



**Assignment: Portfolio of Evidence Part 2**

**Due date: 2025/04/26**

**Name: Blessing Jafari Mbalaka**

**Student Number: ST10466840**

**Assignment Instructions: Java Messaging Class**

## SECTION A-STUDENT DETAILS

**Student Number:** ST10466840

**Name(s):** Blessing Jafari

**Surname:** Mbalaka

**Assessment:** POE Part 2 [Message class]

---

### Links to Github and Youtube Video:

- **Private repo Github link also here:**  
<https://github.com/VCPTA/bca1-prog5121-part-1-submission-ST10466840/pull/new/feature/Login-message-class>
  - **Youtube Video of Unit Program code explanation/Walkthrough and Unit tests:**
  - **Unit Test video:** [https://youtu.be/3Cku\\_lrtTkA](https://youtu.be/3Cku_lrtTkA)
  - **Code Running Video:**  
<https://youtu.be/LBmqUpDA08g>
- 

## Section B-Full Source Code

```
package com.mycompany.messagingapp;
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;

public class Messagingapp_ {
    public static void main(String[] args) {
        // 1) perform login flow
        Login.runLogin();

        // 2) welcome
        JOptionPane.showMessageDialog(
            null,
            "Welcome to QuickChat.",
            "Welcome",
            JOptionPane.INFORMATION_MESSAGE
        );

        MessageStore.init();
        boolean running = true;

        while (running) {
            String[] options = {"Send Messages", "Show Recently Sent Messages", "Quit"};
            int choice = JOptionPane.showOptionDialog(
                null,
                "Choose an option:",
                "Main Menu",
                JOptionPane.DEFAULT_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                null,
                options,
                options[0]
            );
            switch (choice) {
```

```

        case 0: // Send Messages
            sendMessagesFlow();
            break;
        case 1: // Show recently sent
            JOptionPane.showMessageDialog(
                null,
                "Coming Soon.",
                "Feature In Development",
                JOptionPane.INFORMATION_MESSAGE
            );
            break;
        default: // Quit or closed
            running = false;
    }
}

// on exit: export JSON
MessageStore.exportJson();
System.exit(0);
}

private static void sendMessagesFlow() {
    String input = JOptionPane.showInputDialog(
        null,
        "How many messages will you send?",
        "Send Messages",
        JOptionPane.QUESTION_MESSAGE
    );
    int max;
    try {
        max = Integer.parseInt(input.trim());
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(null, "Invalid number.", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    for (int i = 1; i <= max; i++) {
        String title = "Message " + i + " of " + max;
        String recipient = JOptionPane.showInputDialog(null, "Recipient (+...):", title,
JOptionPane.QUESTION_MESSAGE);
        String content = JOptionPane.showInputDialog(null, "Message (≤250 chars):", title,
JOptionPane.QUESTION_MESSAGE);

        Message m = new Message(Message.generateID(), i, recipient, content);

        // validations
        if (!m.checkMessageID()) {
            JOptionPane.showMessageDialog(null, "Message ID invalid (max 10 characters).", "Validation Error",
JOptionPane.ERROR_MESSAGE);
            i--; continue;
        }
        if (!m.checkRecipientCell()) {
            JOptionPane.showMessageDialog(null, "Recipient invalid (max 10 chars, must start with +).",
"Validation Error", JOptionPane.ERROR_MESSAGE);
            i--; continue;
        }

        String[] sendOpts = {"Send", "Discard", "Store"};

```

```

int sendChoice = JOptionPane.showOptionDialog(
    null,
    "Select action for this message:",
    title,
    JOptionPane.DEFAULT_OPTION,
    JOptionPane.QUESTION_MESSAGE,
    null,
    sendOpts,
    sendOpts[0]
);
m.sendMessage(sendOpts[sendChoice]);

JOptionPane.showMessageDialog(null, m.printMessage(), title,
JOptionPane.INFORMATION_MESSAGE);
    MessageStore.addMessage(m);
}
}

// Message class with methods for unit-testing
class Message {
    private String id;
    private int number;
    private String recipient;
    private String content;
    private String hash;
    private String status;
    private static int totalCount = 0;

    public Message(String id, int number, String recipient, String content) {
        this.id      = id;
        this.number  = number;
        this.recipient = recipient;
        this.content = content;
        this.hash    = createMessageHash();
    }

    public boolean checkMessageID() {
        return id != null && id.length() <= 10;
    }

    public boolean checkRecipientCell() {
        return recipient != null && recipient.length() <= 11 && recipient.startsWith("+");
    }

    public String createMessageHash() {
        String idPart = id.substring(0, 2);
        int idx      = number - 1;
        String[] w   = content.trim().split("\\s+");
        String first = w.length > 0 ? w[0].replaceAll("\\W", "") : "";
        String last  = w.length > 0 ? w[w.length - 1].replaceAll("\\W", "") : "";
        return String.format("%s:%d:%s", idPart, idx, (first + last).toUpperCase());
    }

    public static String generateId() {
        StringBuilder sb = new StringBuilder(10);
        for (int i = 0; i < 10; i++) sb.append((int)(Math.random() * 10));
        return sb.toString();
    }
}

```

```

}

public void sendMessage(String choice) {
    this.status = choice;
    totalCount++;
}

public String printMessage() {
    return String.format(
        "ID:%s | Hash:%s | To:%s | \"%s\" | Status:%s",
        id, hash, recipient, content, status
    );
}

public int returnTotalMessages() {
    return totalCount;
}

// getters for JSON export
public String getId() { return id; }
public String getHash() { return hash; }
public String getRecipient() { return recipient; }
public String getContent() { return content; }
public String getStatus() { return status; }
}

// Store and export messages
class MessageStore {
    private static Message[] messages;
    private static int count;
    private static final int INITIAL_CAPACITY = 10;

    public static void init() {
        messages = new Message[INITIAL_CAPACITY];
        count = 0;
    }

    public static void addMessage(Message m) {
        if (count == messages.length) {
            // resize array
            Message[] newArr = new Message[messages.length * 2];
            System.arraycopy(messages, 0, newArr, 0, messages.length);
            messages = newArr;
        }
        messages[count++] = m;
    }
}

/* Chatgpt JSON Logic */
public static void exportJson() {
    int export = JOptionPane.showConfirmDialog(
        null,
        "Would you like to export messages to JSON?",
        "Export",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.QUESTION_MESSAGE
    );
    if (export != JOptionPane.YES_OPTION) return;

    JFileChooser chooser = new JFileChooser();

