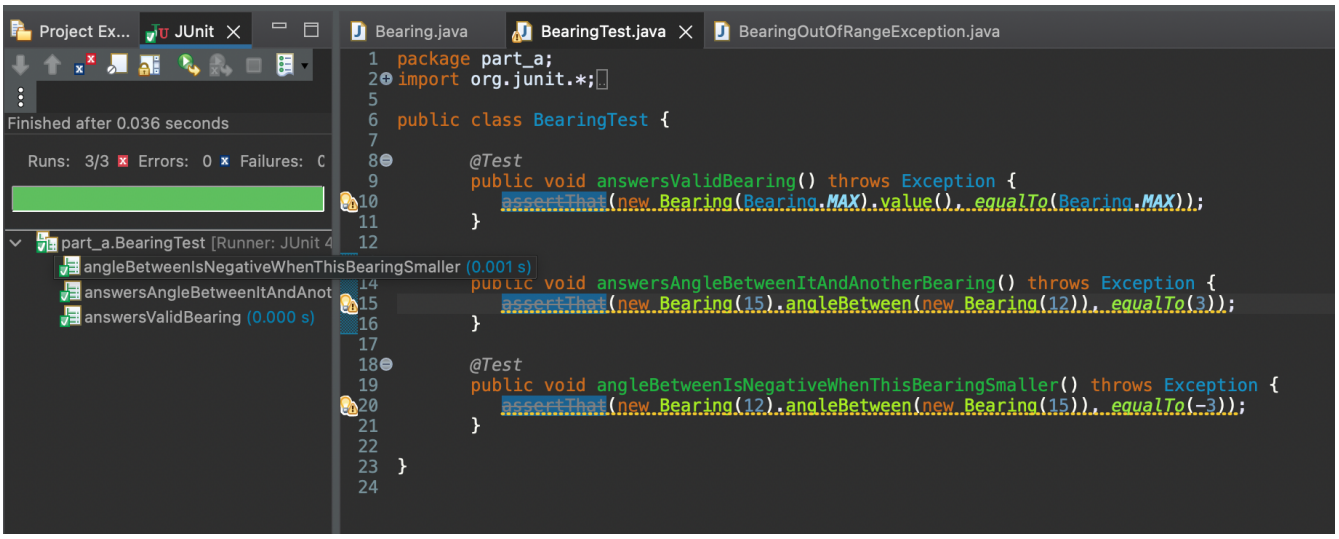
SE 3170: Lab 3a

Testing Boundary Conditions Using Exceptions Handling

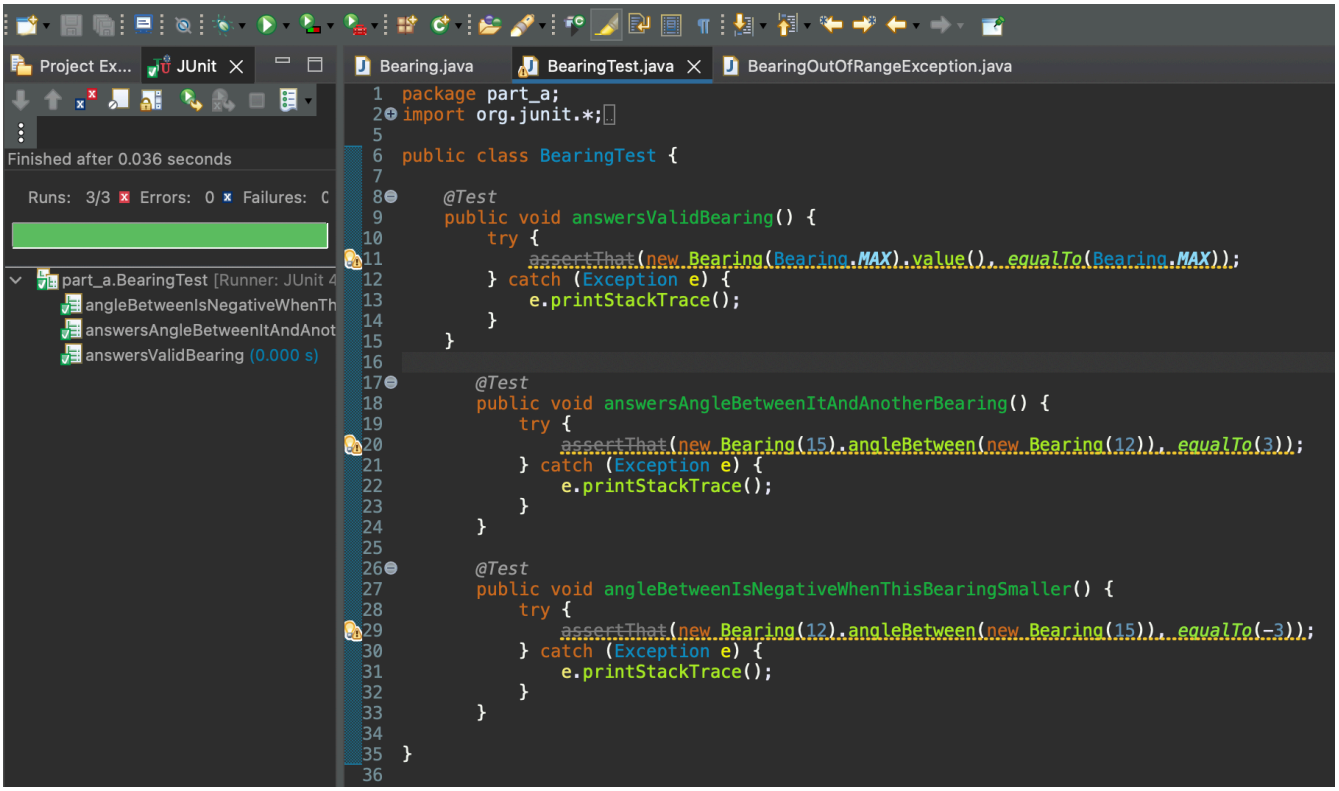
1. **TODO:**

Part 1

1- First, use **throws** method to fix the code. When you finish, take the