



10 September 2025

PROG6112 Assignment

ST10470282



Keenan Mouton

VARSITY COLLEGE CAPE TOWN, BCA1, GR3

Table of Contents

Question 1: Code.....	2
Output for Question 1:.....	10
Unit Test for Question 1:	13
Test Results of the Unit Tests for Question 1:.....	15
Question 2 Code:.....	19
Output for Question 2:.....	27
Unit Test for Question 2 (Class Bank Account):.....	30
Unit Test Results for Bank Account:	31
Unit Test for class Savings Account:	35
Unit Test Results for class Savings Account:.....	36
Unit Test for class Check Account:	37
Unit Test Results for class Check Account:	38
Github Repository link:.....	39

Question 1: Code

```
Source History
1
2 package st10470282_prog6112_assignment;
3 import java.util.Scanner;
4 import java.util.ArrayList;
5
6 public class ST10470282_PROG6112_Assignment {
7
8
9     public static void main(String[] args) {
10         //Called the class Series into the main method by creating an object called MySeries
11         Series mySeries = new Series();
12
13         //Created an arraylist to store the values with SeriesModel objects and the variable name is serieslis
14         ArrayList<SeriesModel> seriesList = new ArrayList<>();
15
16         //Initialized a Scanner with object called Scan
17         Scanner Scan = new Scanner(System.in);
18
19         //Calling the method FirstMessage into the main method from the class series
20         mySeries.firstMessage();
21
22         //Saves what the user wants to do in the variable UserChoice
23         int userChoice = Scan.nextInt();
24         Scan.nextLine();
25
26         //If statement created to check if the user wants to enter the program or exit
27         if (userChoice != 1) {
28             System.out.println("Exiting Application");
29             return;
30         }
31     }
```

```
Source History
32 //Boolean with variable Application created. Set to true
33 //Runs the menu while true
34 boolean Application = true;
35 while (Application) {
36     //Calling the method menu from class Series
37     mySeries.menu();
38
39     //Saves what the user wants to do in the menu under the variable MenuChoice
40     int MenuChoice = Scan.nextInt();
41     Scan.nextLine();
42
43     //Created a switch statement
44     //Passed MenuChoice as a argument into the switch statement
45     //so that the program can run based on the users choice
46     //ChatGPT
47     //Question: how to create a switch statement with cases
48     //Answer: Lines 50-54
49     //URL: https://chatgpt.com/c/68b9fa87-4abc-8330-97fb-9ccae1e20c2d
50     switch (MenuChoice) {
51         //At each case I got the method from the class Series and passed Scan and seriesList as an argument
52         case 1:
53             mySeries.captureSeries(Scan, seriesList);
54             break;
55
56         case 2:
57             mySeries.searchSeries(Scan, seriesList);
58             break;
59
60         case 3:
61             mySeries.updateSeries(Scan, seriesList);
62             break;
63     }
```

```
Source History
68         case 5:
69             mySeries.seriesReport(Scan, seriesList);
70             break;
71
72         case 6:
73             mySeries.exitSeriesApplication();
74             Application = false;
75             continue;
76         default:
77             //If the user does not type 1-6 it will display this error
78             System.out.println("Option is invalid!");
79             System.out.println(" ");
80     }
81
82     //After every case except the exit one it will ask if user wants to relaunch menu or exit
83     System.out.println("*****");
84     System.out.println("Press (1) to launch menu or any other key to exit");
85     //checks that the user only selects 1 for the first message otherwise it will exit the application
86     int Choice;
87     if (Scan.hasNextInt()) {
88         Choice = Scan.nextInt();
89         Scan.nextLine();
90         if (Choice != 1) {
91             System.out.println("Exiting Application");
92             Application = false;
93         }
94     }
95     else {
96         //If user typed something other than a number
97         System.out.println("Exiting Application");
98         Application = false;
```

```
Source History
99     }
100 }
101 }
102 }
103
104 //Class called Series created
105 class Series {
106     //Created a method called FirstMessage
107     //This displays the first message that pops up once the program starts
108     public void firstMessage() {
109         System.out.println("Welcome to the TV series management application!");
110         System.out.println(" ");
111         System.out.println("LATEST SERIES - 2025");
112         System.out.println("*****");
113         System.out.println("Press (1) to launch menu or any other key to exit");
114     }
115
116     //Created a method called Menu
117     //This is the menu of the program
118     public void menu() {
119         System.out.println("Please select one of the following menu items: ");
120         System.out.println("(1) Capture a new series.");
121         System.out.println("(2) Search for a series.");
122         System.out.println("(3) Update series details");
123         System.out.println("(4) Delete a series.");
124         System.out.println("(5) Print series report - 2025");
125         System.out.println("(6) Exit application.");
126     }
127
128     //Created a method called captureSeries
129     //This allows the user to capture a new series at case 1
```

```
Source History
130 public void captureSeries(Scanner Scan, ArrayList<SeriesModel> seriesList) {
131     /*
132     Stack Overflow
133     Question: How to call a method in another class in java?
134     Answer: Line 137 as an example
135     URL: https://stackoverflow.com/questions/4593232/how-to-call-a-method-in-another-class-in-java
136     */
137     int seriesId = SeriesRestrictions.SeriesIdRestriction(Scan);
138     String seriesName = SeriesRestrictions.SeriesNameRestrictions(Scan);
139     int seriesAge = SeriesRestrictions.SeriesAgeRestrictions(Scan);
140     int seriesNumberOfEpisodes = SeriesRestrictions.SeriesEpisodesRestrictions(Scan);
141
142     SeriesModel newSeries = new SeriesModel(seriesId, seriesName, seriesAge, seriesNumberOfEpisodes);
143     seriesList.add(newSeries);
144
145     System.out.println(" ");
146     System.out.println("Series processed successfully!!!");
147     System.out.println("*****");
148 }
149
150 //Created a method called searchSeries
151 //This class handles case 2 that searches for the series from the series id
152 //It then searches for it and displays it if the series is found
153 //If not found it displays an error
154 public void searchSeries(Scanner Scan, ArrayList<SeriesModel> seriesList) {
155     System.out.print("Enter the series id to search: ");
156     int searchId = Scan.nextInt();
157     Scan.nextLine();
158
159     boolean found = false;
160     /*
```

```
Source History
161     GeeksforGeeks
162     //Java for loops
163     //Answer: line 166
164     URL: https://www.geeksforgeeks.org/java/loops-in-java/
165     */
166     for (SeriesModel series : seriesList) {
167         if (series.getSeriesId() == searchId) {
168             System.out.println("Found the Series!");
169             System.out.println("-----");
170             System.out.println(series.Printing());
171             found = true;
172             break;
173         }
174     }
175     if (!found) {
176         System.out.println("Series with series id: " + searchId + " was not found!");
177     }
178 }
179
180
181 // Created a method called updateSeries
182 //This allows the user to update the series details at case 3
183 //The user must first enter the series id and the system will verify if its already created
184 public void updateSeries(Scanner Scan, ArrayList<SeriesModel> seriesList) {
185     System.out.print("Enter the series id to update: ");
186     int updateId = Scan.nextInt();
187     Scan.nextLine();
188     boolean updated = false;
189
190     for (SeriesModel series : seriesList) {
191         if (series.getSeriesId() == updateId) {
```

```
Source History
192         updated = true;
193         System.out.println("Found the series! Current details:");
194         System.out.println(series.Printing());
195         //fetch the series restrictions from the seriesrestrictions class
196         /*
197         Stack Overflow
198         Question: How to call a method in another class in java?
199         Answer: Line 202 as an example
200         URL: https://stackoverflow.com/questions/4593232/how-to-call-a-method-in-another-class-in-java
201         */
202         String newName = SeriesRestrictions.SeriesNameRestrictions(Scan);
203         int newAge = SeriesRestrictions.SeriesAgeRestrictions(Scan);
204         int newEpisodes = SeriesRestrictions.SeriesEpisodesRestrictions(Scan);
205
206         series.setSeriesName(newName);
207         series.setSeriesAge(newAge);
208         series.setSeriesNumberOfEpisodes(newEpisodes);
209
210         System.out.println("Series successfully updated!");
211         break;
212     }
213 }
214 if (!updated) {
215     System.out.println("Series with series id " + updateId + " was not found!");
216 }
217 }
218
219 //Created a method called deleteSeries
220 //This allows the user to delete a series at case 4
221 //The user must first enter the series id and the system will verify if its already created
222 public void deleteSeries(Scanner Scan, ArrayList<SeriesModel> seriesList) {
```

```
Source History
223     System.out.print("Enter the series id to delete: ");
224     int deleteId = Scan.nextInt();
225     Scan.nextLine();
226     boolean found = false;
227     /*
228     GeeksforGeeks
229     Java for loops
230     Answer: line 233
231     URL: https://www.geeksforgeeks.org/java/loops-in-java/
232     */
233     for (int i = 0; i < seriesList.size(); i++) {
234         if (seriesList.get(i).getSeriesId() == deleteId) {
235             found = true;
236             System.out.println("Series found!");
237             System.out.println(seriesList.get(i).Printing());
238             System.out.println("Are you sure you want to delete series " + deleteId + "? Yes (y) to delete.");
239             String confirmation = Scan.nextLine();
240
241             if (confirmation.equalsIgnoreCase("y")) {
242                 seriesList.remove(i);
243                 System.out.println("Series with series id " + deleteId + " WAS deleted!");
244             }
245
246             else {
247                 System.out.println("Delete cancelled!");
248             }
249             break;
250         }
251     }
252
253     if (!found) {
```

```
Source History
254         System.out.println("Series with series id " + deleteId + " was not found!");
255     }
256 }
257
258 //Created a method called seriesReport
259 //This prints out all of the series in the program as a report at case 5
260 public void seriesReport(Scanner Scan, ArrayList<SeriesModel> seriesList) {
261     if (seriesList.isEmpty()) {
262         System.out.println("No series have been captured!");
263     }
264
265     else {
266         System.out.println("SERIES REPORT 2025: ");
267         int number = 1;
268         //GeeksforGeeks
269         //Java for loops
270         //Answer: line 271
271         //URL: https://www.geeksforgeeks.org/java/loops-in-java/
272         for (SeriesModel series : seriesList) {
273             System.out.println("-----");
274             System.out.println("Series " + number);
275             System.out.println("-----");
276             System.out.println(series.Printing());
277             number++;
278         }
279     }
280 }
281
282 }
283
284 //Created a method called exitSeriesApplication
```

```
Source History
285 //This exits the application at case 6
286 public void exitSeriesApplication() {
287     System.out.println("-----");
288     System.out.println("Bye see you soon!");
289 }
290
291 }
292
293
294 //created a new class called SeriesRestrictions
295 //This class consists of different methods
296 //Each method has restrictions for a certain part of the series
297 //For example: restrictions on the series age: Only allowed to be 2,18 or between them
298 //It also asks the questions to the user with an error if the user enters incorrect information
299 class SeriesRestrictions {
300     public static int SeriesIdRestriction(Scanner Scan) {
301         //restrictions on the seriesid
302         //checks that the user enters a number
303         System.out.print("Enter the series ID (NUMBERS ONLY): ");
304         /*
305         GeeksforGeeks
306         While loops
307         Answer: Lines 310-316
308         URL: https://www.w3schools.com/java/java\_while\_loop.asp
309         */
310         while (!Scan.hasNextInt()) {
311             System.out.println("Invalid input! Please enter a number");
312             Scan.next();
313             System.out.print("Enter the series ID (NUMBERS ONLY): ");
314         }
315     }
316 }
```

```
Source History
316     return Scan.nextInt();
317 }
318
319 public static String SeriesNameRestrictions(Scanner Scan) {
320     //restrictions on the seriesname
321     //checks that the user does not leave the line empty
322     Scan.nextLine();
323     System.out.print("Enter series name: ");
324     String name = Scan.nextLine();
325     //ChatGPT
326     //Question: How do you give a while loop where it says a user cant leave a question open
327     //Answer: Lines 329-334
328     //URL: https://chatgpt.com/c/68b9f88f-1af4-8320-b75f-0b3124df1cbc
329     while (name.trim().isEmpty()) {
330         System.out.println("Series name cannot be empty!");
331         System.out.print("Enter series name: ");
332         name = Scan.nextLine();
333     }
334     return name;
335 }
336
337 public static int SeriesAgeRestrictions(Scanner Scan) {
338     //restrictions on the seriesage
339     //checks that the user enters a number and the number must be 2,18 or between them
340     int age;
341     System.out.print("Enter the series age restriction (2-18): ");
342     while (true) {
343         if (Scan.hasNextInt()) {
344             age = Scan.nextInt();
345             Scan.nextLine();
346             if (age >= 2 && age <= 18) {
```

```
Source History
347         return age;
348     }
349     else {
350         System.out.println("You have entered an incorrect series age!");
351         System.out.print("Please re-enter the series age: ");
352     }
353 }
354 }
355 else {
356     System.out.print("That's not a number! Please enter a number: ");
357     Scan.nextLine();
358 }
359 }
360 }
361
362 public static int SeriesEpisodesRestrictions(Scanner Scan) {
363     //restrictions on the seriesepisodes
364     //checks that the user enters a number
365     int episodes;
366     System.out.print("Enter the number of episodes: ");
367     while (true) {
368         if (Scan.hasNextInt()) {
369             episodes = Scan.nextInt();
370             Scan.nextLine();
371             return episodes;
372         }
373     }
374     else {
375         System.out.print("Invalid input! Please enter a number: ");
376         Scan.next();
377     }
```



```
Source History
378 }
379 }
380 }
381 }
382 }
383 //class seriesmodel created to get the series details and store them in the arraylist
384 class SeriesModel {
385     //private variables created that belongs to each object of the class
386     private int seriesId;
387     private String seriesName;
388     private int seriesAge;
389     private int seriesNumberOfEpisodes;
390     //created a constructor called SeriesModel
391     /*
392     W3Schools
393     Constructors
394     Answer: Lines 397-402
395     URL: https://www.w3schools.com/java/java\_constructors.asp
396     */
397     public SeriesModel(int seriesId, String seriesName, int seriesAge, int seriesNumberOfEpisodes) {
398         //Takes the values passed into the constructor and stores it in the object SeriesModel
399         this.seriesId = seriesId;
400         this.seriesName = seriesName;
401         this.seriesAge = seriesAge;
402         this.seriesNumberOfEpisodes = seriesNumberOfEpisodes;
403     }
404     //Creating getters to access the series details later
405     /*
406     GeeksforGeeks
407     Getters and Setters
408     Answer: 411-435
```

```
Source History
409 URL: https://www.geeksforgeeks.org/java/getter-and-setter-in-java/
410 /*
411     public int getSeriesId(){
412         return seriesId; }
413     public String getSeriesName(){
414         return seriesName; }
415     public int getSeriesAge(){
416         return seriesAge; }
417     public int getSeriesNumberOfEpisode(){
418         return seriesNumberOfEpisodes; }
419
420     //Setter created so that the user can update the series details
421     public void setSeriesAge(int seriesAge){
422         this.seriesAge = seriesAge;
423     }
424
425     public void setSeriesName(String seriesName){
426         this.seriesName = seriesName;
427     }
428
429     public void setSeriesNumberOfEpisodes(int seriesNumberOfEpisodes){
430         this.seriesNumberOfEpisodes = seriesNumberOfEpisodes;
431     }
432
433     public void setSeriesID(int seriesId){
434         this.seriesId = seriesId;
435     }
436     //Created a method called printing to print the details of the series
437     public String Printing() {
438         //printing method created to print out the series details
439         return "Series ID: " + seriesId + "\nSeries Name: " + seriesName
```

```
Source History
427 }
428
429 public void setSeriesNumberOfEpisodes(int seriesNumberOfEpisodes){
430     this.seriesNumberOfEpisodes = seriesNumberOfEpisodes;
431 }
432
433 public void setSeriesID(int seriesId){
434     this.seriesId = seriesId;
435 }
436 //Created a method called printing to print the details of the series
437 public String Printing() {
438     //printing method created to print out the series details
439     return "Series ID: " + seriesId + "\nSeries Name: " + seriesName
440         + "\nSeries Age restriction: " + seriesAge
441         + "\nNumber of episodes in " + seriesName + " : " + seriesNumberOfEpisodes;
442 }
443 }
444
445
446
447
```

