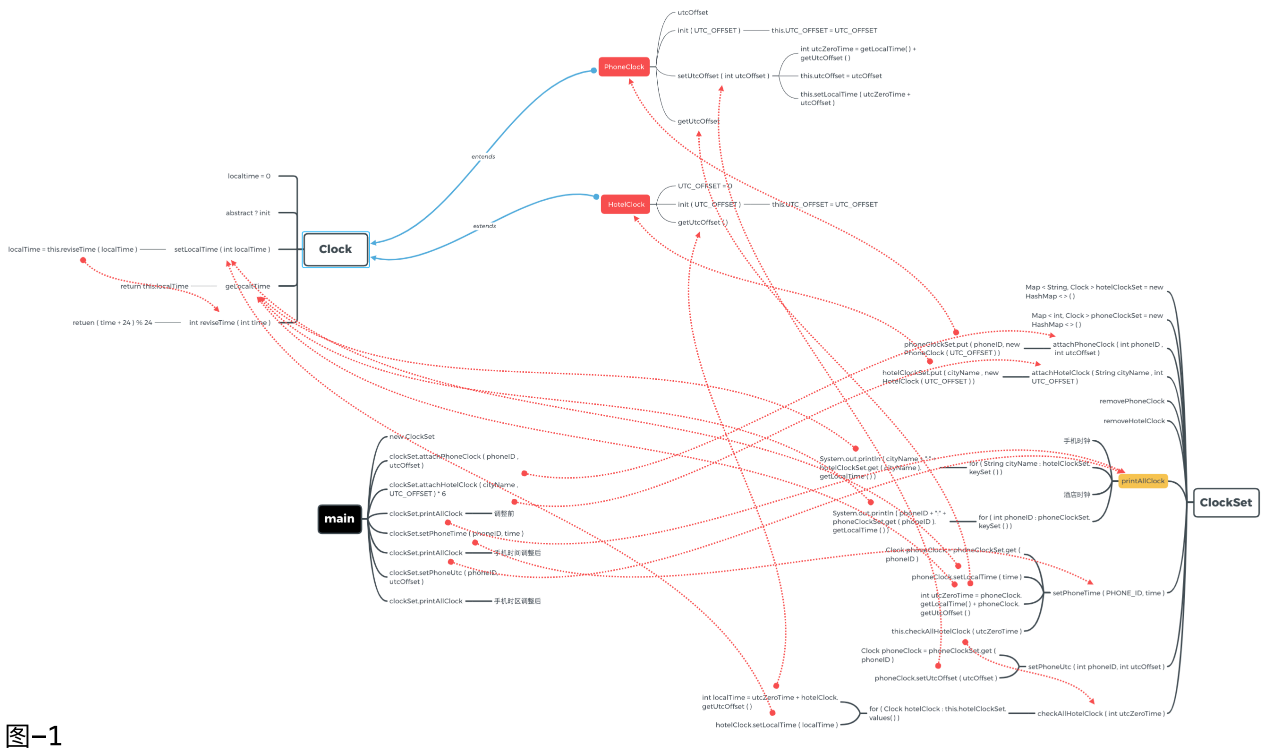
码农世界酒店单元测试报告

初期项目代码设计思维导图




PhoneClock
HotelClock
Clock
ClockSet
main
extends
extends
printAllClock
setPhoneTime (PHONE_ID, time)
setPhoneUtc (int phoneID, int utcOffset)
checkAllHotelClock (int utcZeroTime)
attachPhoneClock (int phoneID , int utcOffset)
attachHotelClock (String cityName , int UTC_OFFSET)
removePhoneClock
removeHotelClock
new ClockSet
clockSet.printAllClock
手机时钟
酒店时钟

图–1

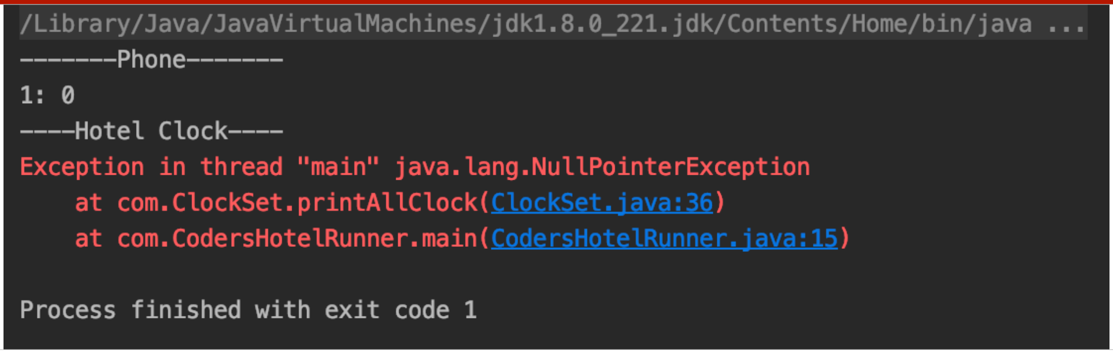
一共 ClockSet, Clock, PhoneClock, HotelClock 四个类 (main 为单元测试使用类)
蓝色箭头代表 extend 类的继承继承；红色虚线代表函数调用