screenshot of the passed result with your code.

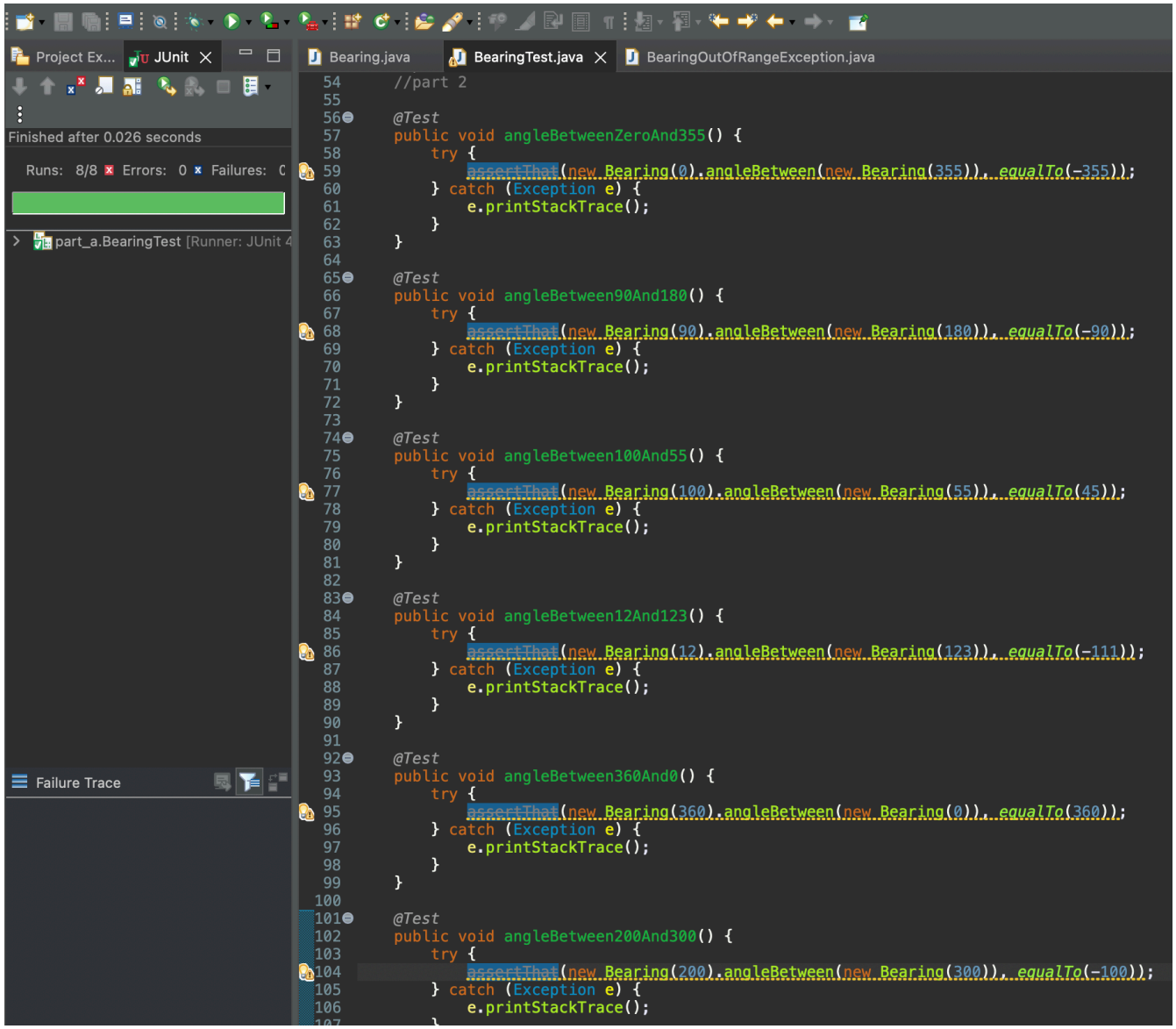


2- Next, Replace the throw exception with the "**try/catch**" method, run the code again and submit the screenshot of the **passed** result with your code