Output for Question 1:

```
Output - ST10470282_PROG6112_Assignment (run) x
run:
Welcome to the TV series management application!

LATEST SERIES - 2025
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
1
Enter the series ID (NUMBERS ONLY): 01
Enter series name: Keenan Mouton
Enter the series age restriction (2-18): 18
Enter the number of episodes: 19

Series processed successfully!!!
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
```

```
Output - ST10470282_PROG6112_Assignment (run) x
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
1
Enter the series ID (NUMBERS ONLY): 02
Enter series name: Pretty Little Liars
Enter the series age restriction (2-18): 16
Enter the number of episodes: 102

Series processed successfully!!!
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
1
Enter the series ID (NUMBERS ONLY): 03
Enter series name: Marvel
Enter the series age restriction (2-18): 12
Enter the number of episodes: 70

Series processed successfully!!!
*****
```

```
Output - ST10470282_PROG6112_Assignment (run) x
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
2
Enter the series id to search: 02
Found the Series!
-----
Series ID: 2
Series Name: Pretty Little Liars
Series Age restriction: 16
Number of episodes in Pretty Little Liars : 102
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
3
```

```
Output - ST10470282_PROG6112_Assignment (run) x
Enter the series id to update: 03
Found the series! Current details:
Series ID: 3
Series Name: Marvel
Series Age restriction: 12
Number of episodes in Marvel : 70

Enter series name: Marvel 2
Enter the series age restriction (2-18): 12
Enter the number of episodes: 120
Series successfully updated!
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
4
Enter the series id to delete: 01
Series found!
Series ID: 1
Series Name: Keenan Mouton
Series Age restriction: 18
Number of episodes in Keenan Mouton : 19
Are you sure you want to delete series 1? Yes (y) to delete.
```

```
Output - ST10470282_PROG6112_Assignment (run) x
Are you sure you want to delete series 1? Yes (y) to delete.
Y
Series with series id 1 WAS deleted!
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
5
SERIES REPORT 2025:
-----
Series 1
-----
Series ID: 2
Series Name: Pretty Little Liars
Series Age restriction: 16
Number of episodes in Pretty Little Liars : 102
-----
Series 2
-----
Series ID: 3
Series Name: Marvel 2
Series Age restriction: 12
Number of episodes in Marvel 2 : 120
-----
408:20 INS
```

```
Output - ST10470282_PROG6112_Assignment (run) x
-----
Series 1
-----
Series ID: 2
Series Name: Pretty Little Liars
Series Age restriction: 16
Number of episodes in Pretty Little Liars : 102
-----
Series 2
-----
Series ID: 3
Series Name: Marvel 2
Series Age restriction: 12
Number of episodes in Marvel 2 : 120
-----
*****
Press (1) to launch menu or any other key to exit
1
Please select one of the following menu items:
(1) Capture a new series.
(2) Search for a series.
(3) Update series details
(4) Delete a series.
(5) Print series report - 2025
(6) Exit application.
6
-----
Bye see you soon!
BUILD SUCCESSFUL (total time: 7 minutes 16 seconds)
|
408:20 INS
```

Unit Test for Question 1:

```
Source History
1 package st10470282_prog6112_assignment;
2
3 import java.util.ArrayList;
4 import org.junit.Test;
5 import static org.junit.Assert.*;
6
7 public class SeriesTest {
8
9     @Test
10     public void TestSearchSeries() {
11         ArrayList<SeriesModel> seriesList = new ArrayList<>();
12         SeriesModel series = new SeriesModel(1, "Marvel", 18, 62);
13         seriesList.add(series);
14
15         boolean found = false;
16         for (SeriesModel s : seriesList) {
17             if (s.getSeriesId() == 1) {
18                 found = true;
19                 assertEquals("Marvel", s.getSeriesName());
20                 assertEquals(18, s.getSeriesAge());
21                 assertEquals(62, s.getSeriesNumberOfEpisode());
22             }
23         }
24         assertTrue(found);
25     }
26
27     @Test
28     public void TestSearchSeries_SeriesNotFound() {
29         ArrayList<SeriesModel> seriesList = new ArrayList<>();
30         seriesList.add(new SeriesModel(1, "Marvel", 18, 62));
31
32         boolean found = false;
```

```
Source History
33         for (SeriesModel s : seriesList) {
34             if (s.getSeriesId() == 99) {
35                 found = true;
36             }
37         }
38         assertFalse(found);
39     }
40
41     @Test
42     public void TestUpdateSeries() {
43         ArrayList<SeriesModel> seriesList = new ArrayList<>();
44         SeriesModel series = new SeriesModel(1, "Old Name", 16, 10);
45         seriesList.add(series);
46
47         // simulate update
48         for (SeriesModel s : seriesList) {
49             if (s.getSeriesId() == 1) {
50                 s.setSeriesName("New Name");
51                 s.setSeriesAge(18);
52                 s.setSeriesNumberOfEpisodes(20);
53             }
54         }
55
56         SeriesModel updated = seriesList.get(0);
57         assertEquals("New Name", updated.getSeriesName());
58         assertEquals(18, updated.getSeriesAge());
59         assertEquals(20, updated.getSeriesNumberOfEpisode());
60     }
61
62     @Test
63     public void TestDeleteSeries() {
64         ArrayList<SeriesModel> seriesList = new ArrayList<>();
```

```
Source History
64 ArrayList<SeriesModel> seriesList = new ArrayList<>();
65 seriesList.add(new SeriesModel(1, "Marvel", 18, 62));
66
67 boolean deleted = false;
68 for (int i = 0; i < seriesList.size(); i++) {
69     if (seriesList.get(i).getSeriesId() == 1) {
70         seriesList.remove(i);
71         deleted = true;
72         break;
73     }
74 }
75 assertTrue(deleted);
76 assertEquals(0, seriesList.size());
77
78
79 @Test
80 public void TestDeleteSeries_SeriesNotFound() {
81     ArrayList<SeriesModel> seriesList = new ArrayList<>();
82     seriesList.add(new SeriesModel(1, "Marvel", 18, 62));
83
84     boolean deleted = false;
85     for (int i = 0; i < seriesList.size(); i++) {
86         if (seriesList.get(i).getSeriesId() == 99) {
87             seriesList.remove(i);
88             deleted = true;
89             break;
90         }
91     }
92     assertFalse(deleted);
93     assertEquals(1, seriesList.size()); // still there
94 }
95
```

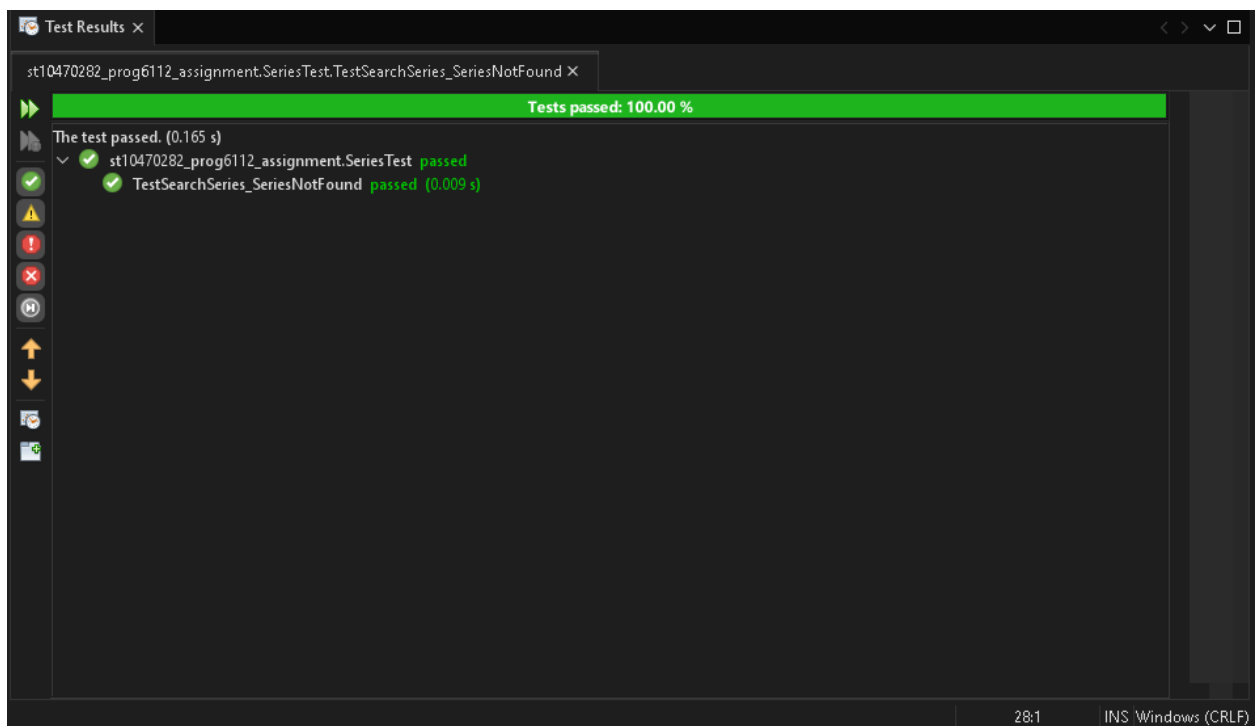
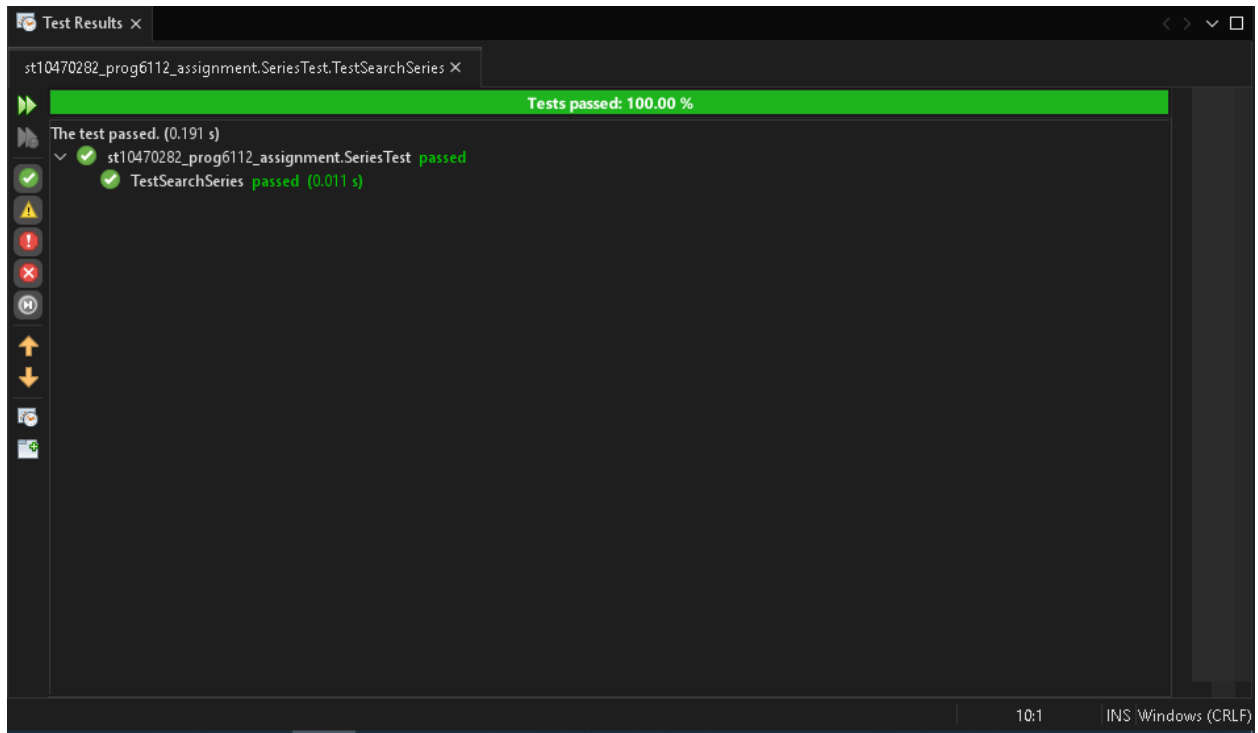
1:1 INS Windows (CRLF)