单元测试

一、基于 main 方法

如图-1中 mai n所示，调用ClockSet.printAllClock()函数，分别展示初始化后，手机时间调整后，手机时区调整后三个状态所有时钟的值

ISSUE 1

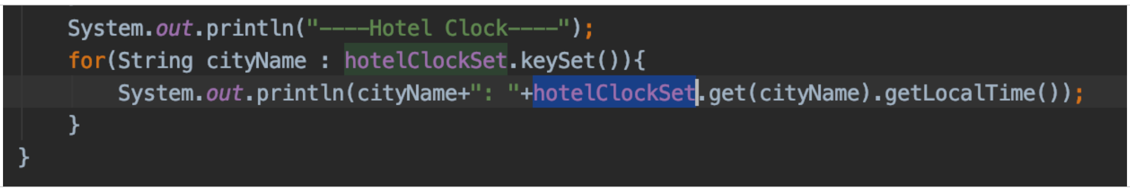


```
/Library/Java/JavaVirtualMachines/jdk1.8.0_221.jdk/Contents/Home/bin/java ...
--------Phone--------
1: 0
----Hotel Clock----
Exception in thread "main" java.lang.NullPointerException
	at com.ClockSet.printAllClock(ClockSet.java:36)
	at com.CodersHotelRunner.main(CodersHotelRunner.java:15)

Process finished with exit code 1
```

图-2

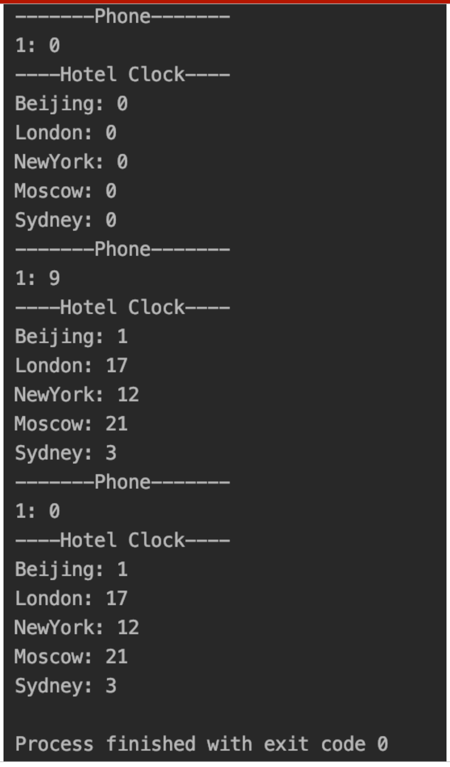
如图-2所示，调用 printAllClock 时抛出 NullPointerException
检查代码后发现在输出酒店信息时，原有错误代码调用了 phoneClockSet对象的get()方法，正确逻辑应该调用 hoteClockSet 对象的 get()方法，修改代码后如图-3所示



```
        System.out.println("----Hotel Clock----");
        for(String cityName : hotelClockSet.keySet()){
            System.out.println(cityName+": "+hotelClockSet.get(cityName).getLocalTime());
        }
    }
```

图-3

ISSUE 2



```
--------Phone--------
1: 0
----Hotel Clock----
Beijing: 0
London: 0
NewYork: 0
Moscow: 0
Sydney: 0
--------Phone--------
1: 9
----Hotel Clock----
Beijing: 1
London: 17
NewYork: 12
Moscow: 21
Sydney: 3
--------Phone--------
1: 0
----Hotel Clock----
Beijing: 1
London: 17
NewYork: 12
Moscow: 21
Sydney: 3

Process finished with exit code 0
```

图-4

输出信息发现与预期不符合，若手机（时区同北京）时间调整为:9后，北京时间应为:1。
检查代码逻辑后发现在 ClockSet.setPhoneTime(int phoneID, int time) 函数中，计算零时区时间的逻辑错误。
原有错误代码为