```

```

chooser.setDialogTitle("Save Messages As");
chooser.setFileFilter(new FileNameExtensionFilter("JSON Files (*.json)", "json"));
if (chooser.showSaveDialog(null) != JFileChooser.APPROVE_OPTION) return;

File file = chooser.getSelectedFile();
if (!file.getName().toLowerCase().endsWith(".json")) {
    file = new File(file.getAbsolutePath() + ".json");
}

try (FileWriter writer = new FileWriter(file)) {
    writer.write("[\n");
    for (int i = 0; i < count; i++) {
        Message m = messages[i];
        writer.write(" {\n");
        writer.write("   \"id\": \"" + m.getId() + "\",\n");
        writer.write("   \"hash\": \"" + m.getHash() + "\",\n");
        writer.write("   \"recipient\": \"" + m.getRecipient() + "\",\n");
        writer.write("   \"content\": \"" + m.getContent().replace("\\", "\\\\") + "\",\n");
        writer.write("   \"status\": \"" + m.getStatus() + "\"\n");
        writer.write(" }" + (i < count - 1 ? "," : "") + "\n");
    }
    writer.write("]\n");
    JOptionPane.showMessageDialog(
        null,
        "Exported to " + file.getAbsolutePath(),
        "Export Complete",
        JOptionPane.INFORMATION_MESSAGE
    );
} catch (IOException e) {
    JOptionPane.showMessageDialog(
        null,
        "Export error: " + e.getMessage(),
        "Error",
        JOptionPane.ERROR_MESSAGE
    );
}
}
}
}

```

screenshots

## Section B—continued: Junit Tests code-Full Test Code to Test Methods

```

package com.mycompany.messagingapp;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class Messagingapp_Test {

    //The tests they specified in the assignment
    @Test
    void testHashForDinnerInvite() {
        Message m1 = new Message(
            "0012345678",
            1,

```

```

        "+277118693002",
        "Hi Mike, can you join us for dinner tonight"
    );

assertEquals("00:0:HITONIGHT", m1.createMessageHash());}

//Method specific tests, might be useful for later.
@Test
void testCreateMessageHash() {
    Message m = new Message("AB1234567", 1, "+1234567890", "Hello World");

    assertEquals("AB:0:HELLOWORLD", m.createMessageHash());
}

@Test
void testCheckRecipientCell() {
    Message ok = new Message("ID", 1, "+27831234567", "X");
    assertTrue(ok.checkRecipientCell());
    Message bad = new Message("ID", 1, "27831234567", "X");
    assertFalse(bad.checkRecipientCell());
}

@Test
void testGenerateIdAndCheckMessageID() {
    String id = Message.generateId();
    assertEquals(10, id.length(), "ID must be 10 digits");
    assertTrue(id.matches("\\d{10}"), "ID must be numeric");
    Message m = new Message(id, 1, "+1", "X");
    assertTrue(m.checkMessageID(), "Generated ID should be valid");
}

@Test
void testContentLengthWithinLimit() {
    String shortText = "A".repeat(250);
    assertEquals(250, shortText.length());
    Message m = new Message("ID", 1, "+1", shortText);
    assertTrue(m.getContent().length() <= 250,
               "Content of length 250 should be allowed");
}

@Test
void testContentLengthExceedsLimit() {
    String longText = "A".repeat(251);
    assertEquals(251, longText.length());
    Message m = new Message("ID", 1, "+1", longText);
    assertTrue(m.getContent().length() > 250,
               "Content exceeding 250 chars should be flagged");
}

@Test
void testSendActionSend() {
    Message m = new Message("ID", 1, "+1", "Hi");
}