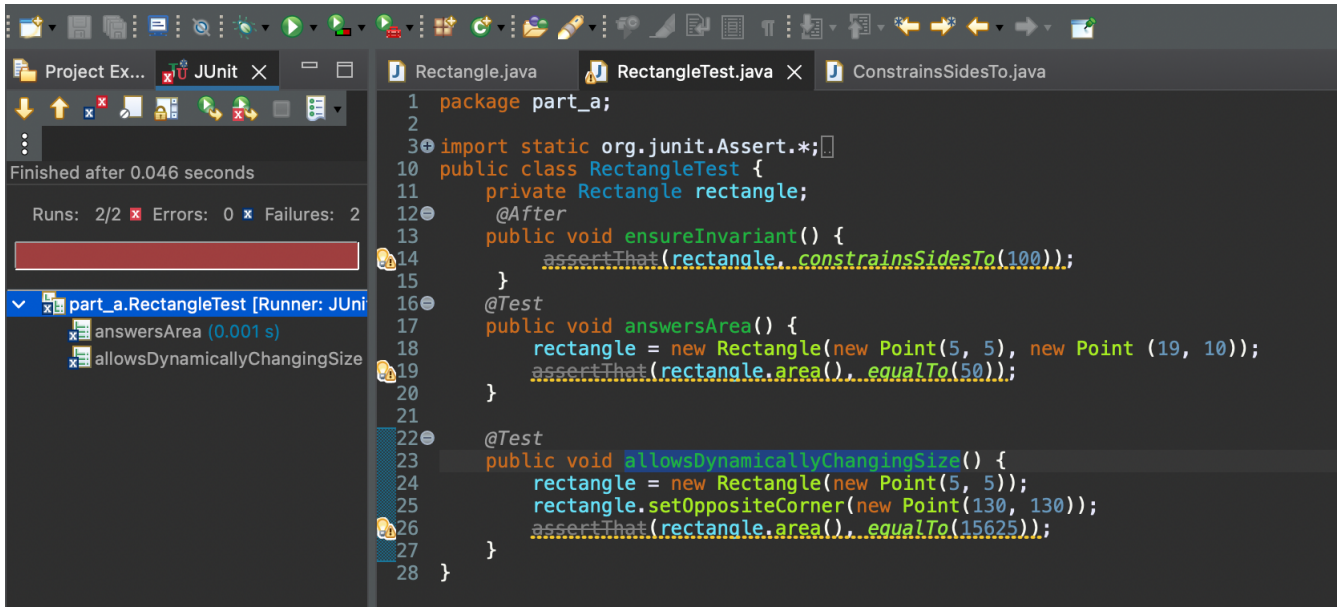


2. **TODO:**

Write 8 test cases

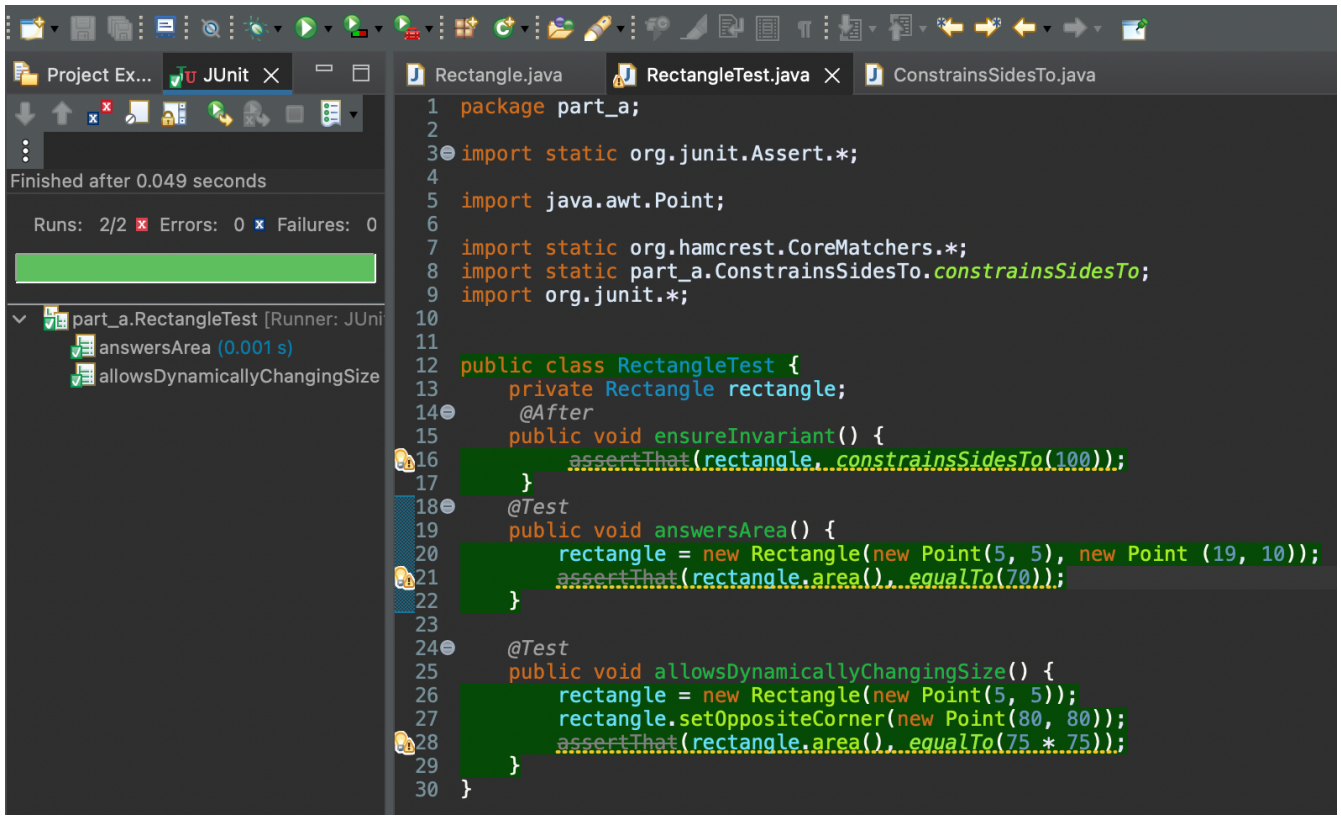


TODO:2



Yes, there are errors!

3. **TODO:** Fix the error(s) of the code and run the code again.



4. **TODO:**
   Answer the following questions:
   1. What is throw exception and how does it fix the code?

-The throw keyword is used to manually generate an exception when an error condition occurs.
-It stops the execution of the program unless handled by try-catch.
-In the lab, it helps ensure that invalid values (e.g., out-of-range bearings) do not proceed, preventing incorrect calculations.

   2. What is try-catch method and how does it fix the code