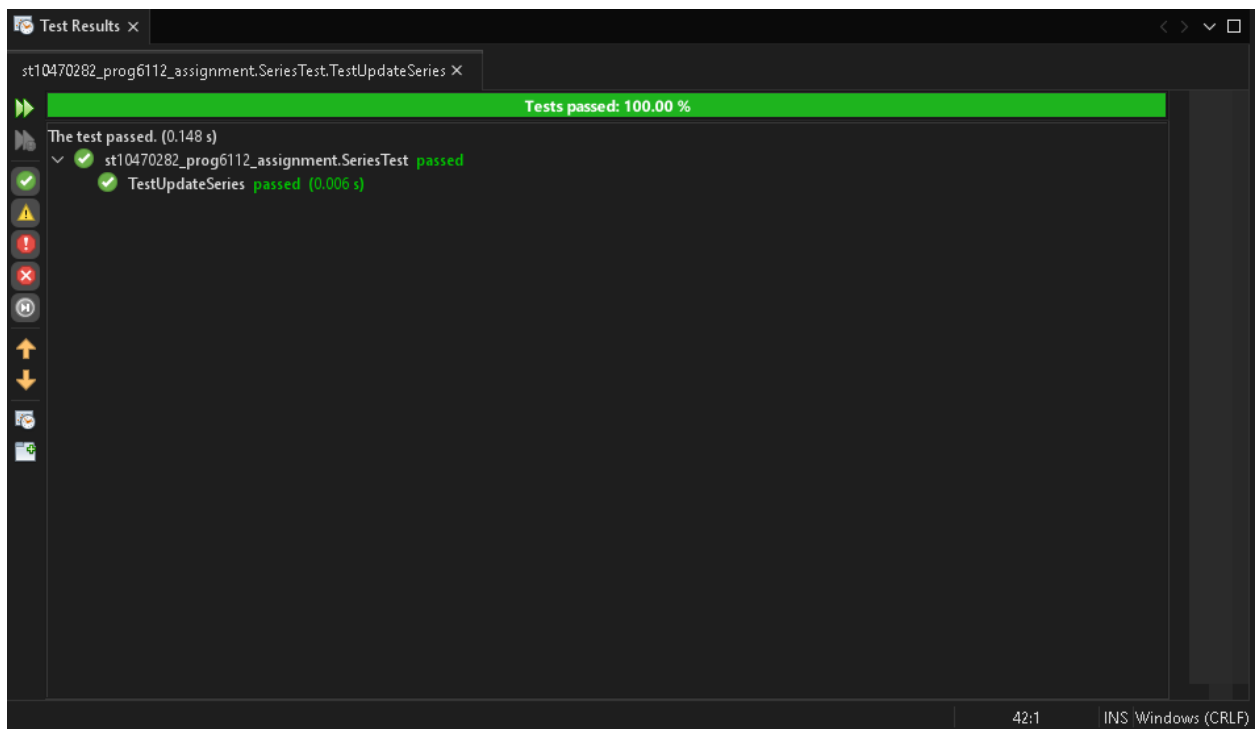
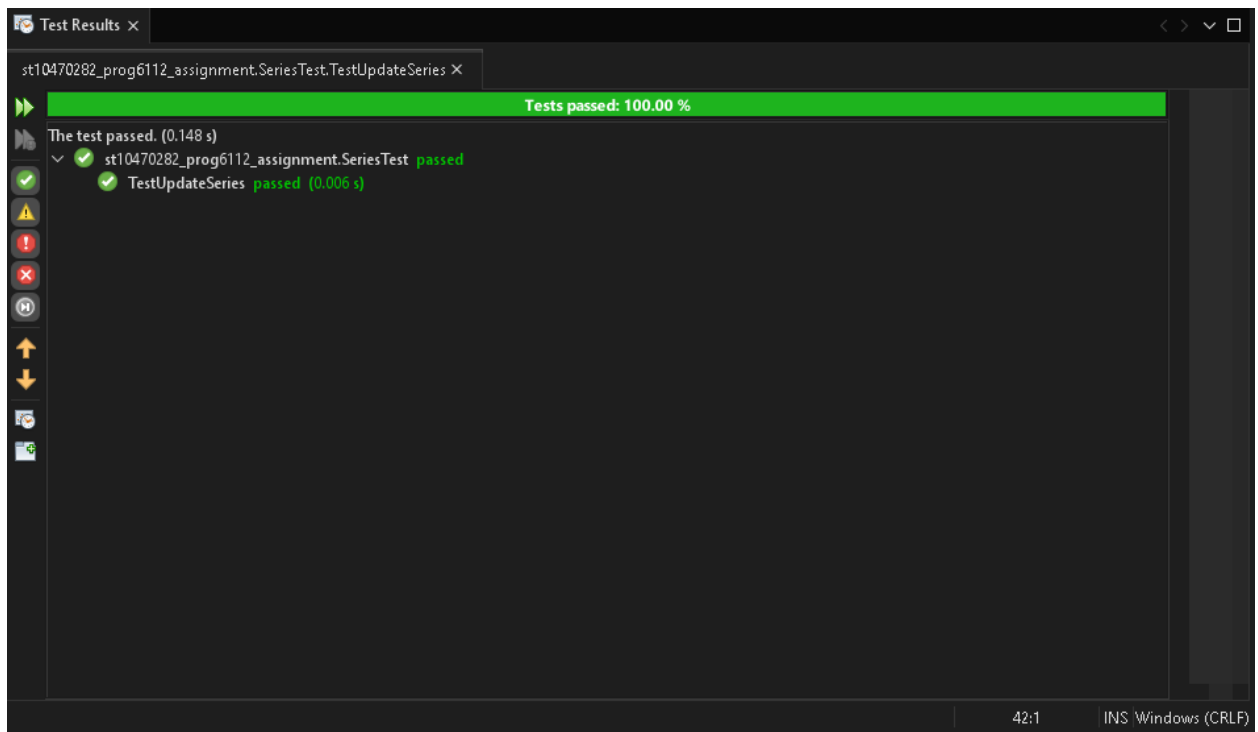
```
Source History
87     seriesList.remove(i);
88     deleted = true;
89     break;
90 }
91
92     }
93     assertFalse(deleted);
94     assertEquals(1, seriesList.size()); // still there
95 }
96
97 @Test
98 public void TestSeriesAgeRestriction_AgeValid() {
99     int validAge = 16;
100     assertTrue(validAge >= 2 && validAge <= 18);
101 }
102
103 @Test
104 public void TestSeriesAgeRestriction_SeriesAgeInvalid() {
105     int invalidAge = 25;
106     assertFalse(invalidAge >= 2 && invalidAge <= 18);
107 }

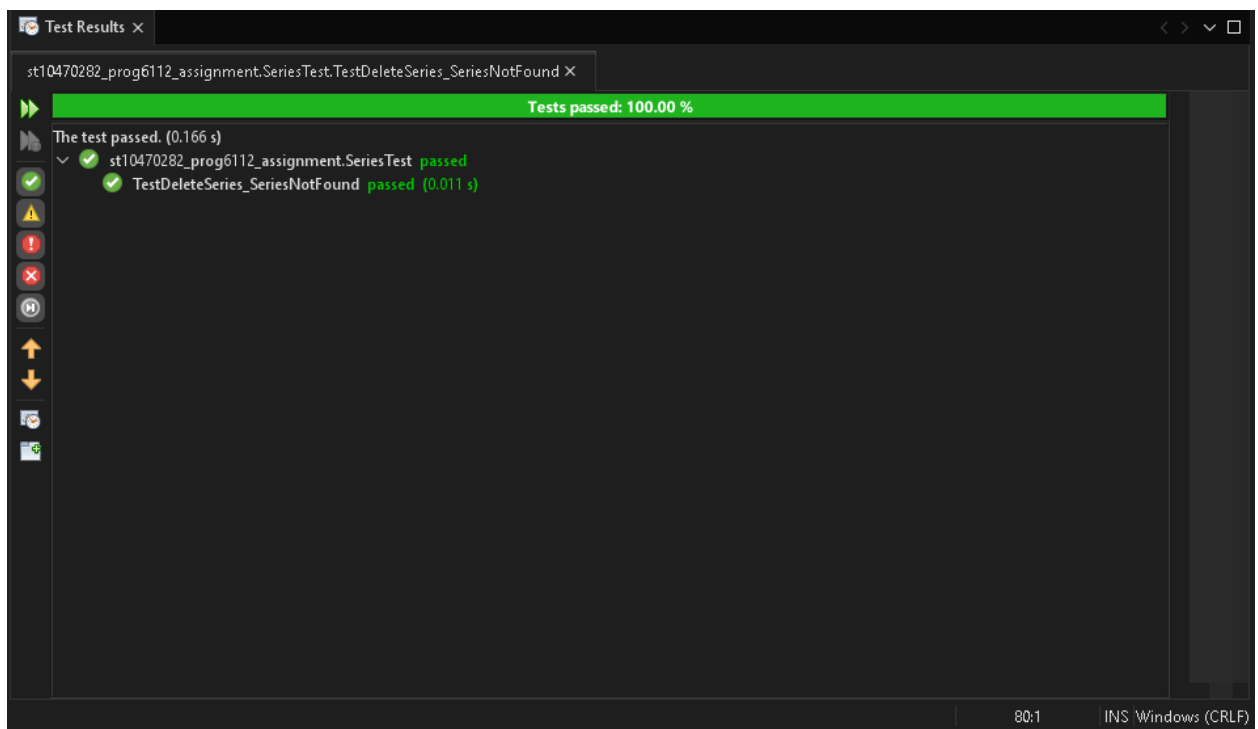
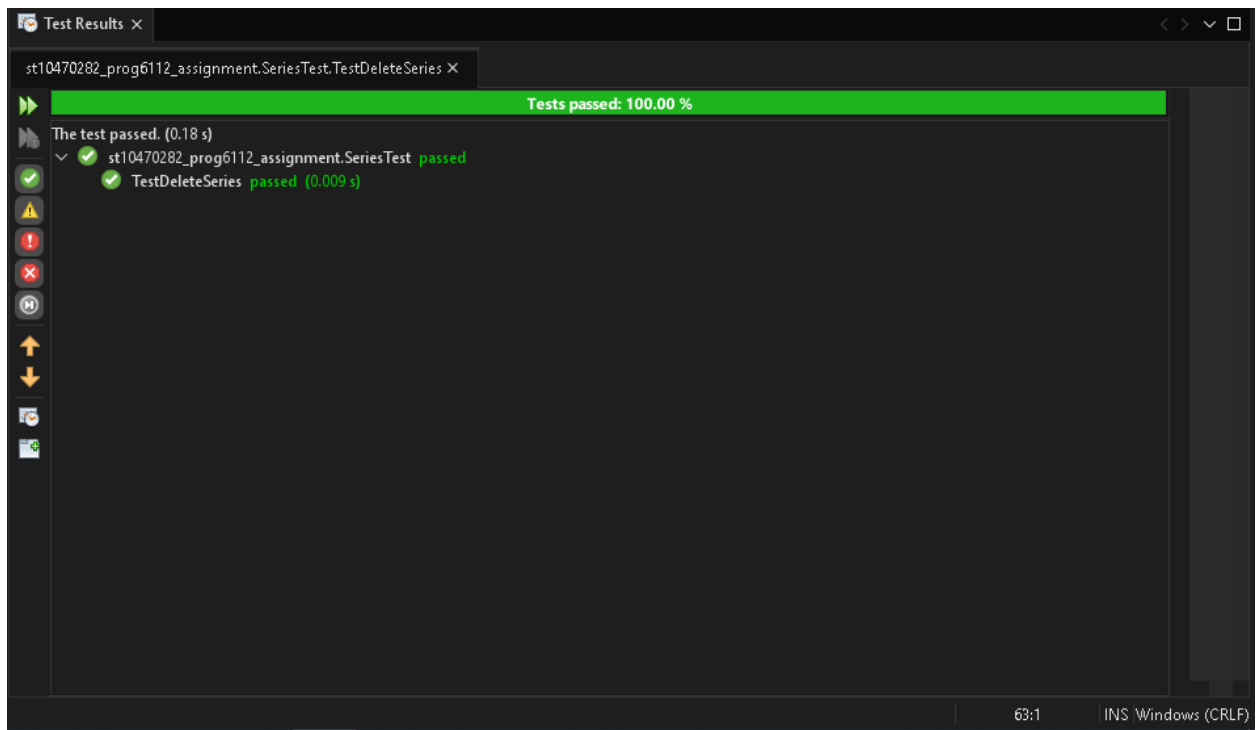
```