```

```

m.sendMessage("Send");
assertEquals("Send", m.getStatus());
assertEquals(1, m.returnTotalMessages());
}

@Test
void testSendActionDiscard() {
    Message m = new Message("ID", 1, "+1", "Hi");
    m.sendMessage("Discard");
    assertEquals("Discard", m.getStatus());
    assertEquals(1, m.returnTotalMessages());
}

@Test
void testSendActionStore() {
    Message m = new Message("ID", 1, "+1", "Hi");
    m.sendMessage("Store");
    assertEquals("Store", m.getStatus());
    assertEquals(1, m.returnTotalMessages());
}

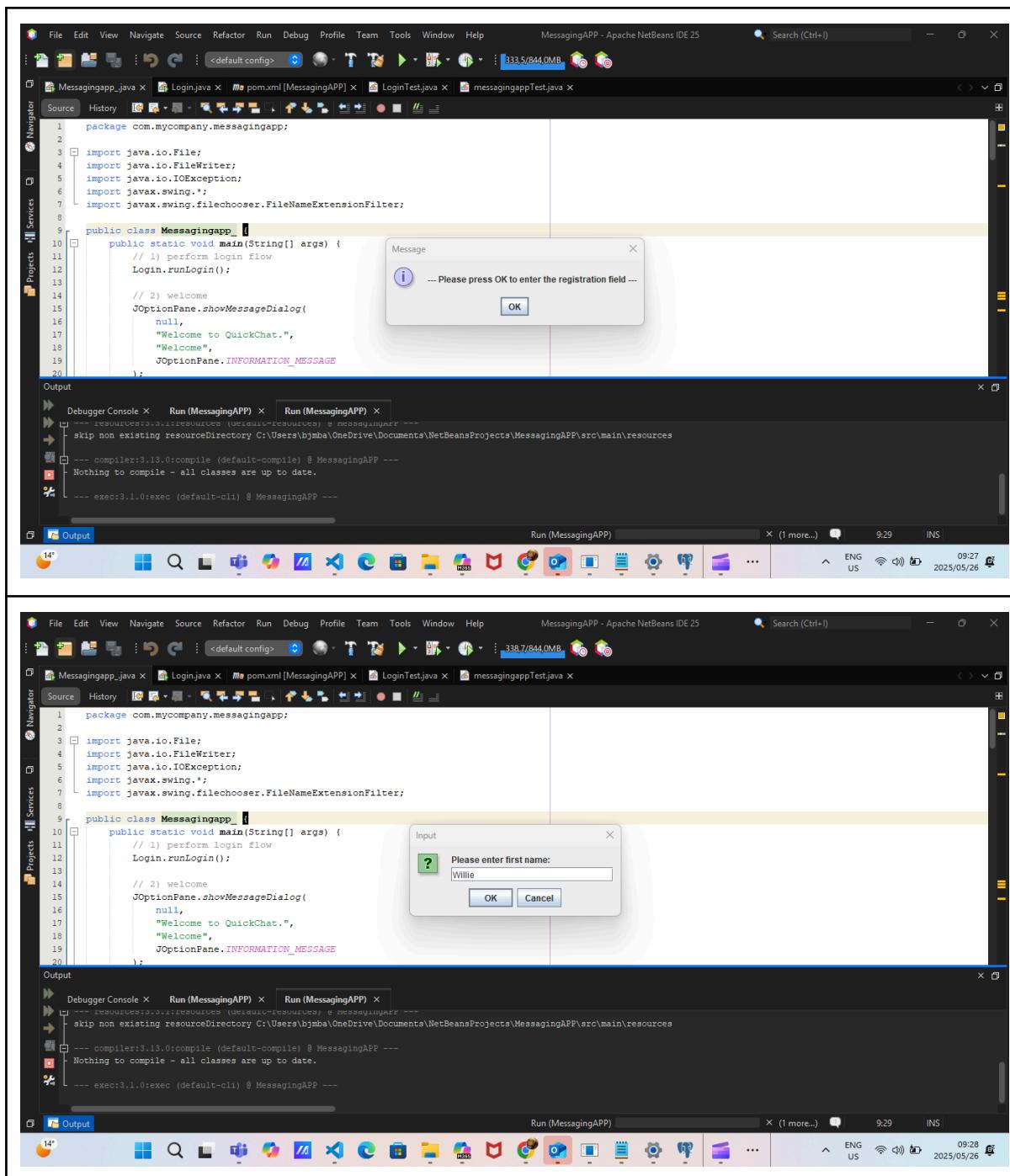
@Test
void testTotalCountAfterMultipleSends() {
    Message m1 = new Message("ID1", 1, "+1", "A");
    Message m2 = new Message("ID2", 2, "+1", "B");
    m1.sendMessage("Send");
    m2.sendMessage("Send");
    assertEquals(2, m2.returnTotalMessages(),
        "After sending twice, totalCount should be 2");
}
}

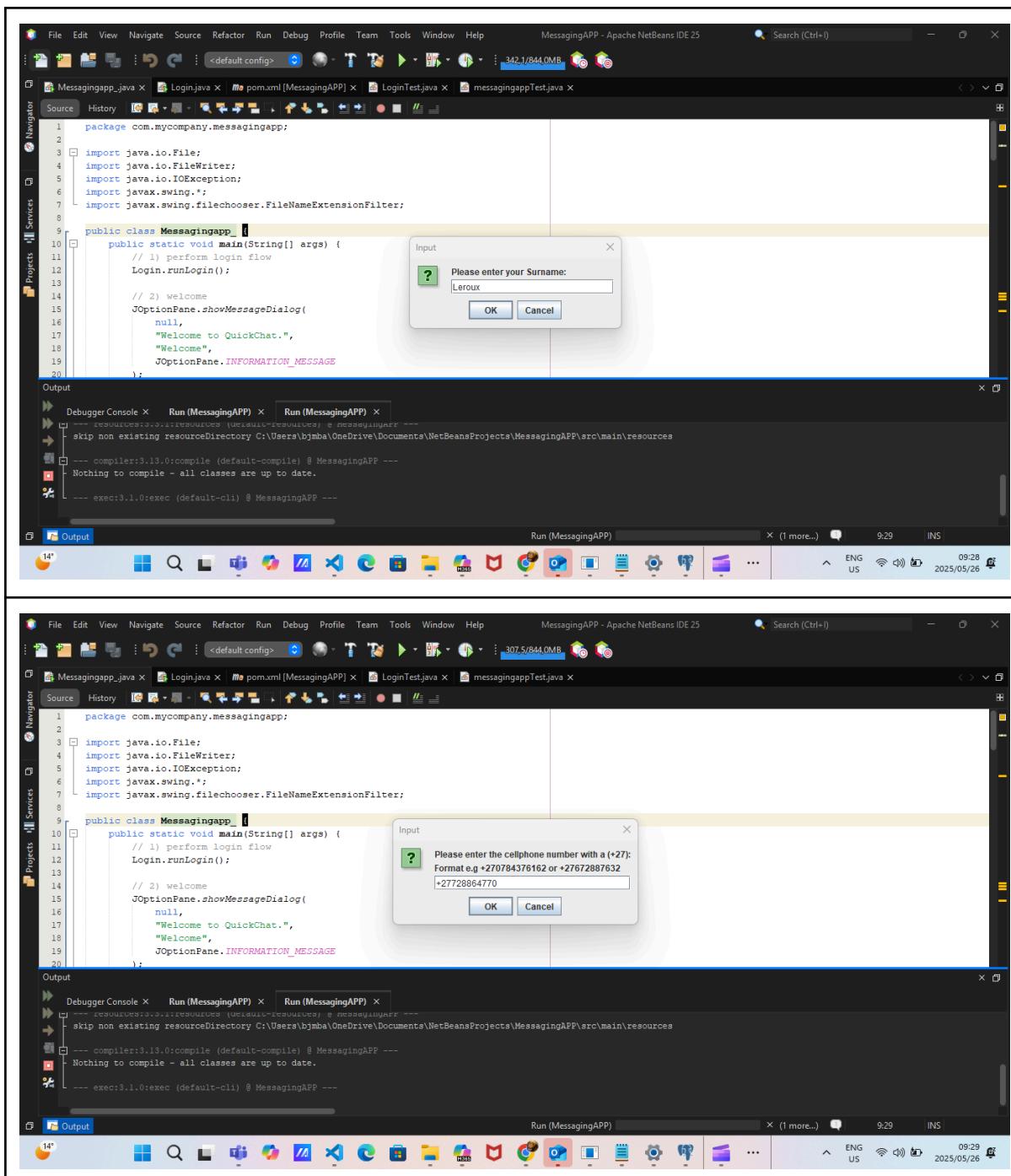
```