-try-catch is used to handle exceptions gracefully instead of stopping execution.
-It allows the program to continue running by catching errors and executing alternative logic.
-In the lab, it ensures that even if an exception occurs, the program can recover instead of crashing.

   3. Is there any difference between throw exception and try-catch method? If yes, explain.

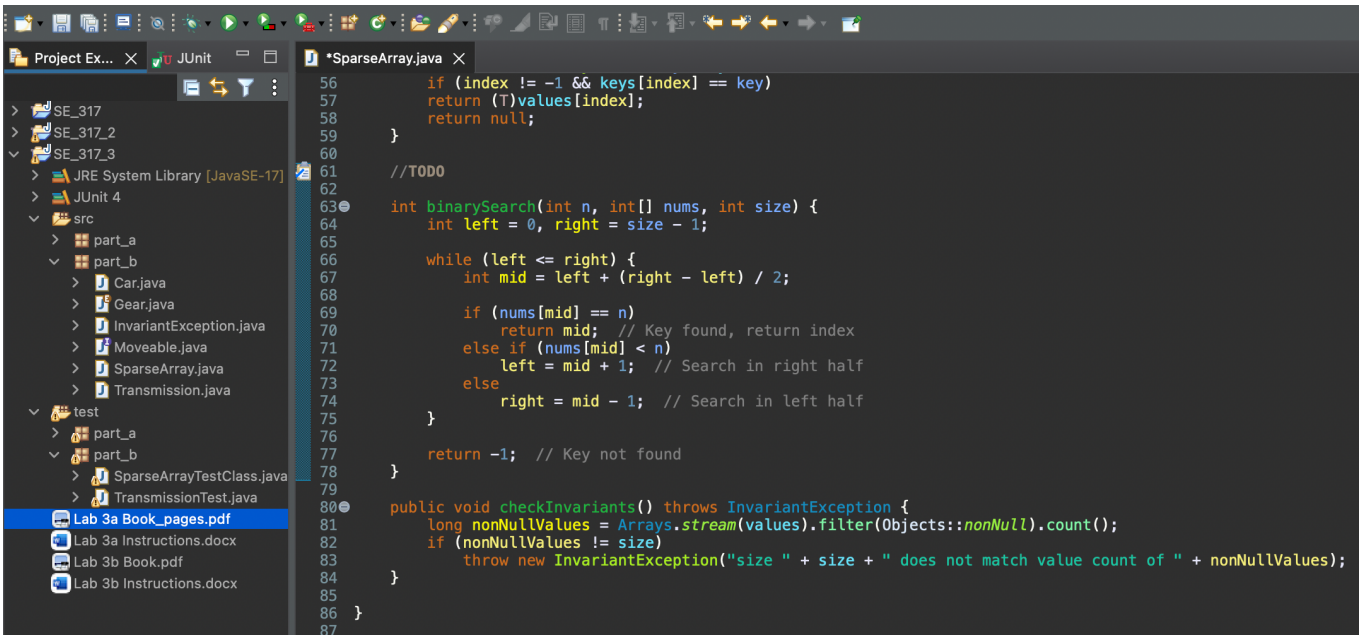
Yes, there is a difference:

-throw is used to generate an exception when something goes wrong.
-try-catch is used to handle an exception and prevent a program crash.
-throw is used inside a method to reject invalid input, while try-catch is used outside the method to handle errors and continue execution.