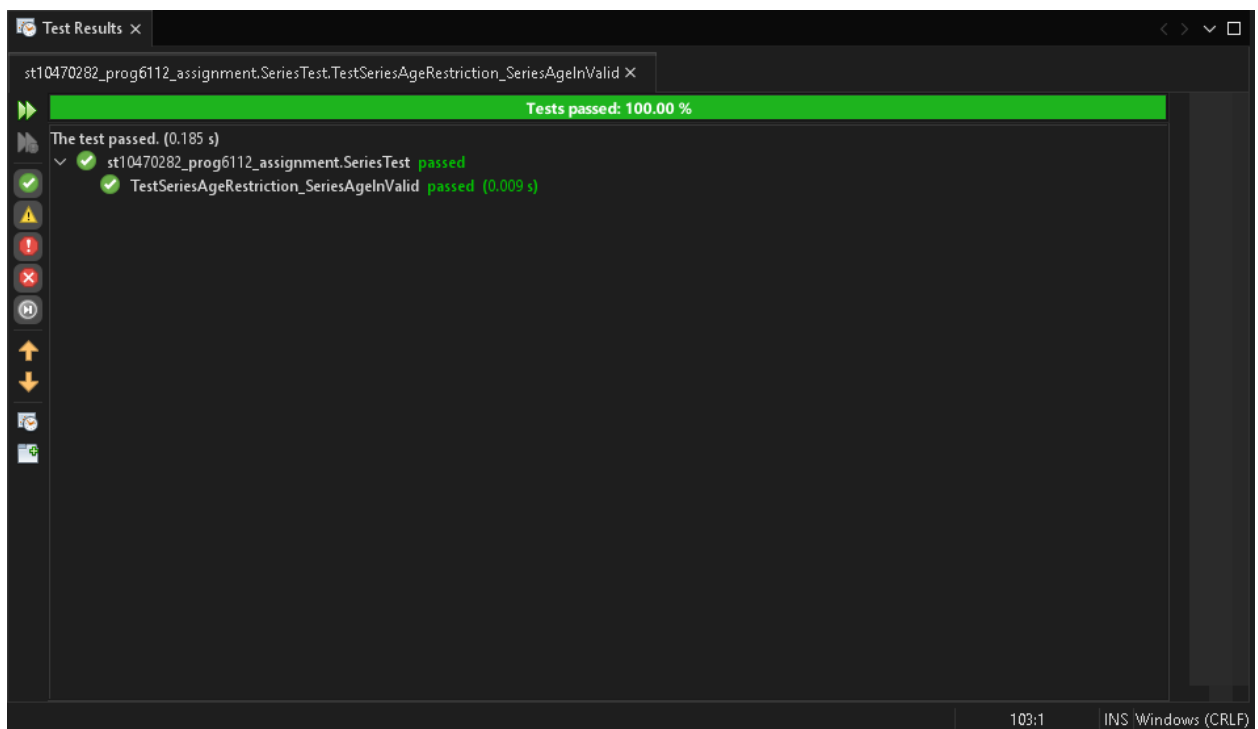
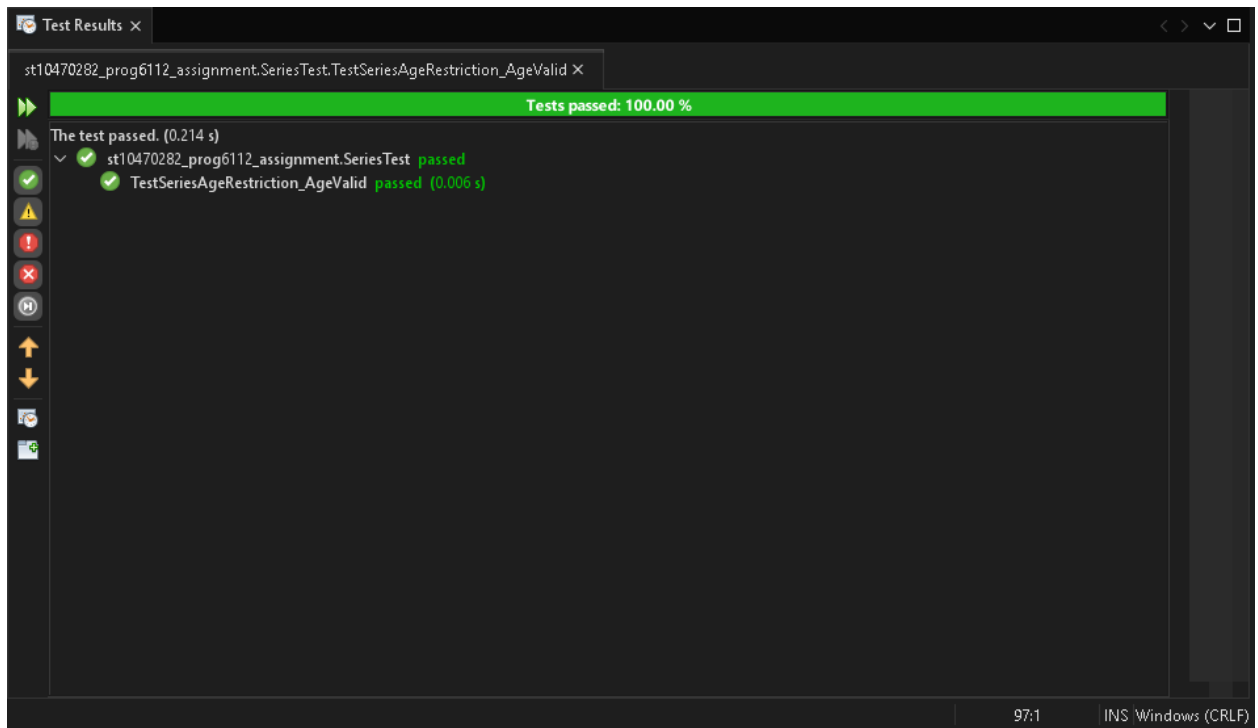
1:1 INS Windows (CRLF)

Test Results of the Unit Tests for Question 1:









Question 2 Code:

```
Source History
1
2 package prog6112.q2.st10470282.assignment;
3 import java.util.Scanner;
4 import java.util.ArrayList;
5 import java.util.InputMismatchException;
6 public class PROG6112Q2ST10470282Assignment {
7
8
9     public static void main(String[] args) {
10         //Initialized a Scanner with object called Scan
11         Scanner Scan = new Scanner(System.in);
12         //Created an arraylist to store the values with BankAccount objects and the variable name is account
13         ArrayList<BankAccount> Account = new ArrayList<>();
14
15         //Boolean with variable APP created. Set to true
16         //Runs the menu while true
17         boolean APP = true;
18         /*
19         GeeksforGeeks
20         While loops
21         Answer: Line 24
22         URL: https://www.w3schools.com/java/java\_while\_loop.asp
23         */
24         while(APP){
25             //Calling the method ShowMenu
26             ShowMenu();
27             //saves the users choice in order to run the specific command
28             int Choice = -1;
29             /*
30             GeeksforGeeks
31             Java Exception
```

```
Source History
32         Answer: Lines 35-43
33         URL: https://www.w3schools.com/java/java\_try\_catch.asp
34         */
35         try {
36             Choice = Scan.nextInt();
37
38         }
39         catch (InputMismatchException e) {
40             System.out.println("Please enter a valid integer option");
41             Scan.nextLine();
42             continue;
43         }
44         Scan.nextLine();
45         //switch statement created with argument Choice passed through
46         /*
47         ChatGPT
48         Question: how to create a switch statement with cases
49         Answer: Lines 52-57
50         URL: https://chatgpt.com/c/68b9fa87-4abc-8330-97fb-9ccalle20c2d
51         */
52         switch(Choice){
53             //Different cases created based on the menu
54             case 1:
55                 //Calling method CreateAccount with argument Scan and Account
56                 CreateAccount(Scan, Account);
57                 break;
58
59             case 2:
60                 //Calling method DepositMoney with argument Scan and Account
61                 DepositMoney(Scan, Account);
62                 break;
```

```

64         case 3:
65             //Calling method WithdrawMoney with argument Scan and Account
66             WithdrawMoney(Scan, Account);
67             break;
68
69         case 4:
70             //Calling method PrintReport with argument Scan and Account
71             PrintReport(Scan, Account);
72             break;
73
74         case 5:
75             SortAccount(Scan, Account);
76             break;
77
78         case 6:
79             //Calling method ExitApplication with argument Scan and Account
80             ExitApplication(Scan);
81             APP = false;
82             break;
83         //if the user does not select a number between 1 and 5 it will display and error
84         //The menu will then loop again
85         default:
86             System.out.println("Option is invalid!");
87             System.out.println(" ");
88     }
89 }
90 }
91 }
92 //created a method called ShowMenu
93 //It prints out the menu of the banking system
94 public static void ShowMenu(){

```

```

95     System.out.println("Welcome to the Banking System!!");
96     System.out.println("1. Create Account");
97     System.out.println("2. Deposit Money");
98     System.out.println("3. Withdraw Money");
99     System.out.println("4. Show my report");
100    System.out.println("5. Sort Accounts by balance in Ascending order");
101    System.out.println("6. Exit");
102    System.out.print("Choose and option (1-6): ");
103
104 }
105 //created a method called CreateAccount
106 //Used to create the account of the user
107 public static void CreateAccount(Scanner Scan , ArrayList<BankAccount> Account){
108     String AccountHolder;
109     /*
110     GeeksforGeeks
111     While loops
112     Answer: Line 115
113     URL: https://www.w3schools.com/java/java\_while\_loop.asp
114     */
115     while(true){
116         System.out.print("Enter account holder name: ");
117         AccountHolder = Scan.nextLine();
118         if(AccountHolder.matches("[a-zA-Z ]+")){
119             break;
120         }
121         else{
122             System.out.println("Account holder name must only contain letters!");
123             System.out.println(" ");
124         }
125     }