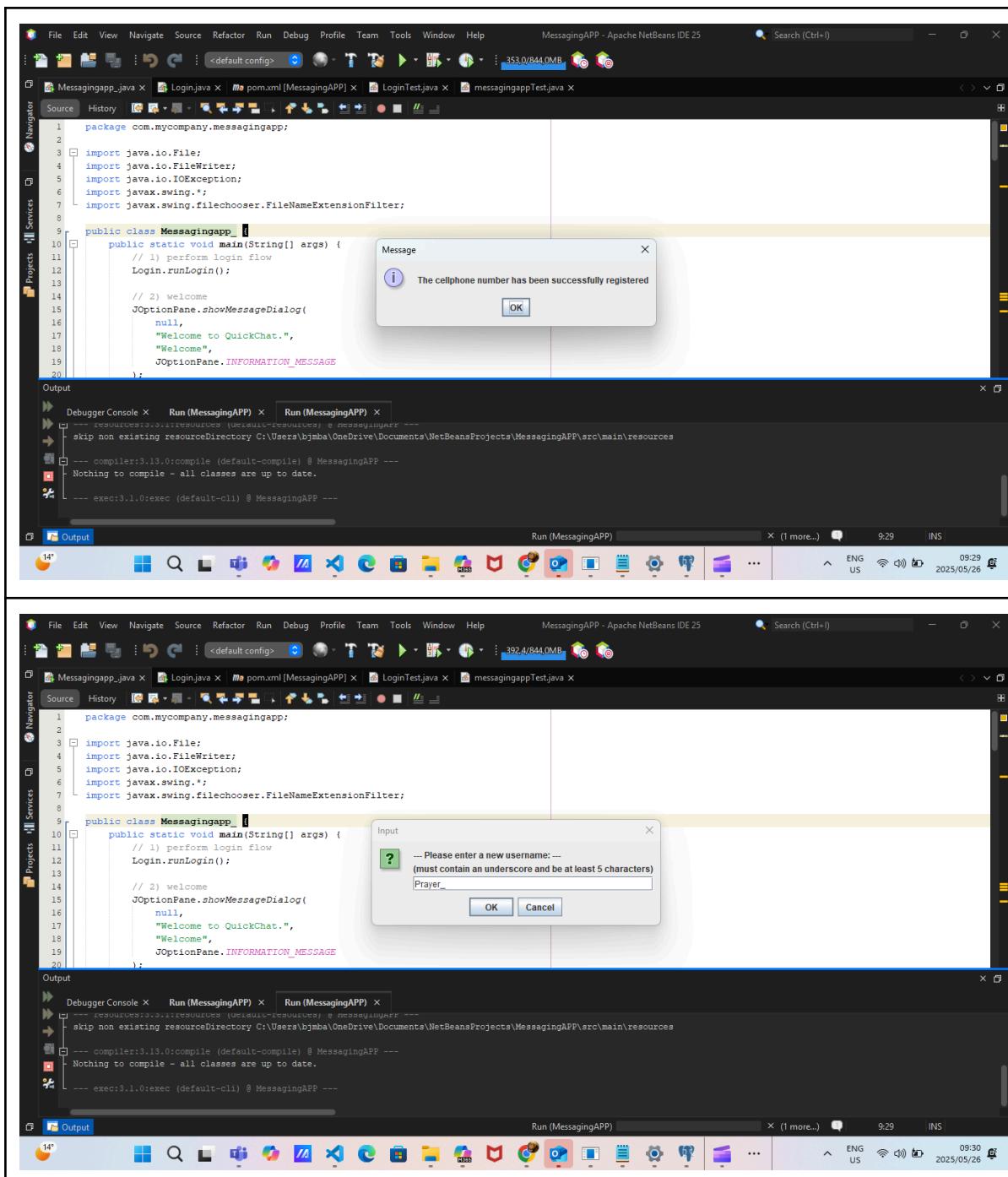
## **SECTION C: Screenshots of Code Running.**