零时区时间 = 手机本地时间 + 手机时差

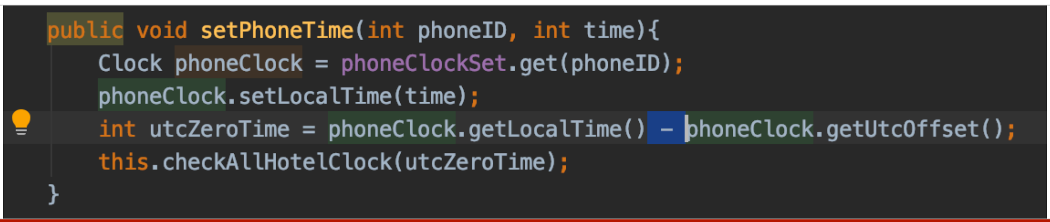
正确逻辑为

手机本地时间 = 零时区时间 + 手机时差

即

零时区时间 = 手机本地时间 - 手机时差

更改后代码如图-5所示

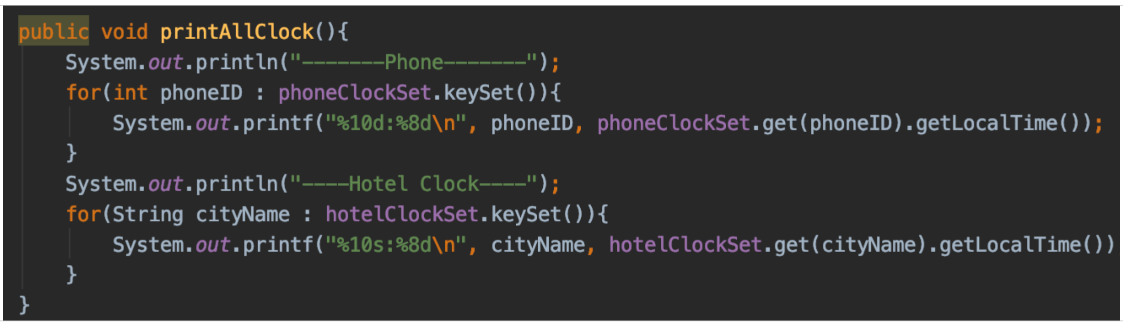


```java
public void setPhoneTime(int phoneID, int time){
    Clock phoneClock = phoneClockSet.get(phoneID);
    phoneClock.setLocalTime(time);
    int utcZeroTime = phoneClock.getLocalTime() - phoneClock.getUtcOffset();
    this.checkAllHotelClock(utcZeroTime);
}
```

图-5

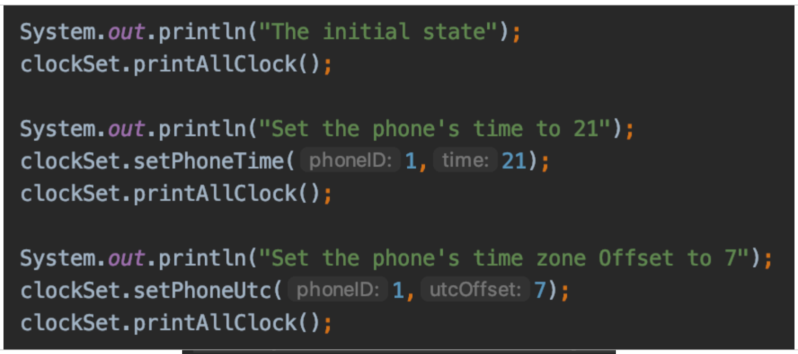
* ISSUE 3

调整输出格式，附加输出信息，正确输出信息如图-8所示



```java
public void printAllClock(){
    System.out.println("--------Phone--------");
    for(int phoneID : phoneClockSet.keySet()){
        System.out.printf("%10d:%8d\n", phoneID, phoneClockSet.get(phoneID).getLocalTime());
    }
    System.out.println("----Hotel Clock----");
    for(String cityName : hotelClockSet.keySet()){
        System.out.printf("%10s:%8d\n", cityName, hotelClockSet.get(cityName).getLocalTime())
    }
}
```

图-6

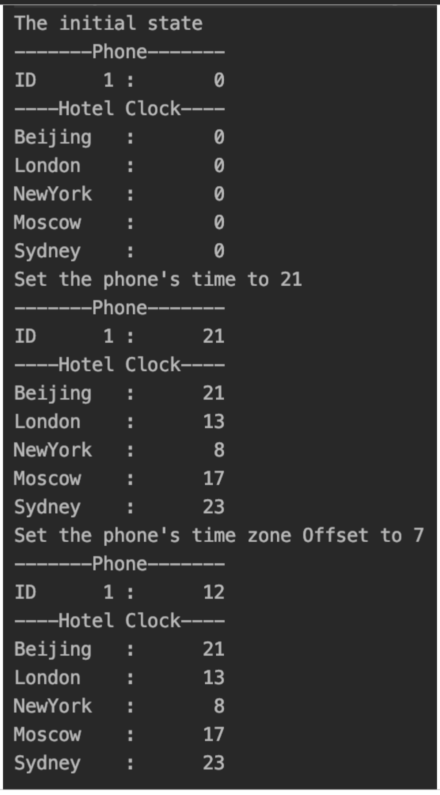


```
System.out.println("The initial state");
clockSet.printAllClock();

System.out.println("Set the phone's time to 21");
clockSet.setPhoneTime( phoneID: 1, time: 21);
clockSet.printAllClock();

System.out.println("Set the phone's time zone Offset to 7");
clockSet.setPhoneUtc( phoneID: 1, utcOffset: 7);
clockSet.printAllClock();
```