```

```
Source History
127     int AccountNumber;
128     /*
129     GeeksforGeeks
130     While loops
131     Answer: Line 134
132     URL: https://www.w3schools.com/java/java\_while\_loop.asp
133     */
134     while(true){
135         /*
136         GeeksforGeeks
137         Java Exception
138         Answer: Line 141
139         URL: https://www.w3schools.com/java/java\_try\_catch.asp
140         */
141         try{
142             System.out.print("Enter account number: ");
143             AccountNumber = Scan.nextInt();
144             break;
145         }
146         catch(InputMismatchException e){
147             System.out.println("Account number must only contain integers.");
148             System.out.println(" ");
149             Scan.nextLine();
150         }
151     }
152     int Balance;
153     /*
154     GeeksforGeeks
155     While loops
156     Answer: Line 159
157     URL: https://www.w3schools.com/java/java\_while\_loop.asp
```

```
Source History
158     /*
159     while(true){
160         /*
161         GeeksforGeeks
162         Java Exception
163         Answer: Lines 166
164         URL: https://www.w3schools.com/java/java\_try\_catch.asp
165         */
166         try {
167             System.out.print("Enter opening balance (must be an integer): ");
168             Balance = Scan.nextInt();
169             break;
170         }
171         catch(InputMismatchException e){
172             System.out.println("Opening balance must only contain integers.");
173             System.out.println(" ");
174             Scan.nextLine();
175         }
176     }
177     Scan.nextLine();
178     int AccType;
179     /*
180     GeeksforGeeks
181     While loops
182     Answer: Line 185
183     URL: https://www.w3schools.com/java/java\_while\_loop.asp
184     */
185     while(true){
186         /*
187         GeeksforGeeks
188         Java Exception
```

```
Source History
189 Answer: Lines 192
190 URL: https://www.w3schools.com/java/java\_try\_catch.asp
191 */
192 try{
193     System.out.print("Choose an account type (1. Savings, 2. Check): ");
194     AccType = Scan.nextInt();
195     if(AccType == 1 || AccType == 2){
196         break;
197     }
198     else{
199         System.out.println("Please enter 1 for Savings or 2 for Check.");
200         System.out.println(" ");
201     }
202 }
203 catch(InputMismatchException e){
204     System.out.println("Invalid input. Please enter 1 for Savings or 2 for Check.");
205     System.out.println(" ");
206     Scan.nextLine();
207 }
208 }
209 Scan.nextLine();
210 BankAccount NewAccount;
211 if(AccType == 1){
212     NewAccount = new SavingsAccount(AccountHolder, AccountNumber, Balance, 0.05);
213 }
214 else{
215     NewAccount = new CheckAccount(AccountHolder, AccountNumber, Balance, 2.50);
216 }
217
218 Account.add(NewAccount);
219 System.out.println("Account successfully created!");
```

```
Source History
220     System.out.println(" ");
221 }
222
223
224 //method DepositMoney created to deposit money into the account
225 public static void DepositMoney(Scanner Scan, ArrayList<BankAccount> Account){
226     System.out.print("Enter Account Holder name: ");
227     String AccName = Scan.nextLine();
228     System.out.print("Enter account number: ");
229     int AccNum = Scan.nextInt();
230     Scan.nextLine();
231     BankAccount FoundAcc = null;
232     /*
233     GeeksforGeeks
234     //Java for loops
235     //Answer: line 238
236     URL: https://www.geeksforgeeks.org/java/loops-in-java/
237     */
238     for(BankAccount Acc : Account) {
239         if (Acc.getAccountNumber() == AccNum && Acc.getAccountHolder().equals(AccName)) {
240             FoundAcc = Acc;
241             break;
242         }
243     }
244     if (FoundAcc == null){
245         System.out.println("Account not found!");
246         System.out.println(" -----");
247         return;
248     }
249
250     System.out.print("Enter amount to deposit: ");
```

```
Source History
251 double DepositAmm = Scan.nextDouble();
252 Scan.nextLine();
253 //if the account number entered = to an account number in the system it will allow you to deposit money
254 FoundAcc.Deposit(DepositAmm);
255 System.out.println("Deposit was successful!");
256 System.out.println("-----New Balance-----: R" + FoundAcc.getBalance());
257 }
258
259
260 //method withdrawmoney created to withdraw money from an account
261 public static void WithdrawMoney(Scanner Scan, ArrayList<BankAccount> Account){
262     System.out.print("Enter Account Holder name: ");
263     String AccName = Scan.nextLine();
264     System.out.print("Enter the Account Number: ");
265     int AccNum = Scan.nextInt();
266     Scan.nextLine();
267     BankAccount foundAcc = null;
268     /*
269     GeeksforGeeks
270     //Java for loops
271     //Answer: line 274
272     URL: https://www.geeksforgeeks.org/java/loops-in-java/
273     */
274     for(BankAccount Acc : Account){
275         if(Acc.getAccountNumber() == AccNum && Acc.getAccountHolder().equals(AccName)) {
276             foundAcc = Acc;
277             break;
278         }
279     }
280
281     if(foundAcc == null){
```

```
Source History
282         System.out.println("Account not found!");
283         System.out.println(" -----");
284         return;
285     }
286
287     System.out.print("Enter amount to withdraw: ");
288     double WithdrawAmm = Scan.nextDouble();
289     Scan.nextLine();
290     //account holder name and account must be equal to one in the system to withdraw money
291     if(foundAcc.getBalance() >= WithdrawAmm){
292         foundAcc.Withdraw(WithdrawAmm);
293         System.out.println("Withdrawal was successful!");
294         System.out.println("-----New Balance-----: R" + foundAcc.getBalance());
295     }
296     else{
297         System.out.println("Withdrawal unsuccessful, Insufficient funds!");
298         System.out.println(" ");
299     }
300 }
301
302 //method PrintReport created to print the final report of the banking system
303 public static void PrintReport(Scanner Scan, ArrayList<BankAccount> Account){
304     System.out.println("-----Account Report-----");
305     System.out.println("-----");
306     if(Account.isEmpty()){
307         System.out.println("No Report available");
308     }
309     else{
310         int i = 1;
311         /*
312         GeeksforGeeks
```



```
Source History
313      Java for loops
314      Answer: line 317
315      URL: https://www.geeksforgeeks.org/java/loops-in-java/
316      */
317      for(BankAccount Acc : Account){
318          System.out.println("Account " + i + ": " + Acc.Display());
319          i++;
320      }
321  }
322  }
323  }
324  //method SortAccount created to sort the account in ascending order
325  public static void SortAccount(Scanner Scan, ArrayList<BankAccount> Account){
326      /*
327      GeeksforGeeks
328      Bubble Sort Algorithm
329      Answer: Lines 332-342
330      URL: https://www.geeksforgeeks.org/dsa/bubble-sort-algorithm/
331      */
332      for(int i =0; i<Account.size()-1; i++){
333          for(int j=0; j<Account.size() -i-1; j++){
334              if(Account.get(j).getBalance() > Account.get(j + 1).getBalance()){
335                  BankAccount temp = Account.get(j);
336                  Account.set(j, Account.get(j + 1));
337                  Account.set(j + 1, temp);
338              }
339          }
340      }
341      System.out.println("Accounts are sorted in Ascending order");
342      PrintReport(Scan, Account);
343  }
```

```
Source History
345  public static void ExitApplication(Scanner Scan){
346      System.out.println("-----");
347      System.out.println("Bye! See you soon!");
348  }
349  }
350  }
351  //super class BankAccount created
352  class BankAccount{
353      //private variables created that belongs to each object of the class
354      private String AccountHolder;
355      private int AccountNumber;
356      protected double Balance;
357      /*
358      W3Schools
359      Constructors
360      Answer: Lines 363-368
361      URL: https://www.w3schools.com/java/java\_constructors.asp
362      */
363      public BankAccount(String AccountHolder, int AccountNumber, double Balance){
364          //Takes the values passed into the constructor and stores it in the object BankAccount
365          this.AccountHolder = AccountHolder;
366          this.AccountNumber = AccountNumber;
367          this.Balance = Balance;
368      }
369      // Creating getters to access the account details later
370      /*
371      GeeksforGeeks
372      Getters and Setters
373      Answer: 376-385
374      URL: https://www.geeksforgeeks.org/java/getter-and-setter-in-java/
375      */
```

```
Source History
376 public String getAccountHolder() {
377     return AccountHolder;
378 }
379 public int getAccountNumber(){
380     return AccountNumber;
381 }
382
383 public double getBalance(){
384     return Balance;
385 }
386
387 //Adds money to the balance if it meets the requirements
388 public void Deposit(double Amount){
389     if(Amount > 0){
390         Balance += Amount;
391     }
392 }
393 //Subtracts money from the balance if it meets the requirements
394 public void Withdraw(double Amount){
395     if(Amount > 0 && Amount <= Balance){
396         Balance -= Amount;
397     }
398 }
399 //method created to print out the details of the account
400 public String Display(){
401     return "Account Holder: " + AccountHolder + "\n | Account Number: " + AccountNumber
402         + "\n | Balance: " + Balance + "\n ";
403 }
404
405 }
406 }
```

```
Source History
408 //created a sub class savingsaccount that extends the class BankAccount
409 class SavingsAccount extends BankAccount{
410     private double InterestRate;
411
412     public SavingsAccount(String AccountHolder, int AccountNumber, double Balance, double InterestRate){
413         super(AccountHolder, AccountNumber, Balance);
414         this.InterestRate = InterestRate;
415     }
416     public void ApplyInterest(){
417         Deposit(getBalance() * InterestRate);
418     }
419     //Override the Display method from super class
420     /*
421     GeeksforGeeks
422     Overriding in Java
423     Answer: Lines 426-428
424     URL: https://www.geeksforgeeks.org/java/overriding-in-java/
425     */
426     @Override
427     public String Display(){
428         return super.Display() + "| Type: Savings Account(InterestRate: " + InterestRate * 100 + "%";
429     }
430 }
431
432
433 //created a sub class called CheckAccount that extends the class BankAccount
434 class CheckAccount extends BankAccount{
435     private double Fee;
436
437     public CheckAccount(String AccountHolder, int AccountNumber, double Balance, double fee){
438         super(AccountHolder, AccountNumber, Balance);
```

```
439     this.Fee = fee;
440 }
441 //method created to add the fee with the withdrawl and check if the available balance is enough
442 public void Withdraw(double Amount){
443     if(Amount + Fee <= getBalance()){
444         super.Withdraw(Amount + Fee);
445     }
446     else{
447         System.out.println("Insufficient funds to cover withdrawl and Fee!");
448     }
449 }
450 //Override the display method from super class BankAccount
451 /*
452 GeeksforGeeks
453 Overriding in Java
454 Answer: Lines 457-459
455 URL: https://www.geeksforgeeks.org/java/overriding-in-java/
456 */
457 @Override
458 public String Display(){
459     return super.Display() + "| Type: Check Account Fee: R" + Fee;
460 }
461
462
463 }
464 }
```

Output for Question 2:

```
Output - PROG6112 Q2 ST10470282 Assignment (run) x
run:
Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 1
Enter account holder name: Keenan Mouton
Enter account number: 10345863
Enter opening balance (must be an integer): 100000
Choose an account type (1. Savings, 2. Check): 1
Account successfully created!

Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 1
Enter account holder name: KateLin Geldenhuys
Enter account number: 1826365
Enter opening balance (must be an integer): 75000
Choose an account type (1. Savings, 2. Check): 2
Account successfully created!
```

454:23 INS

```
Output - PROG6112 Q2 ST10470282 Assignment (run) x
Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 1
Enter account holder name: Charlotte Stoltz
Enter account number: 281365
Enter opening balance (must be an integer): 1000000
Choose an account type (1. Savings, 2. Check): 1
Account successfully created!

Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 2
Enter Account Holder name: Keenan Mouton
Enter account number: 10345863
Enter amount to deposit: 23000
Deposit was successful!
-----New Balance-----: R123000.0
Welcome to the Banking System!!
1. Create Account
```

454:23 INS

```
Output - PROG6112 Q2 ST10470282 Assignment (run) x
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 3
Enter Account Holder name: Charlotte Stoltz
Enter the Account Number: 281365
Enter amount to withdraw: 350000
Withdrawal was successful!
----New Balance----: R650000.0
Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 4
-----Account Report-----
-----
Account 1: Account Holder: Keenan Mouton
| Account Number: 10345863
| Balance: 123000.0
| Type: Savings Account(InterestRate: 5.0%
Account 2: Account Holder: Katerlin Geldenhuys
| Account Number: 1826365
| Balance: 75000.0
| Type: Check Account Fee: R2.5
Account 3: Account Holder: Charlotte Stoltz
| Account Number: 281365
| Balance: 650000.0
| Type: Savings Account(InterestRate: 5.0%
Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 5
Accounts are sorted in Ascending order
-----Account Report-----
-----
Account 1: Account Holder: Katerlin Geldenhuys
| Account Number: 1826365
| Balance: 75000.0
| Type: Check Account Fee: R2.5
Account 2: Account Holder: Keenan Mouton
| Account Number: 10345863
| Balance: 123000.0
| Type: Savings Account(InterestRate: 5.0%
Account 3: Account Holder: Charlotte Stoltz
| Account Number: 281365
| Balance: 650000.0
| Type: Savings Account(InterestRate: 5.0%
Welcome to the Banking System!!
```

```
Output - PROG6112 Q2 STI0470282 Assignment (run) x
6. Exit
Choose and option (1-6): 5
Accounts are sorted in Ascending order
-----Account Report-----
Account 1: Account Holder: Katelin Geldenhuys
| Account Number: 1826365
| Balance: 75000.0
| Type: Check Account Fee: R2.5
Account 2: Account Holder: Keenan Mouton
| Account Number: 10345863
| Balance: 123000.0
| Type: Savings Account(InterestRate: 5.0%
Account 3: Account Holder: Charlotte Stoltz
| Account Number: 281365
| Balance: 650000.0
| Type: Savings Account(InterestRate: 5.0%
Welcome to the Banking System!!
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Show my report
5. Sort Accounts by balance in Ascending order
6. Exit
Choose and option (1-6): 6
-----
Bye! See you soon
BUILD SUCCESSFUL (total time: 3 minutes 53 seconds)
```

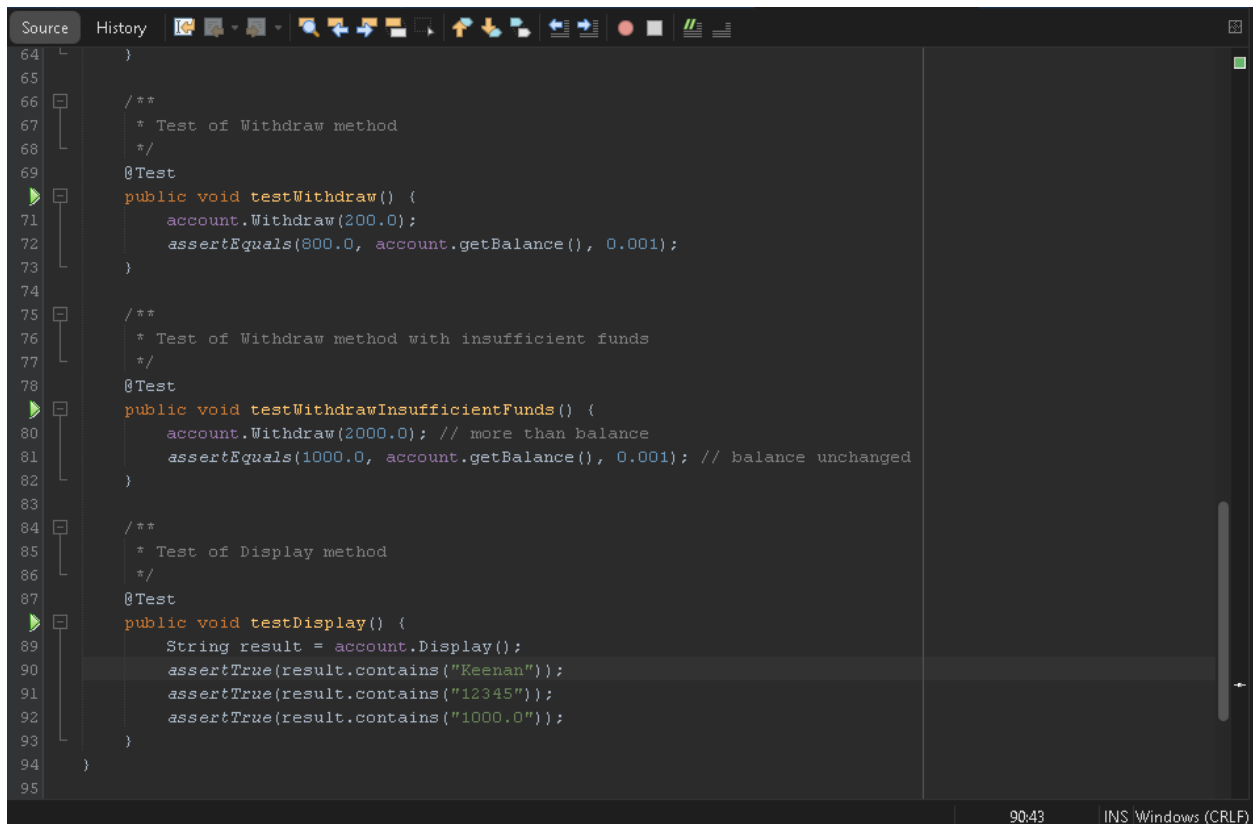
Unit Test for Question 2 (Class Bank Account):

```
Source History
1 package prog6112.q2.st10470282.assignment;
2
3 import org.junit.After;
4 import org.junit.AfterClass;
5 import org.junit.Before;
6 import org.junit.BeforeClass;
7 import org.junit.Test;
8 import static org.junit.Assert.*;
9
10 public class BankAccountTest {
11
12     private BankAccount account;
13
14     @BeforeClass
15     public static void setUpClass() {
16     }
17
18     @AfterClass
19     public static void tearDownClass() {
20     }
21
22     @Before
23     public void setUp() {
24         // Create a test BankAccount before each test
25         account = new BankAccount("Keenan", 12345, 1000.0);
26     }
27
28     @After
29     public void tearDown() {
30         account = null;
31     }
32 }
```

90:43 INS Windows (CRLF)

```
Source History
33 /**
34  * Test of getAccountHolder method
35  */
36 @Test
37 public void testGetAccountHolder() {
38     assertEquals("Keenan", account.getAccountHolder());
39 }
40
41 /**
42  * Test of getAccountNumber method
43  */
44 @Test
45 public void testGetAccountNumber() {
46     assertEquals(12345, account.getAccountNumber());
47 }
48
49 /**
50  * Test of getBalance method
51  */
52 @Test
53 public void testGetBalance() {
54     assertEquals(1000.0, account.getBalance(), 0.001);
55 }
56
57 /**
58  * Test of Deposit method
59  */
60 @Test
61 public void testDeposit() {
62     account.Deposit(500.0);
63     assertEquals(1500.0, account.getBalance(), 0.001);
64 }
```