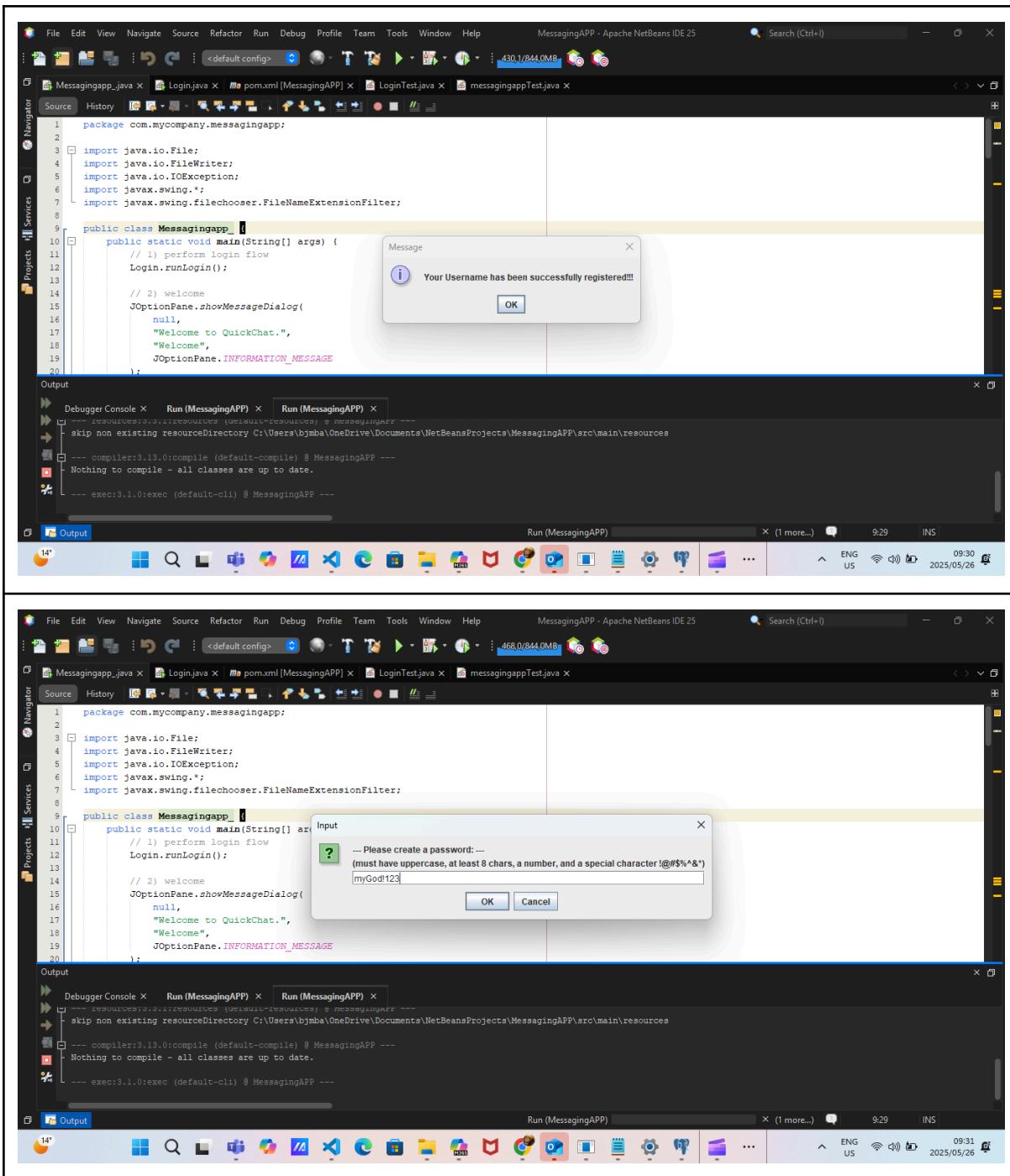
*Code running on Netbeans Screenshots:*

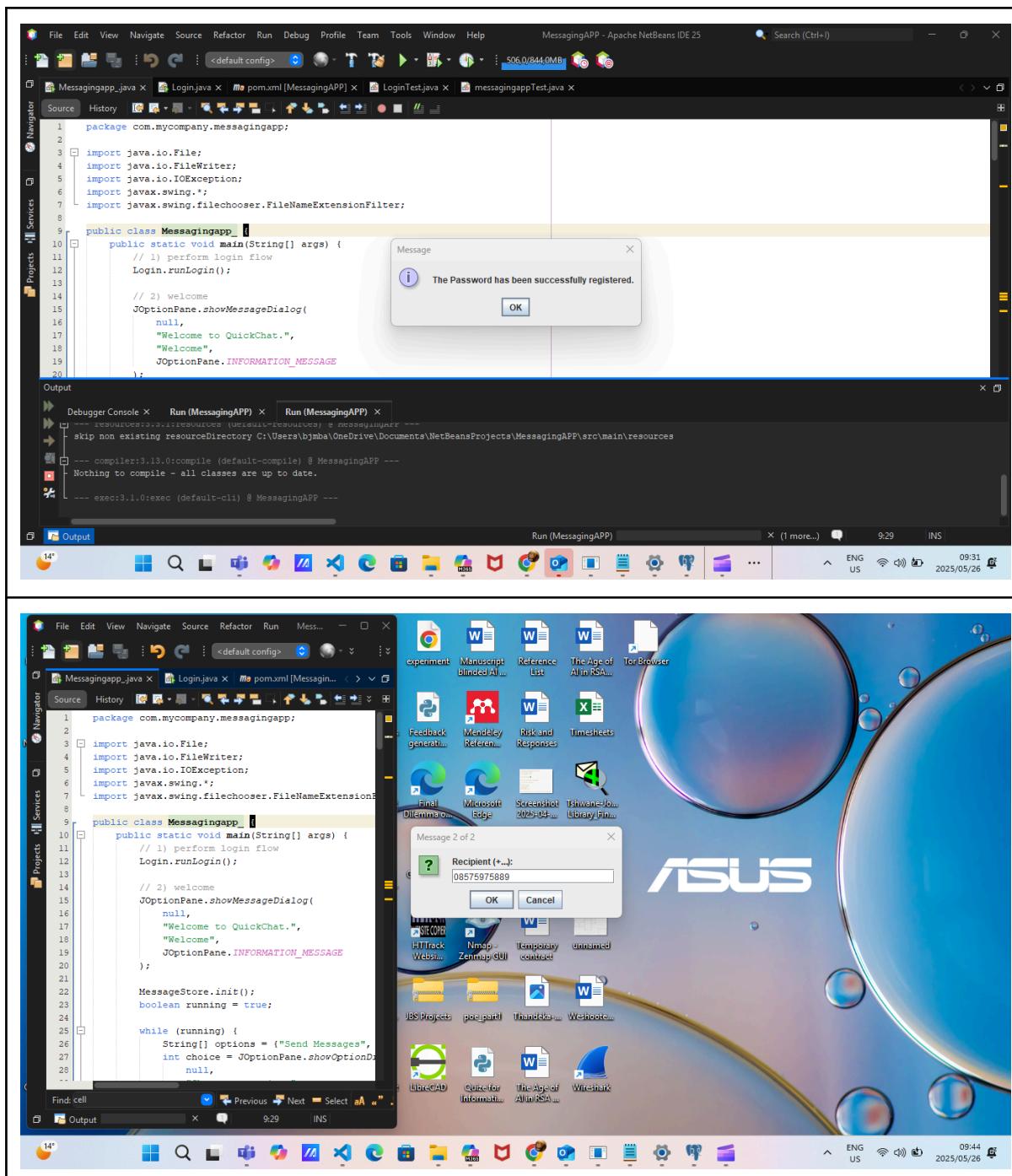
User Login Logic to the Message class.

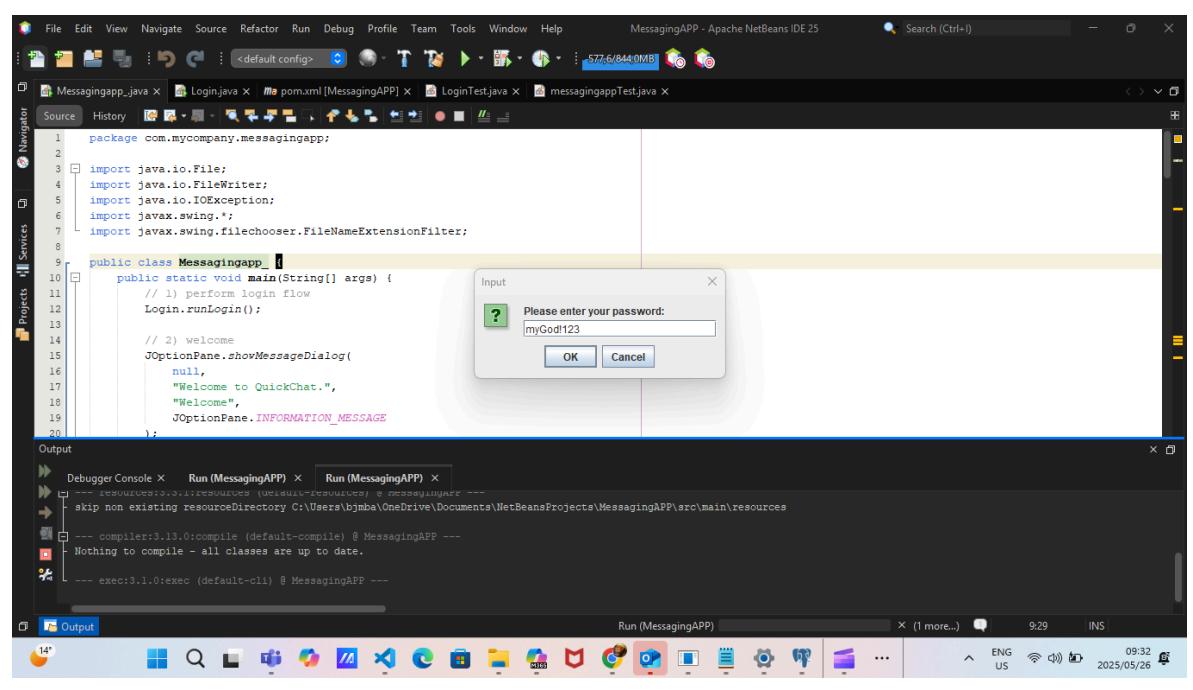












MessagingAPP - Apache NetBeans IDE 25

```

1 package com.mycompany.messagingapp;
2
3 import java.io.File;
4 import java.io.FileWriter;
5 import java.io.IOException;
6 import javax.swing.*;
7 import javax.swing.filechooser.FileNameExtensionFilter;
8
9 public class Messagingapp {
10     public static void main(String[] args) {
11         // 1) perform login flow
12         Login.runLogin();
13
14         // 2) welcome
15         JOptionPane.showMessageDialog(
16             null,
17             "Welcome to QuickChat.",
18             "Welcome",
19             JOptionPane.INFORMATION_MESSAGE
20         );
21     }
22 }