图–7

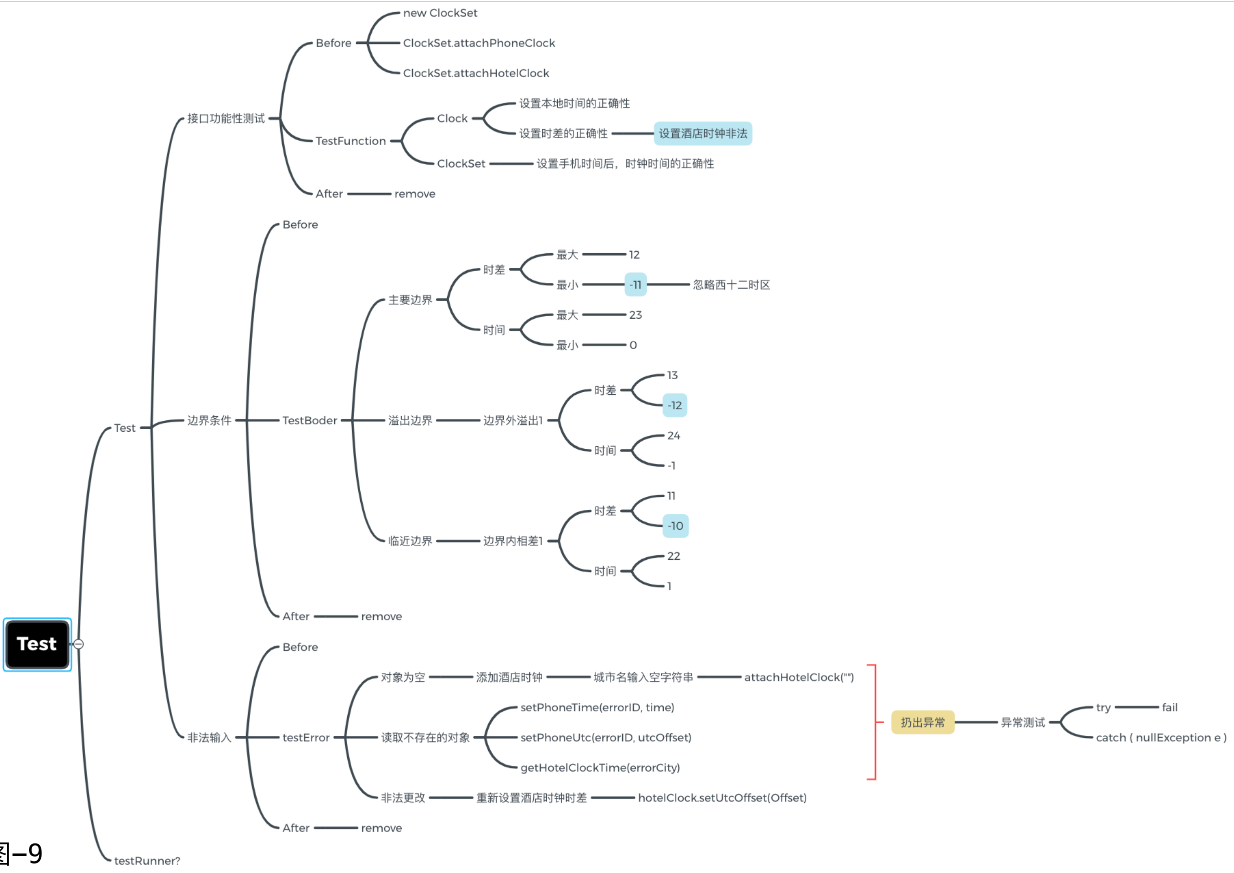


```
The initial state
--------Phone--------
ID      1 :       0
----Hotel Clock----
Beijing   :       0
London    :       0
NewYork   :       0
Moscow    :       0
Sydney    :       0
Set the phone's time to 21
--------Phone--------
ID      1 :      21
----Hotel Clock----
Beijing   :      21
London    :      13
NewYork   :       8
Moscow    :      17
Sydney    :      23
Set the phone's time zone Offset to 7
--------Phone--------
ID      1 :      12
----Hotel Clock----
Beijing   :      21
London    :      13
NewYork   :       8
Moscow    :      17
Sydney    :      23
```

图–8

二、基于自动化测试框架 Junit

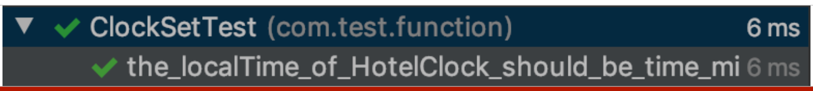
测试逻辑块设计思维导图



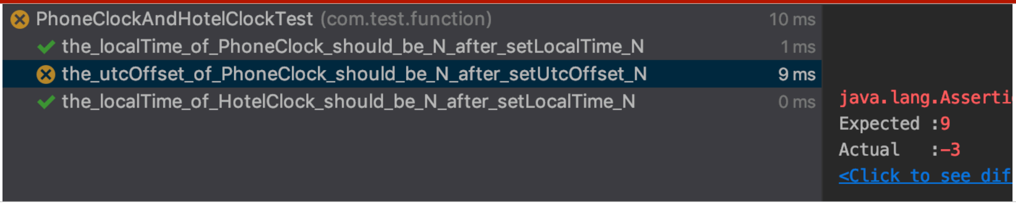
图–9

注：测试方法命名模版采用Behaviour-Driven development思想，使句子自然地命名测试方法

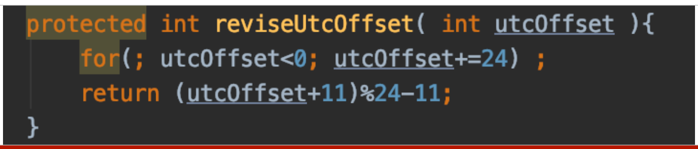
成功执行测试示例

图–10

```
▼ ✓ ClockSetTest (com.test.function)                          6 ms
    ✓ the_localTime_of_HotelClock_should_be_time_mi 6 ms
```