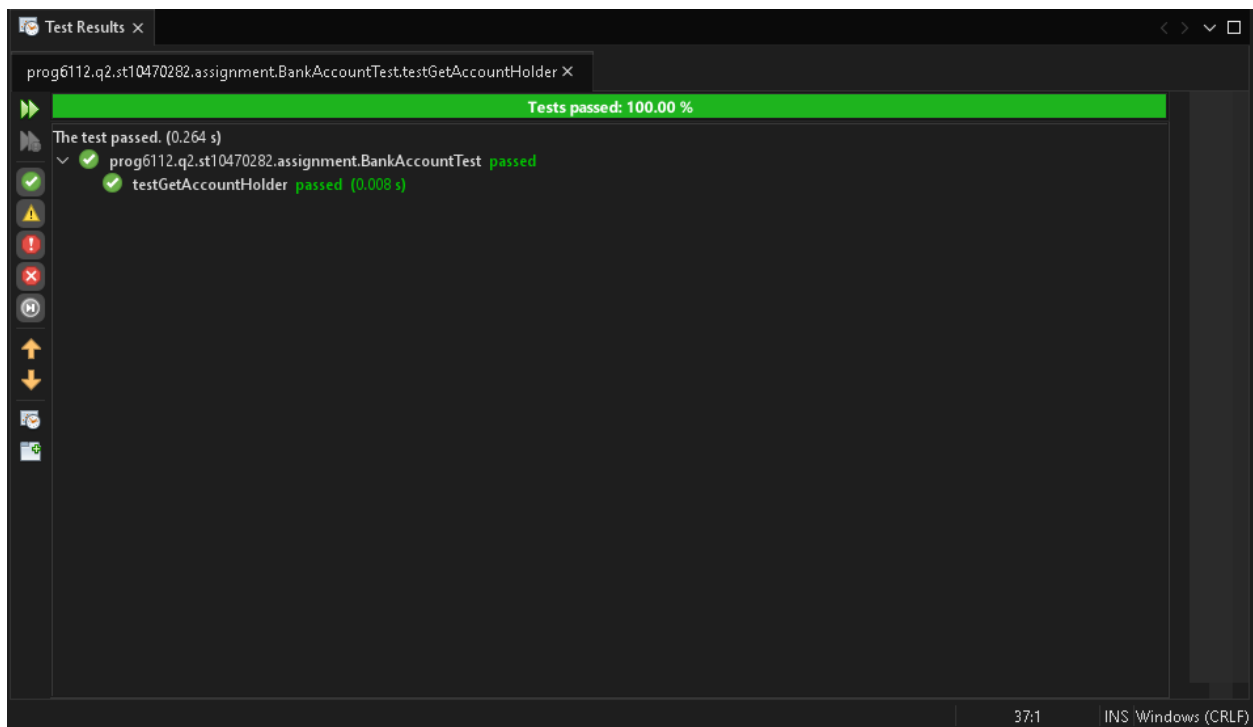
90:43 INS



The screenshot shows an IDE window with a dark theme. The 'Source' tab is active, displaying Java code for unit tests. The code is organized into three test methods, each preceded by a Javadoc comment and an @Test annotation. The first method, testWithdraw(), tests the Withdraw method with a balance of 200.0 and an expected balance of 800.0. The second method, testWithdrawInsufficientFunds(), tests the Withdraw method with a balance of 2000.0 and an expected balance of 1000.0. The third method, testDisplay(), tests the Display method, asserting that the result contains 'Keenan', '12345', and '1000.0'. The code is line-numbered from 64 to 95. The status bar at the bottom right shows '90:43' and 'INS Windows (CRLF)'.

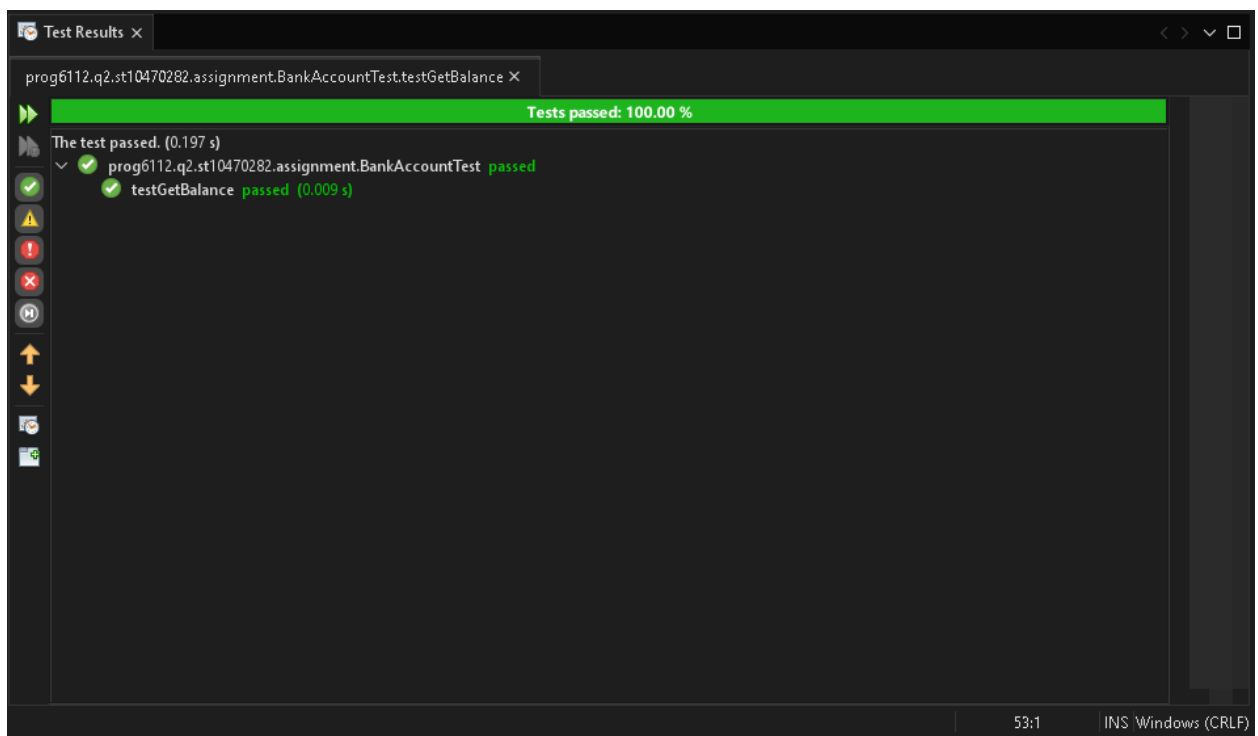
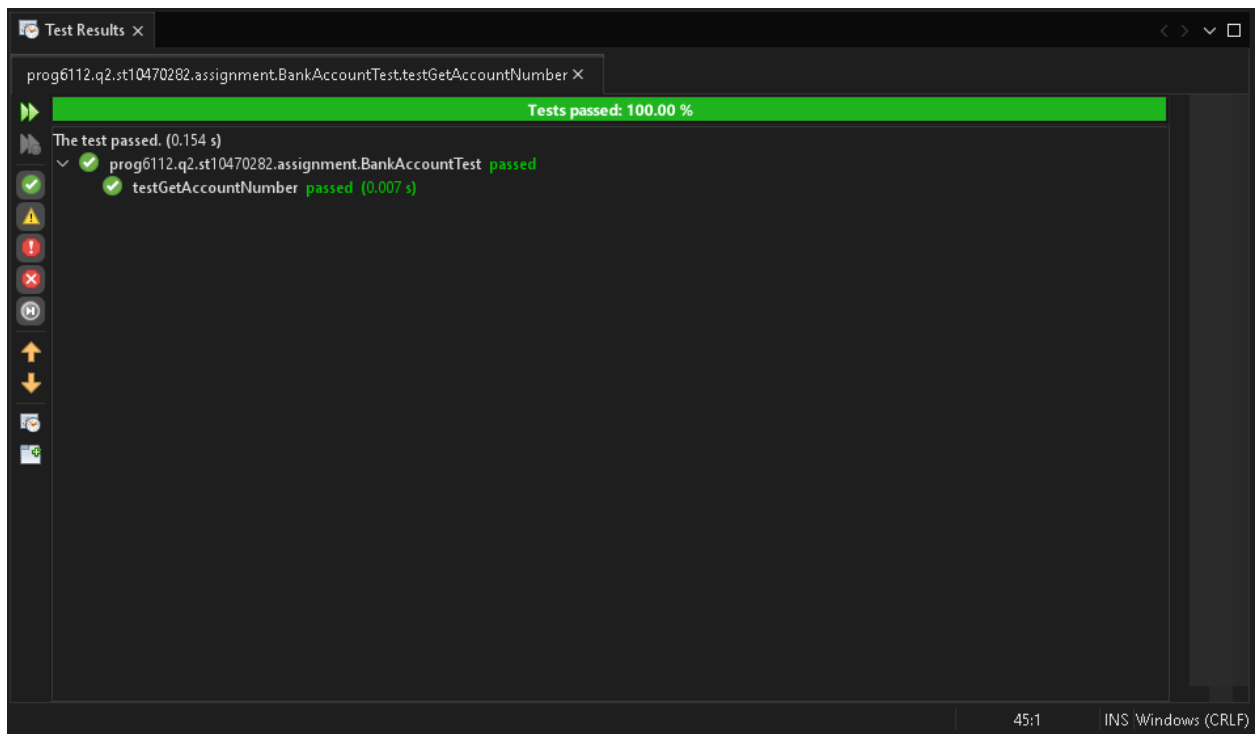
```
64 }
65
66 /**
67  * Test of Withdraw method
68  */
69 @Test
70 public void testWithdraw() {
71     account.Withdraw(200.0);
72     assertEquals(800.0, account.getBalance(), 0.001);
73 }
74
75 /**
76  * Test of Withdraw method with insufficient funds
77  */
78 @Test
79 public void testWithdrawInsufficientFunds() {
80     account.Withdraw(2000.0); // more than balance
81     assertEquals(1000.0, account.getBalance(), 0.001); // balance unchanged
82 }
83
84 /**
85  * Test of Display method
86  */
87 @Test
88 public void testDisplay() {
89     String result = account.Display();
90     assertTrue(result.contains("Keenan"));
91     assertTrue(result.contains("12345"));
92     assertTrue(result.contains("1000.0"));
93 }
94 }
95
```

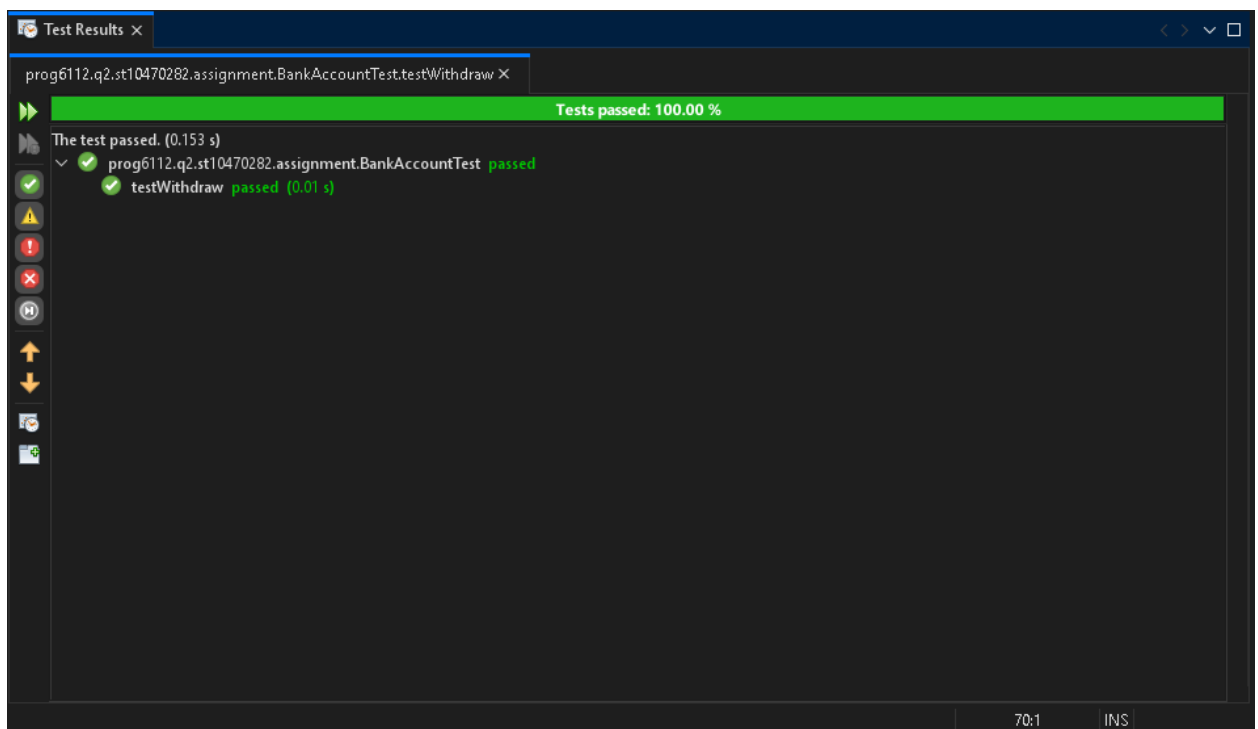
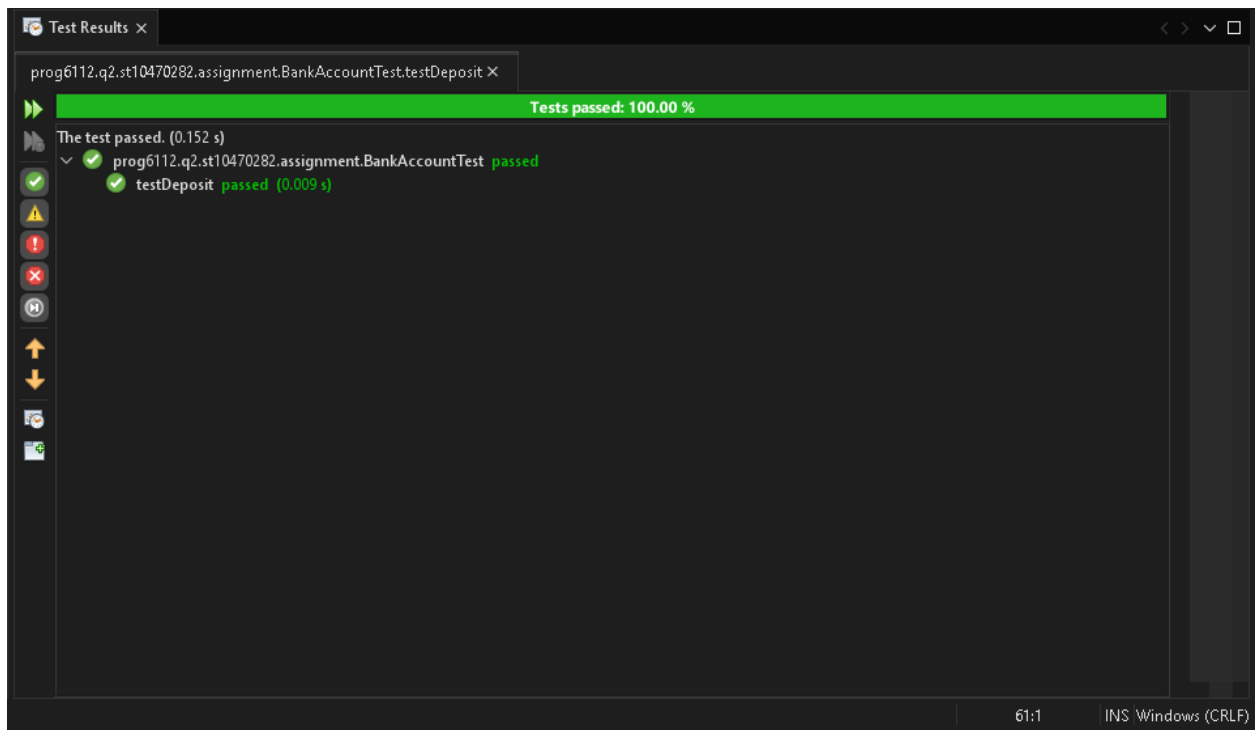
Unit Test Results for Bank Account:

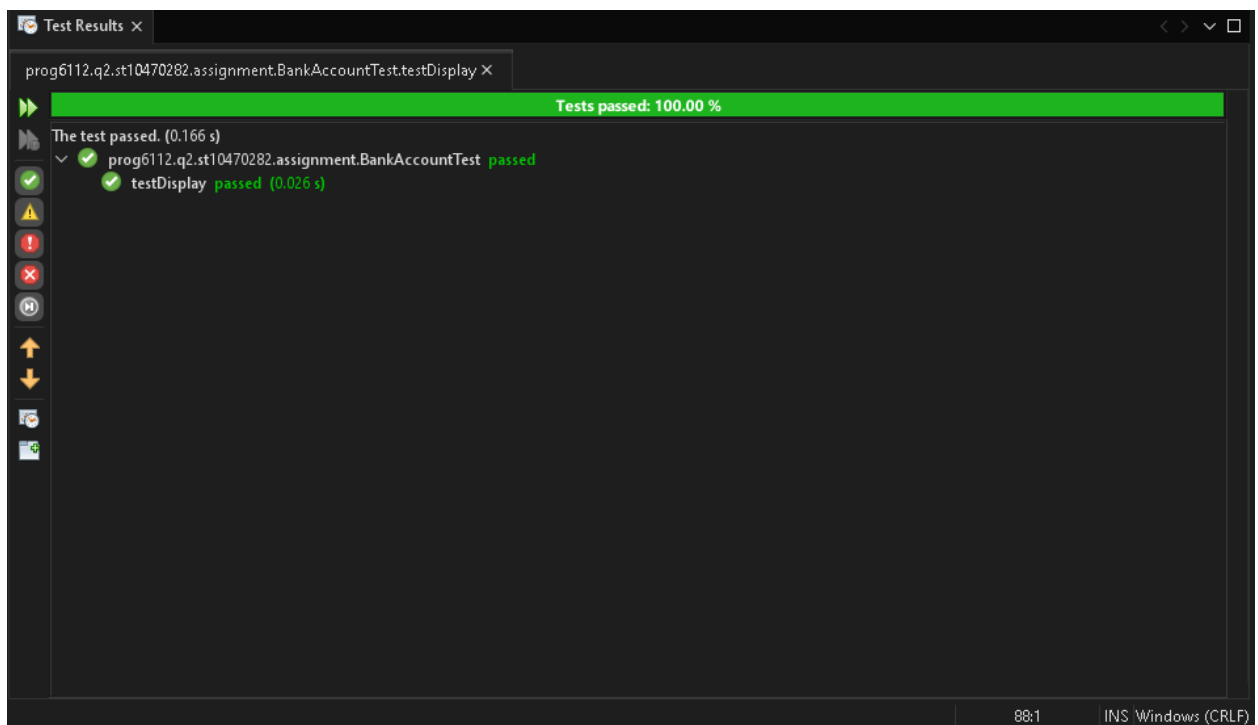
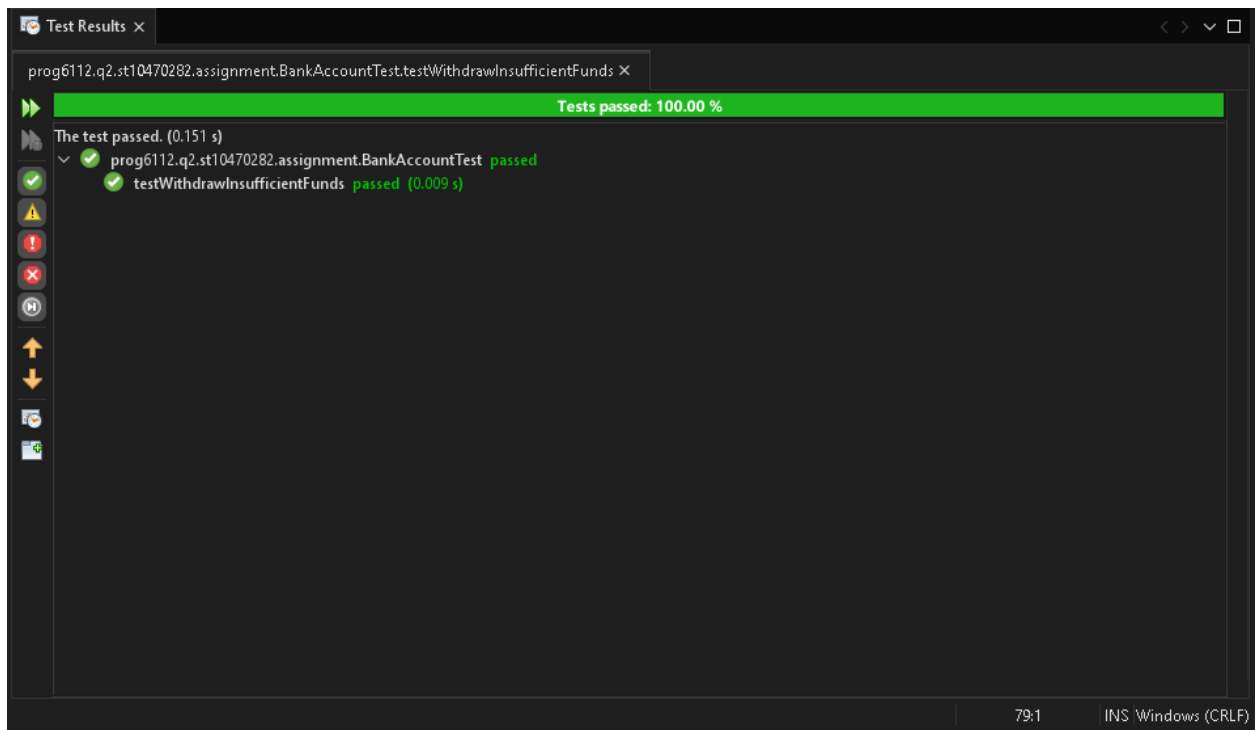


The screenshot shows a 'Test Results' window in an IDE. The window title is 'Test Results'. The main area displays the test results for the class 'prog6112.q2.st10470282.assignment.BankAccountTest'. A green bar at the top indicates 'Tests passed: 100.00 %'. Below this, the text 'The test passed. (0.264 s)' is shown. The test results are listed in a tree view, showing that the test 'testGetAccountHolder' passed (0.008 s). The status bar at the bottom right shows '37:1' and 'INS Windows (CRLF)'.

```
Test Results
prog6112.q2.st10470282.assignment.BankAccountTest.testGetAccountHolder X
Tests passed: 100.00 %
The test passed. (0.264 s)
prog6112.q2.st10470282.assignment.BankAccountTest passed
testGetAccountHolder passed (0.008 s)
```



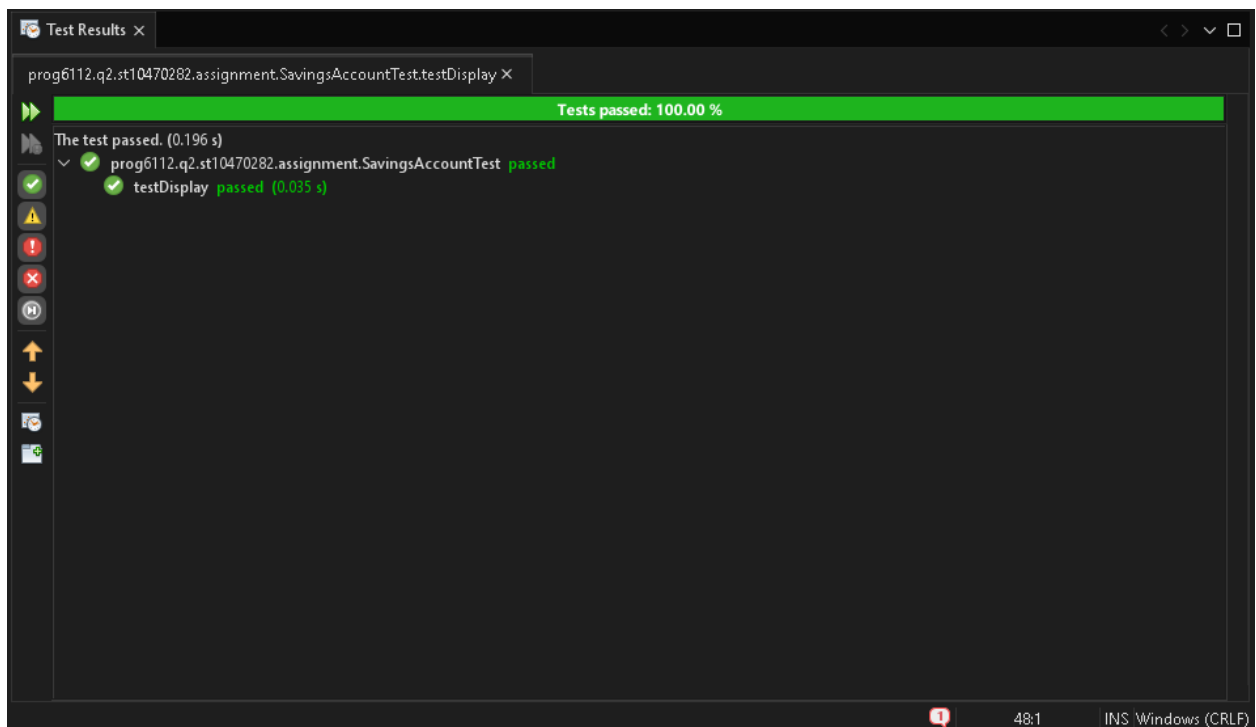
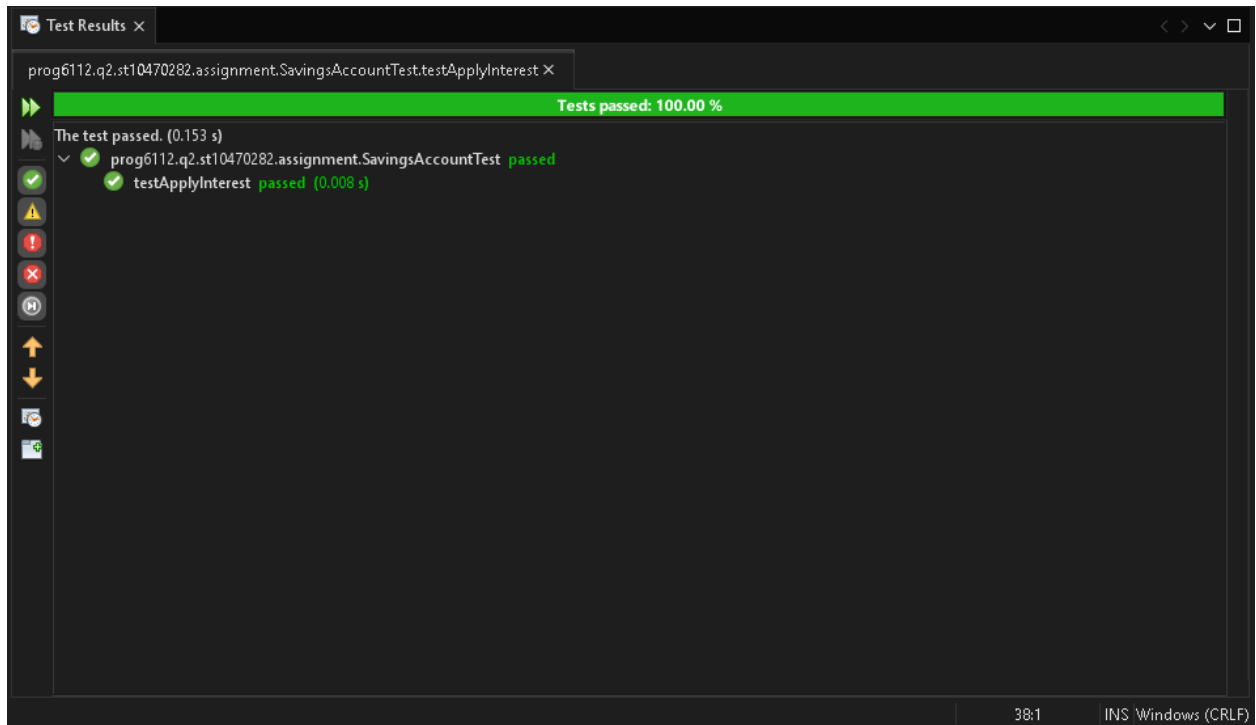