```

Please enter your password:  
myGod123

OK Cancel

Output

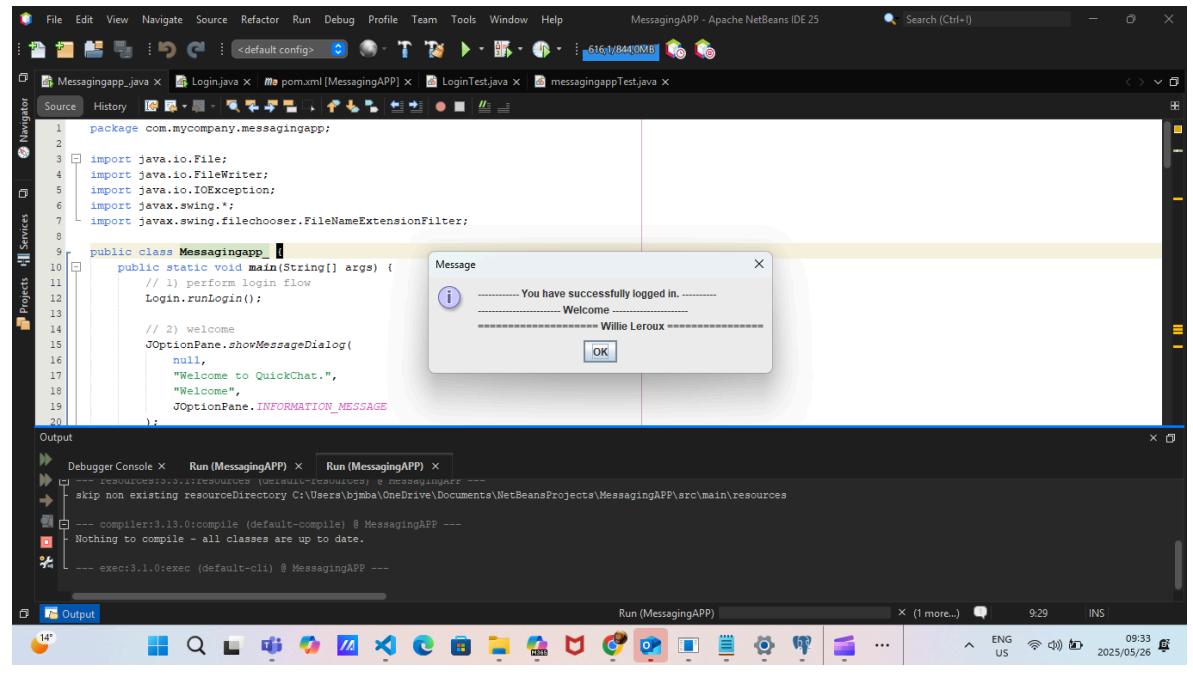
```

> Debugger Console X Run (MessagingAPP) X Run (MessagingAPP) X
> --- resources:jar:resources (default-resources) @ messagingapp ---
> skip non existing resourceDirectory C:\Users\bjmba\OneDrive\Documents\NetBeansProjects\MessagingAPP\src\main\resources
[ ] --- compiler:3.13.0:compile (default-compile) @ MessagingAPP ---
[ ] Nothing to compile - all classes are up to date.
[ ] --- exec:3.1.0:exec (default-cli) @ MessagingAPP ---

```

Run (MessagingAPP) X (1 more...) 9:29 INS

14° ENG US 09:32 2025/05/26

MessagingAPP - Apache NetBeans IDE 25

```

1 package com.mycompany.messagingapp;
2
3 import java.io.File;
4 import java.io.FileWriter;
5 import java.io.IOException;
6 import javax.swing.*;
7 import javax.swing.filechooser.FileNameExtensionFilter;
8
9 public class Messagingapp {
10     public static void main(String[] args) {
11         // 1) perform login flow
12         Login.runLogin();
13
14         // 2) welcome
15         JOptionPane.showMessageDialog(
16             null,
17             "Welcome to QuickChat.",
18             "Welcome",
19             JOptionPane.INFORMATION_MESSAGE
20         );
21     }
22 }

```

Message

----- You have successfully logged in. -----  
----- Welcome -----  
----- Willie Leroux -----