ISSUE 4

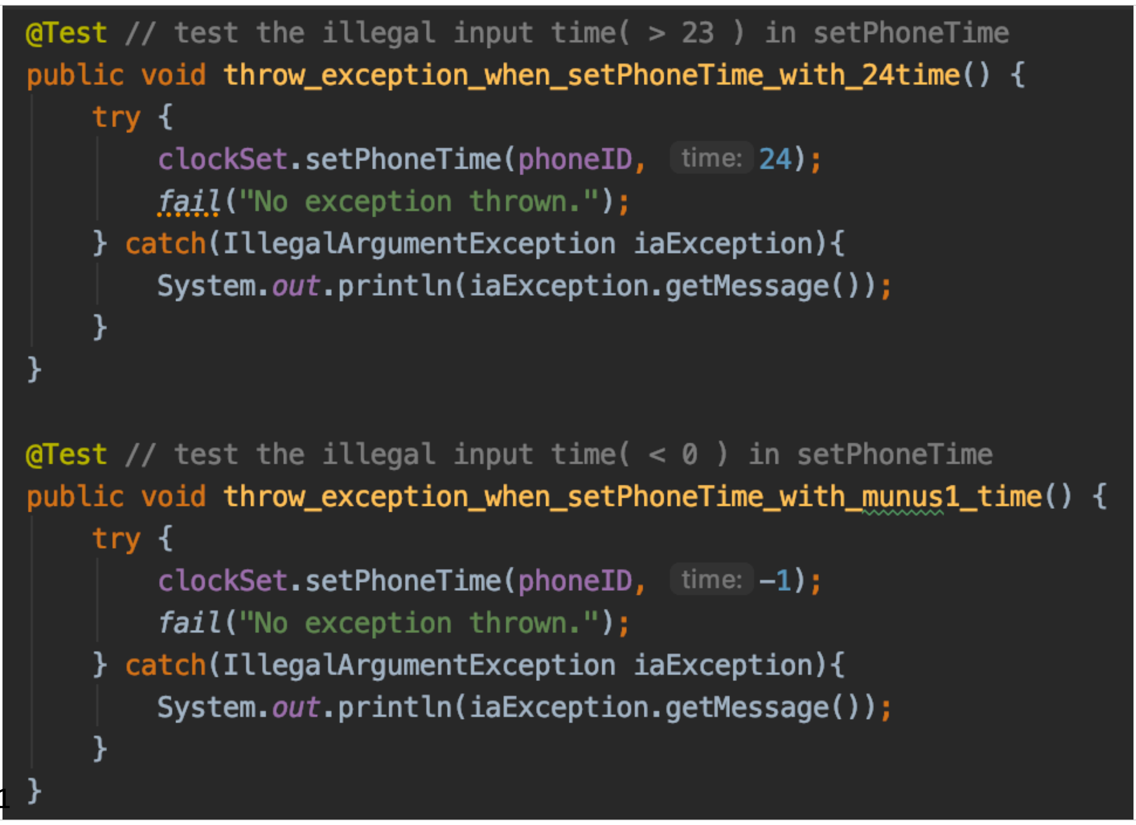

```
ⓧ PhoneClockAndHotelClockTest (com.test.function)                       10 ms
   ✓ the_localTime_of_PhoneClock_should_be_N_after_setLocalTime_N         1 ms
   ⓧ the_utcOffset_of_PhoneClock_should_be_N_after_setUtcOffset_N         9 ms
   ✓ the_localTime_of_HotelClock_should_be_N_after_setLocalTime_N         0 ms
java.lang.Asserti
Expected :9
Actual   :-3
<Click to see dif
```
图–11

关于时间偏差的数据格式化有误,应限制输入时差范围为-11~12
创建 CLock.reviseUtcOffset(int Offset) 函数，并在传入时差参数时调用


```java
protected int reviseUtcOffset( int utcOffset ){
    for(; utcOffset<0; utcOffset+=24) ;
    return (utcOffset+11)%24-11;
}
```

ISSUE 5

测试设计不周全
1. 在 ClockSet.setPhoneTime 的传入参数 time 超出 0-23 时，应该抛出 IllegalArgumentException 异常
2. 在 ClockSet.attachHotelClock() 的传入参数 city 为空字符串 ”” 时，应该抛出 IllegalArgumentException 异常

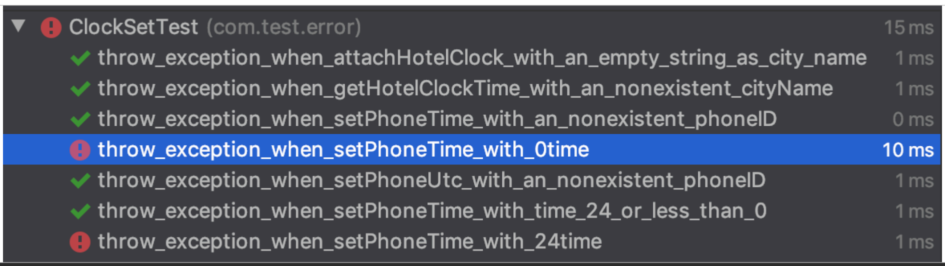


```
@Test // test the illegal input time( > 23 ) in setPhoneTime
public void throw_exception_when_setPhoneTime_with_24time() {
    try {
        clockSet.setPhoneTime(phoneID,  time: 24);
        fail("No exception thrown.");
    } catch(IllegalArgumentException iaException){
        System.out.println(iaException.getMessage());
    }
}

@Test // test the illegal input time( < 0 ) in setPhoneTime
public void throw_exception_when_setPhoneTime_with_munus1_time() {
    try {
        clockSet.setPhoneTime(phoneID,  time: -1);
        fail("No exception thrown.");
    } catch(IllegalArgumentException iaException){
        System.out.println(iaException.getMessage());
    }
}
```