Unit Test for class Savings Account:

```
Source History
1 package prog6112.q2.st10470282.assignment;
2
3 import org.junit.After;
4 import org.junit.AfterClass;
5 import org.junit.Before;
6 import org.junit.BeforeClass;
7 import org.junit.Test;
8 import static org.junit.Assert.*;
9
10 public class SavingsAccountTest {
11
12     private SavingsAccount savingsAccount;
13
14     @BeforeClass
15     public static void setUpClass() {
16     }
17
18     @AfterClass
19     public static void tearDownClass() {
20     }
21
22     @Before
23     public void setUp() {
24         // Create a test SavingsAccount before each test
25         // Balance = 1000, InterestRate = 5% (0.05)
26         savingsAccount = new SavingsAccount("Keenan", 67890, 1000.0, 0.05);
27     }
28
29     @After
30     public void tearDown() {
31         savingsAccount = null;
32     }
33 }
```

```
Source History
34 /**
35  * Test of ApplyInterest method
36  */
37 @Test
38 public void testApplyInterest() {
39     savingsAccount.ApplyInterest();
40     // 5% of 1000 = 50 added → new balance = 1050
41     assertEquals(1050.0, savingsAccount.getBalance(), 0.001);
42 }
43
44 /**
45  * Test of Display method
46  */
47 @Test
48 public void testDisplay() {
49     String result = savingsAccount.Display();
50     assertTrue(result.contains("Keenan"));
51     assertTrue(result.contains("67890"));
52     assertTrue(result.contains("1000.0"));
53     assertTrue(result.contains("Savings Account"));
54     assertTrue(result.contains("5.0%"));
55 }
56 }
```

Unit Test Results for class Savings Account:



Unit Test for class Check Account:

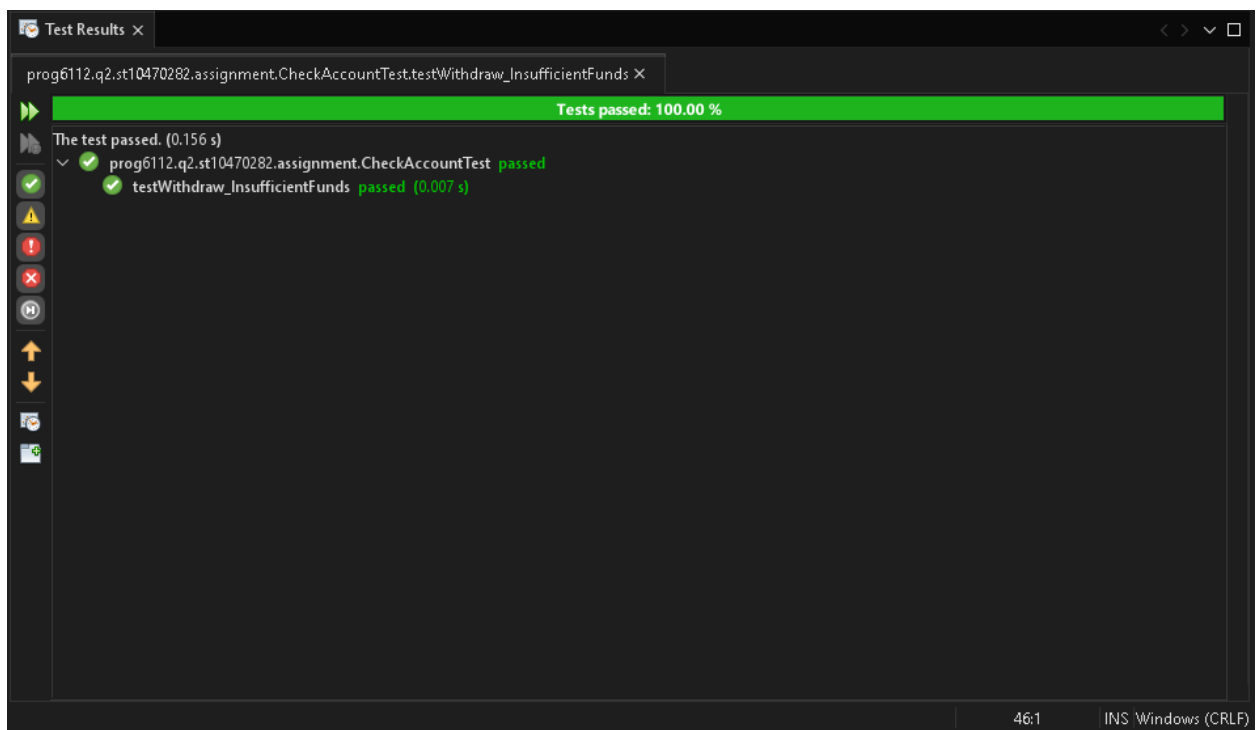
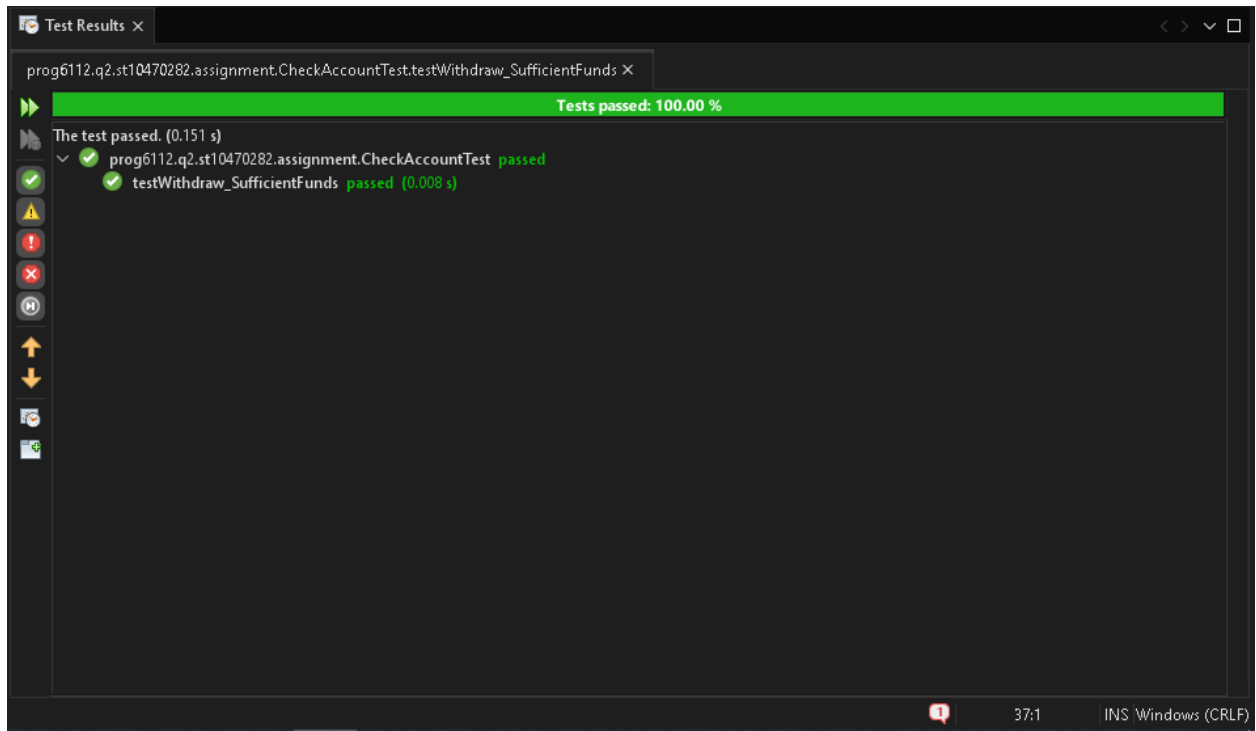
```
Source History
1 package prog6112.q2.st10470282.assignment;
2
3 import org.junit.After;
4 import org.junit.AfterClass;
5 import org.junit.Before;
6 import org.junit.BeforeClass;
7 import org.junit.Test;
8 import static org.junit.Assert.*;
9
10 public class CheckAccountTest {
11
12     private CheckAccount checkAccount;
13
14     @BeforeClass
15     public static void setUpClass() {
16     }
17
18     @AfterClass
19     public static void tearDownClass() {
20     }
21
22     @Before
23     public void setUp() {
24         // Create a test CheckAccount before each test
25         checkAccount = new CheckAccount("Keenan", 54321, 1000.0, 50.0); // R50 fee
26     }
27
28     @After
29     public void tearDown() {
30         checkAccount = null;
31     }
32 }
```

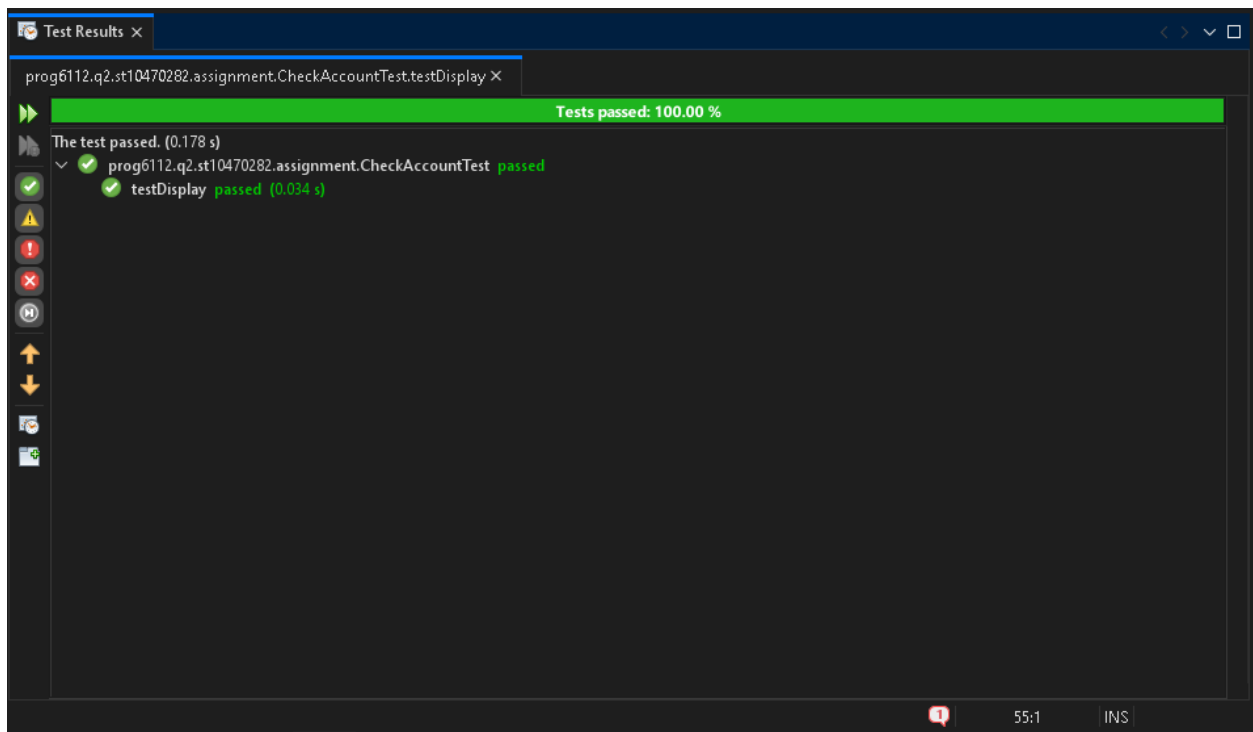
55:1 INS Windows (CRLF)

```
Source History
33 /**
34  * Test of Withdraw method with enough balance (includes fee)
35  */
36 @Test
37 public void testWithdraw_SufficientFunds() {
38     checkAccount.Withdraw(200.0); // should deduct 200 + 50 fee = 250
39     assertEquals(750.0, checkAccount.getBalance(), 0.001);
40 }
41
42 /**
43  * Test of Withdraw method with insufficient funds (amount + fee > balance)
44  */
45 @Test
46 public void testWithdraw_InsufficientFunds() {
47     checkAccount.Withdraw(2000.0); // too high, should fail
48     assertEquals(1000.0, checkAccount.getBalance(), 0.001); // balance unchanged
49 }
50
51 /**
52  * Test of Display method
53  */
54 @Test
55 public void testDisplay() {
56     String result = checkAccount.Display();
57     assertTrue(result.contains("Keenan"));
58     assertTrue(result.contains("54321"));
59     assertTrue(result.contains("1000.0"));
60     assertTrue(result.contains("Check Account"));
61     assertTrue(result.contains("50.0"));
62 }
63 }
```

55:1 INS Windows (CRLF)

Unit Test Results for class Check Account:





Github Repository link:

<https://github.com/VCCT-PROG6112-2025-G3/ST10470282-PROG6112-Assignment-Keenan.git>