OK

Output

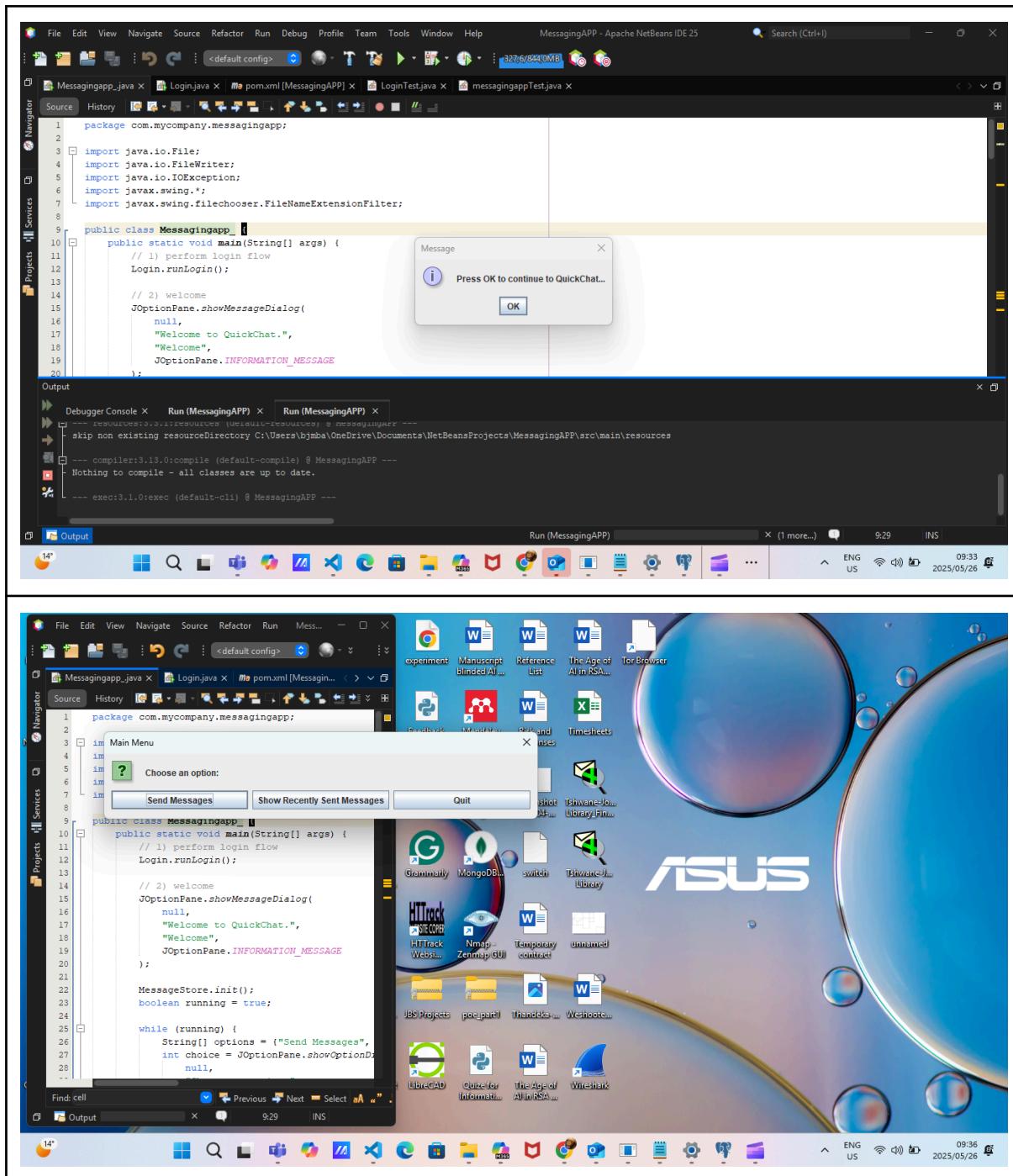
```

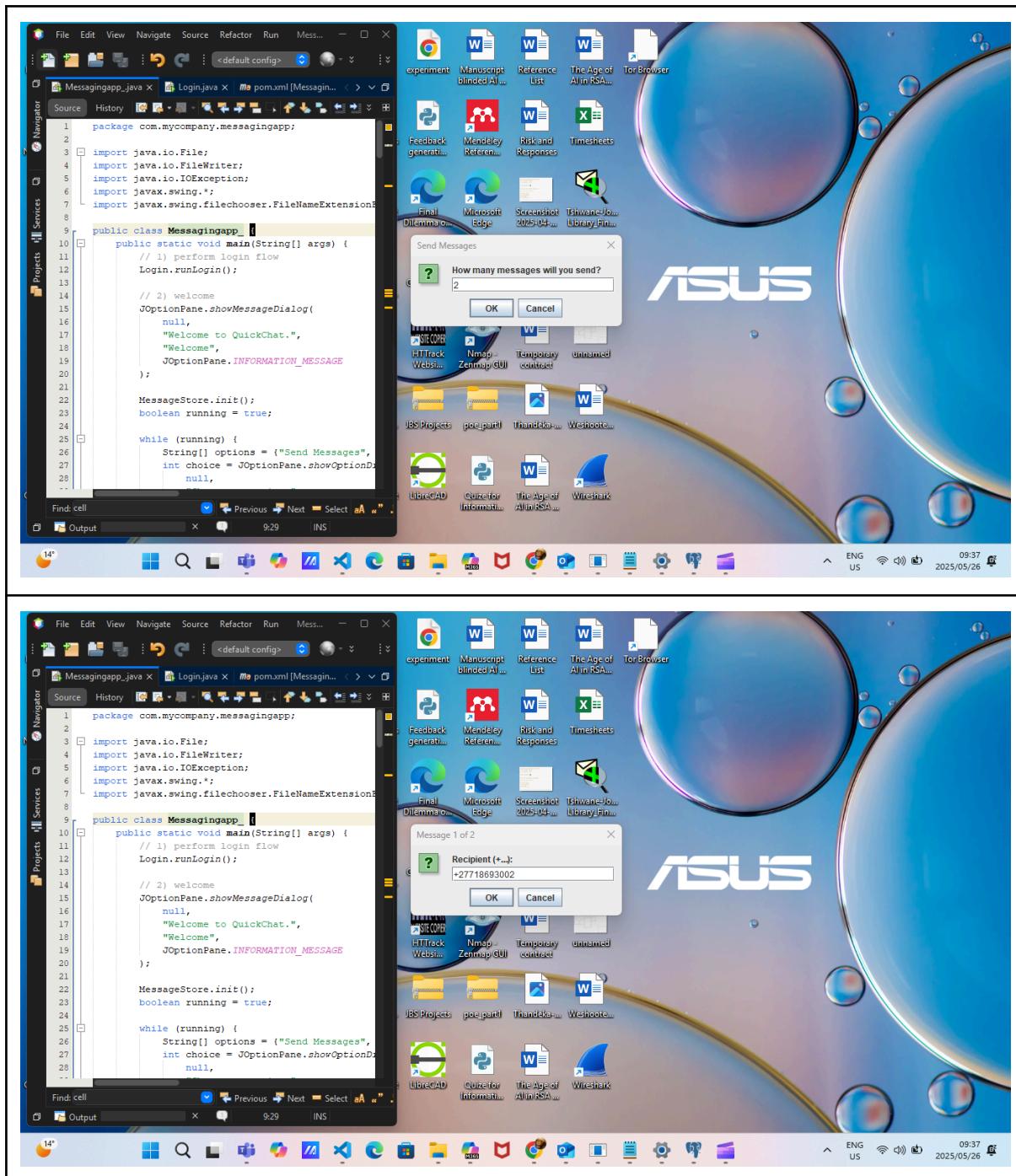
> Debugger Console X Run (MessagingAPP) X Run (MessagingAPP) X
> --- resources:jar:resources (default-resources) @ messagingapp ---
> skip non existing resourceDirectory C:\Users\bjmba\OneDrive\Documents\NetBeansProjects\MessagingAPP\src\main\resources
[ ] --- compiler:3.13.0:compile (default-compile) @ MessagingAPP ---
[ ] Nothing to compile - all classes are up to date.
[ ] --- exec:3.1.0:exec (default-cli) @ MessagingAPP ---

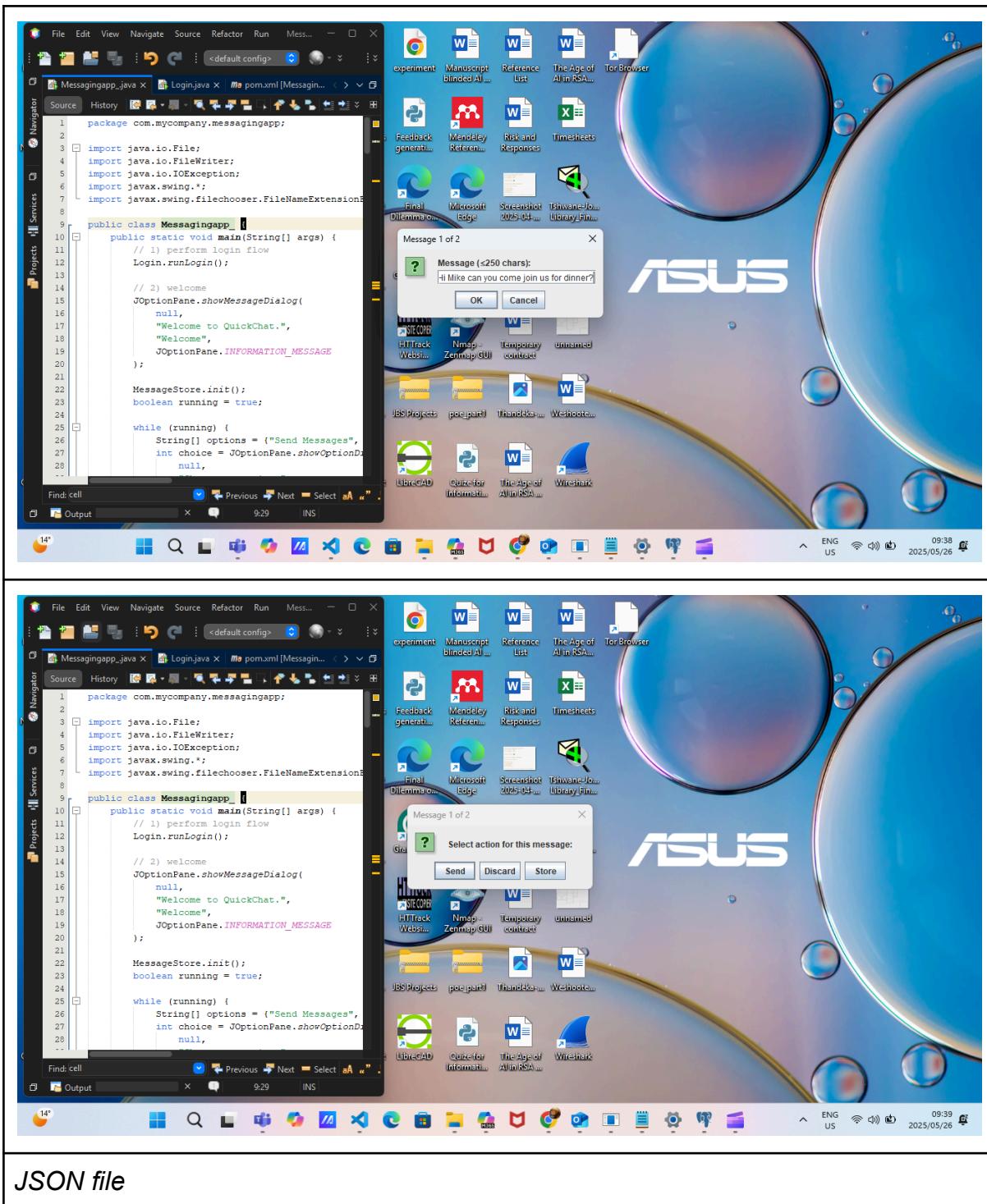
```