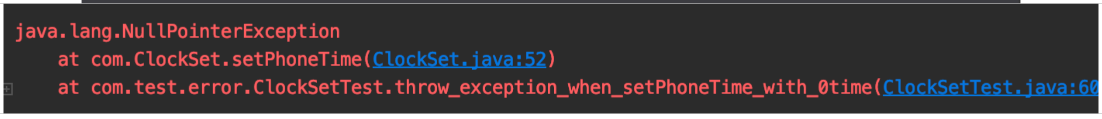
图-11

如图-12和图-13在运行时发现，在 ClockSet.attachHotelClock() 的传入参数 city 为空字符串 ”” 时，最初代码 抛出NullPointerException，与预期抛出 IllegalArugumentException 不符合，修改 attachHotelClock()中抛出异常类型



```
▼ ClockSetTest (com.test.error)                                      15 ms
    ✓ throw_exception_when_attachHotelClock_with_an_empty_string_as_city_name   1 ms
    ✓ throw_exception_when_getHotelClockTime_with_an_nonexistent_cityName       1 ms
    ✓ throw_exception_when_setPhoneTime_with_an_nonexistent_phoneID             0 ms
    ! throw_exception_when_setPhoneTime_with_0time                             10 ms
    ✓ throw_exception_when_setPhoneUtc_with_an_nonexistent_phoneID              1 ms
    ✓ throw_exception_when_setPhoneTime_with_time_24_or_less_than_0             1 ms
    ! throw_exception_when_setPhoneTime_with_24time                             1 ms
```

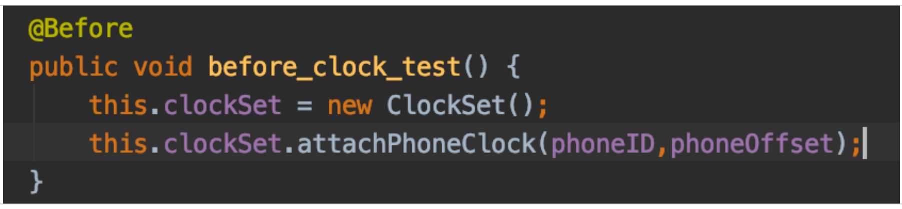
图-12



```
java.lang.NullPointerException
    at com.ClockSet.setPhoneTime(ClockSet.java:52)
    at com.test.error.ClockSetTest.throw_exception_when_setPhoneTime_with_0time(ClockSetTest.java:60
```

图-13

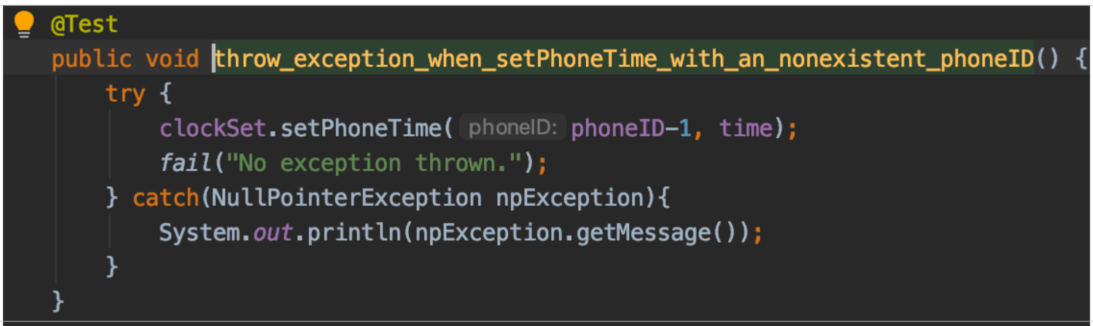
修改后发现扔抛出NullPointerException，检查后发现错误代码没有添加phoneClock(phoneId,phoneOffset) 对象，却引用这个对象，调用setPhoneTime传入time参数所以在测试前@Before中初始化phoneClock(phoneId,phoneOffset) 对象，如图-14所示



```
@Before
public void before_clock_test() {
    this.clockSet = new ClockSet();
    this.clockSet.attachPhoneClock(phoneID,phoneOffset);
}
```

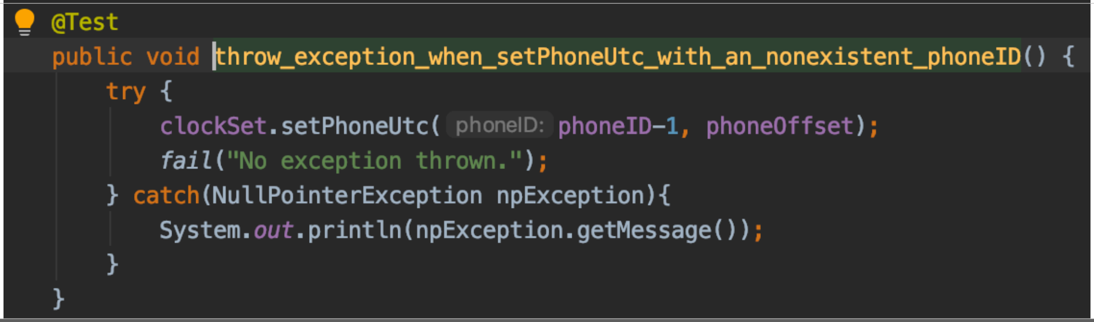
图-14

运行后发现原来通过的测试——测试setPhoneTime(phoneID,time)当不存在phoneID编号时抛出异常，执行失败，检查后发现是因为原来未初始化phoneClock(phoneId,phoneOffset）对象
所以修改测试变量如图-15和图-16所示，使得phoneID-1不存在于手机时钟列表phoneClockSet中



```
@Test
public void throw_exception_when_setPhoneTime_with_an_nonexistent_phoneID() {
    try {
        clockSet.setPhoneTime( phoneID: phoneID-1, time);
        fail("No exception thrown.");
    } catch(NullPointerException npException){
        System.out.println(npException.getMessage());
    }
}
```