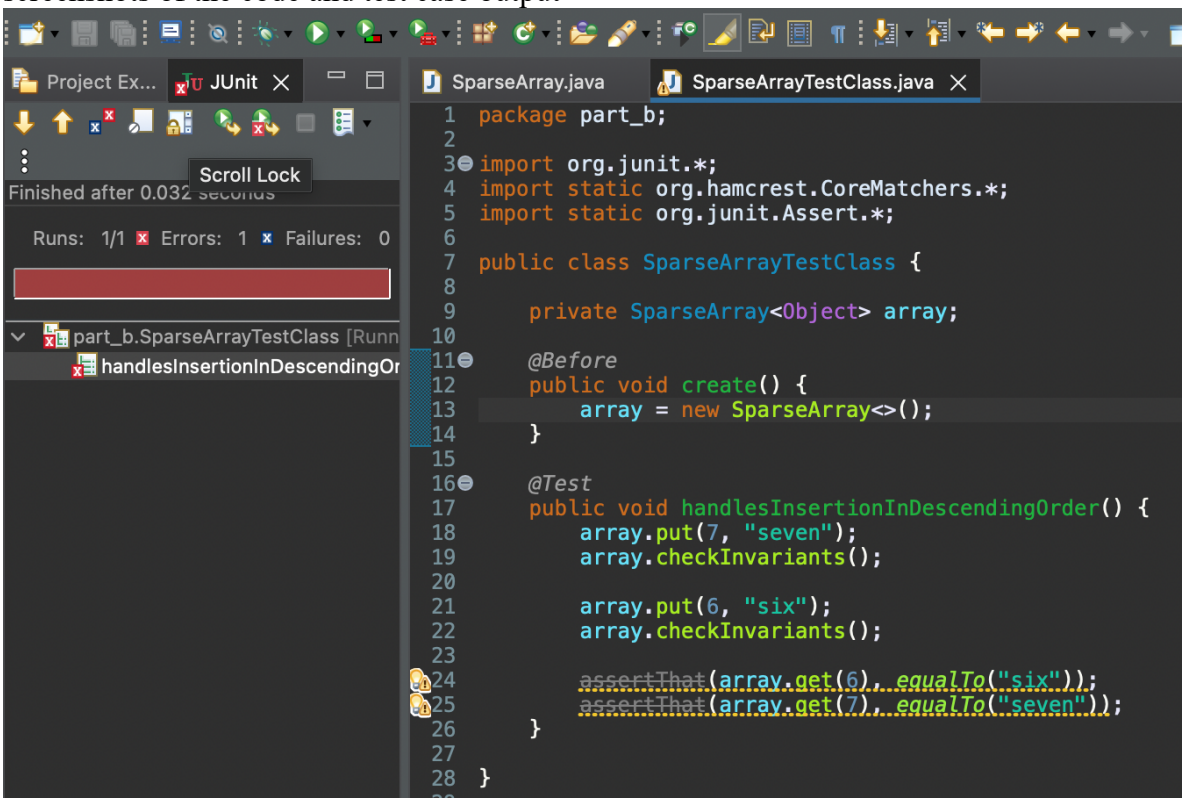
SE 3170: Lab 3b

Testing Ranges by Embedding Invariant Methods

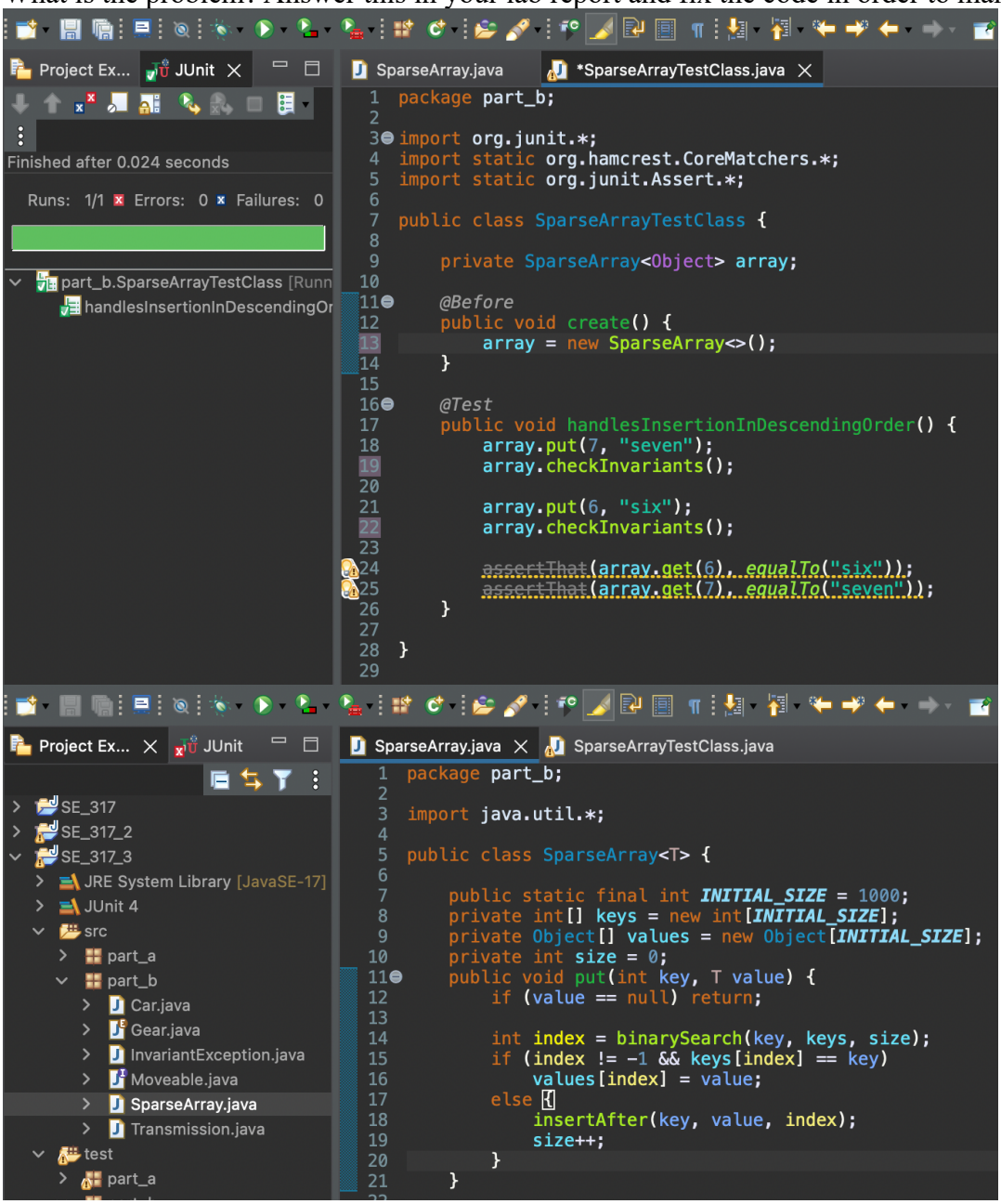
2- TODO 1: Implement the iterative Binary search function in SparseArray.java and take screenshot of the code



4- TODO 2: Write the following new test class in SparseArrayTestClass.java. Run the code, and take screenshots of the code and test case output



What is the problem? Answer this in your lab report and fix the code in order to make the test pass.