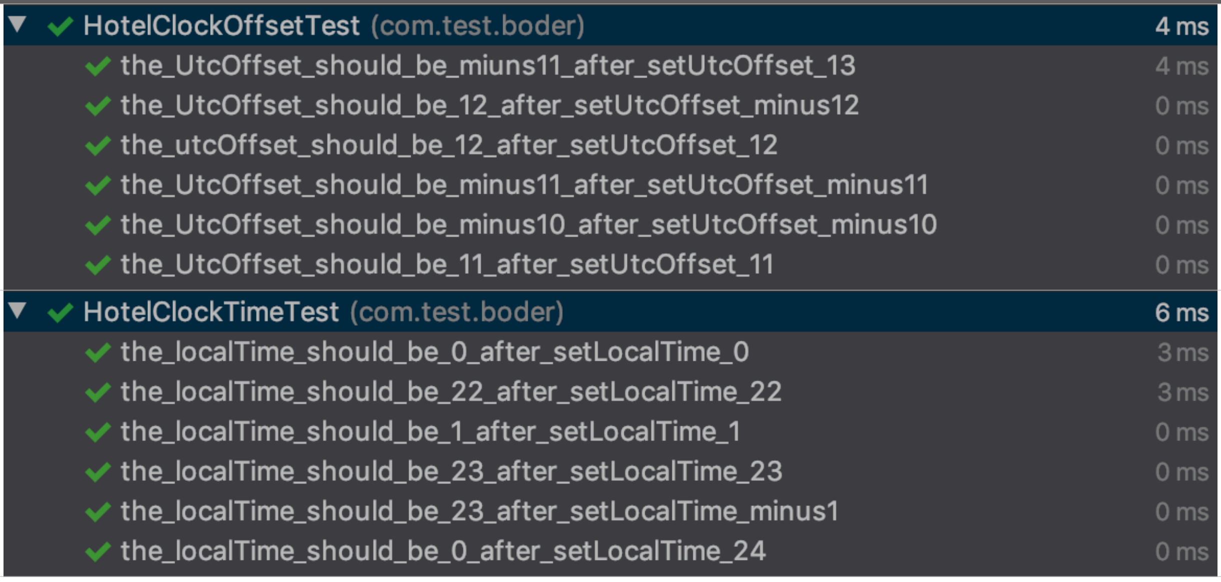
图-15



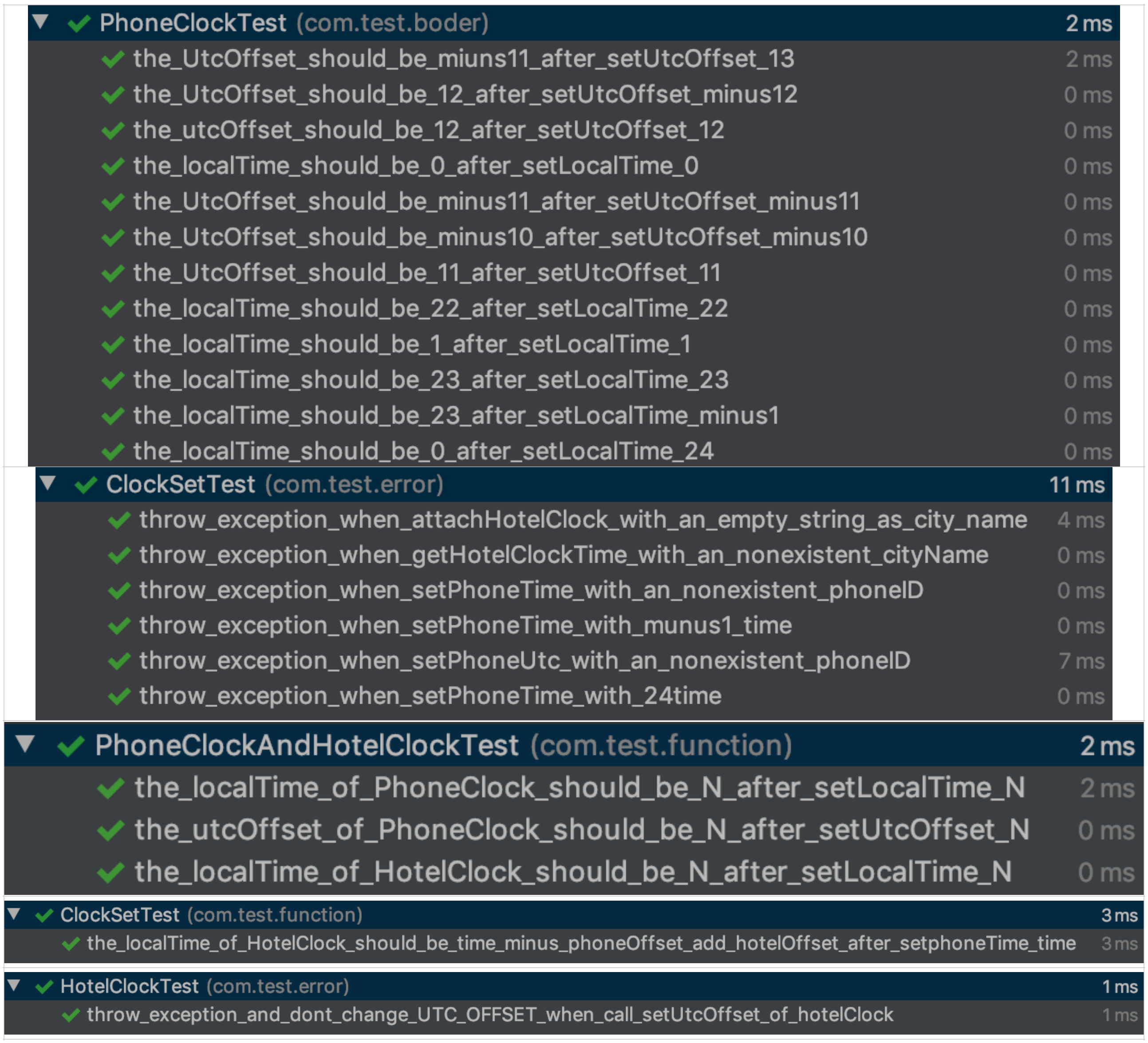
```
@Test
public void throw_exception_when_setPhoneUtc_with_an_nonexistent_phoneID() {
    try {
        clockSet.setPhoneUtc( phoneID: phoneID-1, phoneOffset);
        fail("No exception thrown.");
    } catch(NullPointerException npException){
        System.out.println(npException.getMessage());
    }
}
```