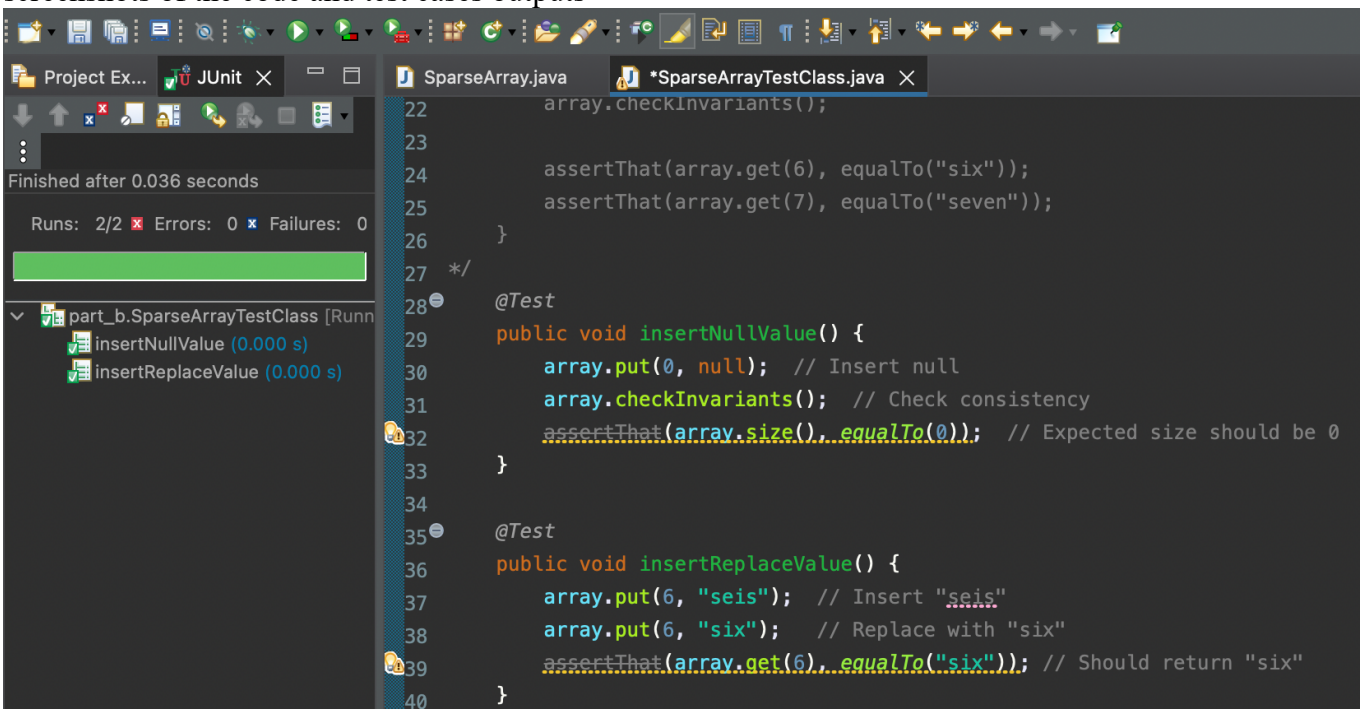


What Was the Issue?

-The size was not updating correctly when inserting new elements.
-checkInvariants() counted more non-null values than size, causing InvariantException.

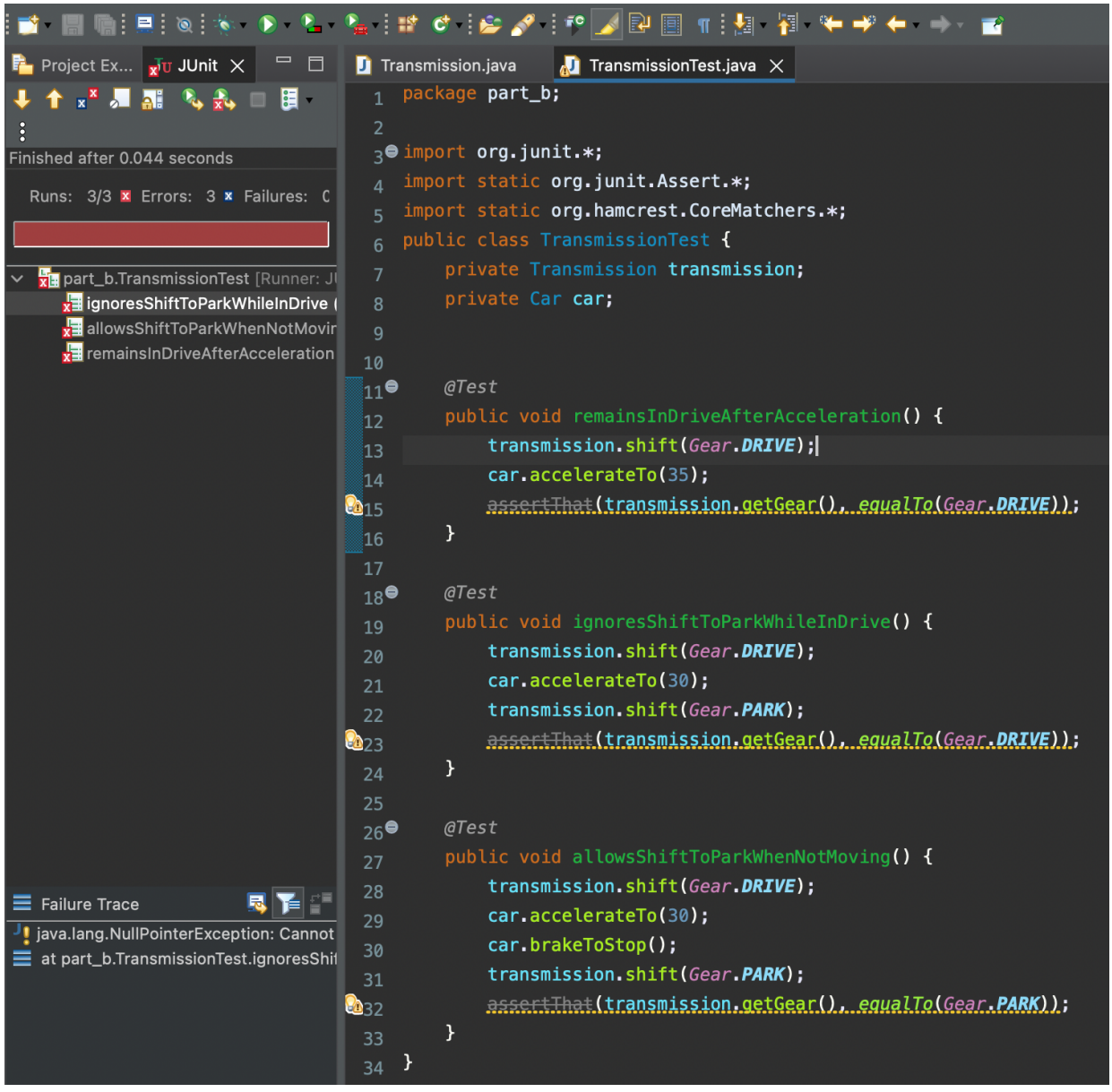
How Was It Fixed?
-We increment size++ only when a new key-value pair is inserted.

5- TODO 3: Write two test cases for the following conditions. Run the 2 test codes, and take screenshots of the code and test cases outputs

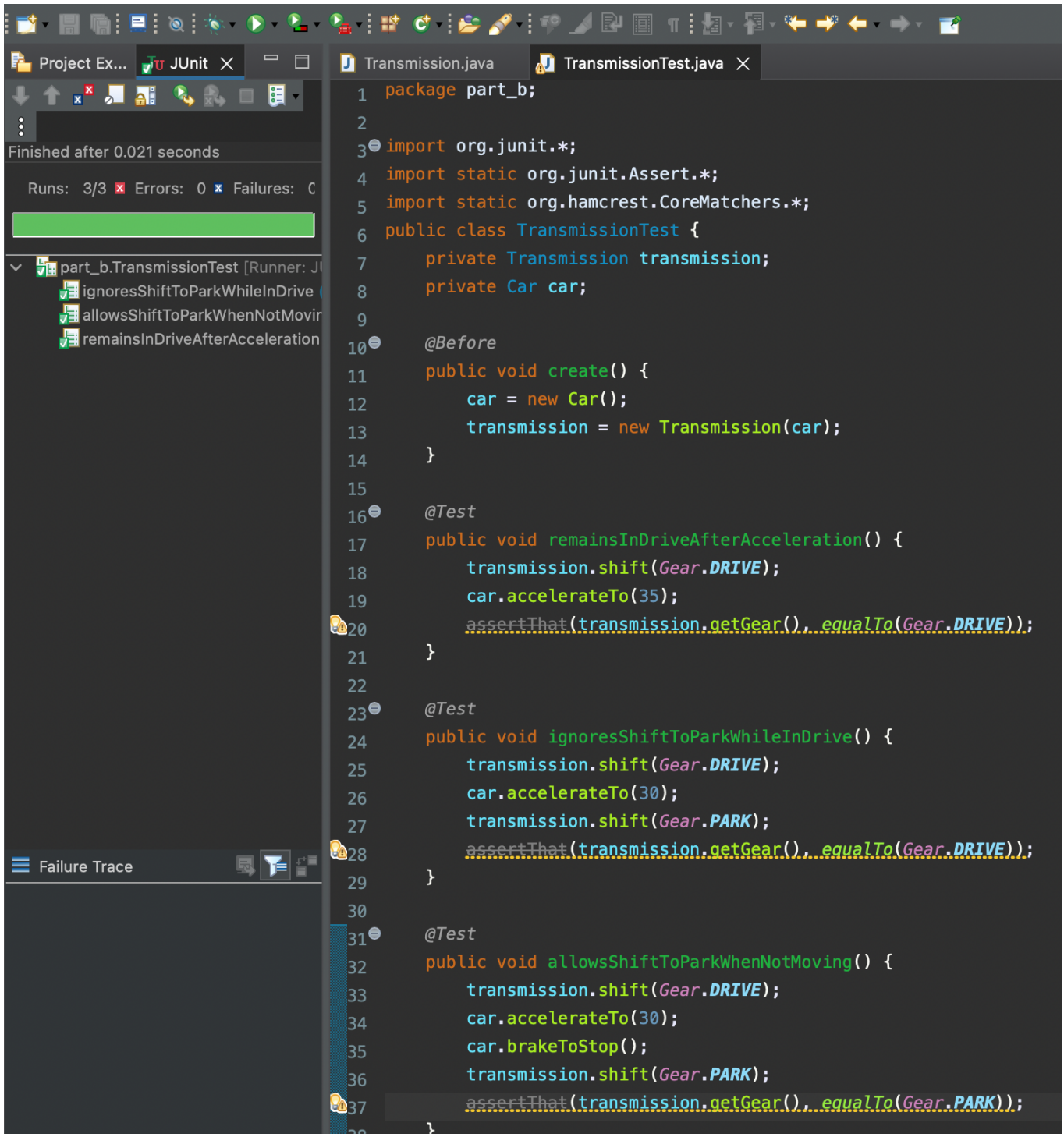


Part 2
Correct Reference (Correct Initial State)

6- TODO 4: Inspect and run the following TransmissionTest code. Take a screenshot of the code and output



7- - TODO 5: The code above will not run. You will need to fix the test code as follows and take a screenshot:



Part 3
Submission

1. Answer to “what does checkInvariants () method do?”

-The checkInvariants() method ensures data consistency in the SparseArray class.
-It counts the non-null values stored in the array and compares them to the size variable.
-If there is a mismatch, it throws an InvariantException to prevent incorrect data tracking.
-This helps detect errors where the internal size does not correctly reflect the actual number of stored elements.

2. Answer to “what Transmission.Java class does?”

-The Transmission class controls a car's gear shifting behavior.
-It ensures that dangerous or invalid gear shifts are prevented:
    --If the car is moving, it ignores a shift to Park.
    --If the car is stopped, it allows shifting to Park.
-It maintains the correct state of the transmission based on the car's speed.
-It mimics real-world transmission behavior by enforcing gear shift constraints.