图-16

所有测试均通过



```
▼ ✓ HotelClockOffsetTest (com.test.boder)                          4 ms
    ✓ the_UtcOffset_should_be_miuns11_after_setUtcOffset_13        4 ms
    ✓ the_UtcOffset_should_be_12_after_setUtcOffset_minus12        0 ms
    ✓ the_utcOffset_should_be_12_after_setUtcOffset_12             0 ms
    ✓ the_UtcOffset_should_be_minus11_after_setUtcOffset_minus11   0 ms
    ✓ the_UtcOffset_should_be_minus10_after_setUtcOffset_minus10   0 ms
    ✓ the_UtcOffset_should_be_11_after_setUtcOffset_11             0 ms
▼ ✓ HotelClockTimeTest (com.test.boder)                            6 ms
    ✓ the_localTime_should_be_0_after_setLocalTime_0               3 ms
    ✓ the_localTime_should_be_22_after_setLocalTime_22             3 ms
    ✓ the_localTime_should_be_1_after_setLocalTime_1               0 ms
    ✓ the_localTime_should_be_23_after_setLocalTime_23             0 ms
    ✓ the_localTime_should_be_23_after_setLocalTime_minus1         0 ms
    ✓ the_localTime_should_be_0_after_setLocalTime_24              0 ms
```



```
▼ ✓ PhoneClockTest (com.test.boder)                                              2 ms
      ✓ the_UtcOffset_should_be_miuns11_after_setUtcOffset_13                    2 ms
      ✓ the_UtcOffset_should_be_12_after_setUtcOffset_minus12                    0 ms
      ✓ the_utcOffset_should_be_12_after_setUtcOffset_12                         0 ms
      ✓ the_localTime_should_be_0_after_setLocalTime_0                           0 ms
      ✓ the_UtcOffset_should_be_minus11_after_setUtcOffset_minus11               0 ms
      ✓ the_UtcOffset_should_be_minus10_after_setUtcOffset_minus10               0 ms
      ✓ the_UtcOffset_should_be_11_after_setUtcOffset_11                         0 ms
      ✓ the_localTime_should_be_22_after_setLocalTime_22                         0 ms
      ✓ the_localTime_should_be_1_after_setLocalTime_1                           0 ms
      ✓ the_localTime_should_be_23_after_setLocalTime_23                         0 ms
      ✓ the_localTime_should_be_23_after_setLocalTime_minus1                     0 ms
      ✓ the_localTime_should_be_0_after_setLocalTime_24                          0 ms
▼ ✓ ClockSetTest (com.test.error)                                               11 ms
      ✓ throw_exception_when_attachHotelClock_with_an_empty_string_as_city_name  4 ms
      ✓ throw_exception_when_getHotelClockTime_with_an_nonexistent_cityName      0 ms
      ✓ throw_exception_when_setPhoneTime_with_an_nonexistent_phoneID            0 ms
      ✓ throw_exception_when_setPhoneTime_with_munus1_time                       0 ms
      ✓ throw_exception_when_setPhoneUtc_with_an_nonexistent_phoneID             7 ms
      ✓ throw_exception_when_setPhoneTime_with_24time                            0 ms
▼ ✓ PhoneClockAndHotelClockTest (com.test.function)                              2 ms
      ✓ the_localTime_of_PhoneClock_should_be_N_after_setLocalTime_N             2 ms
      ✓ the_utcOffset_of_PhoneClock_should_be_N_after_setUtcOffset_N             0 ms
      ✓ the_localTime_of_HotelClock_should_be_N_after_setLocalTime_N             0 ms
▼ ✓ ClockSetTest (com.test.function)                                             3 ms
      ✓ the_localTime_of_HotelClock_should_be_time_minus_phoneOffset_add_hotelOffset_after_setphoneTime_time   3 ms
▼ ✓ HotelClockTest (com.test.error)                                              1 ms
      ✓ throw_exception_and_dont_change_UTC_OFFSET_when_call_setUtcOffset_of_hotelClock   1 ms
```