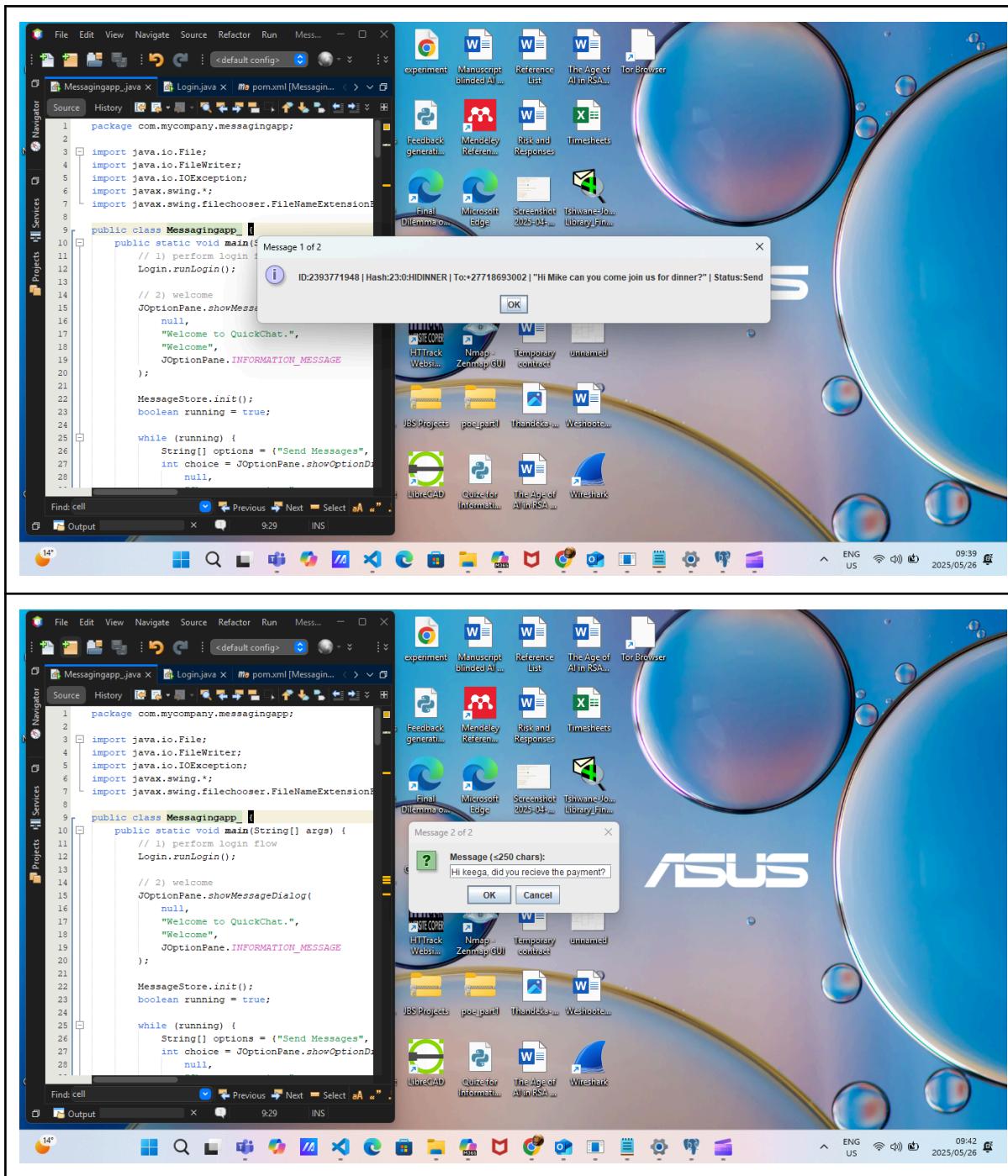
Run (MessagingAPP) X (1 more...) 9:29 INS

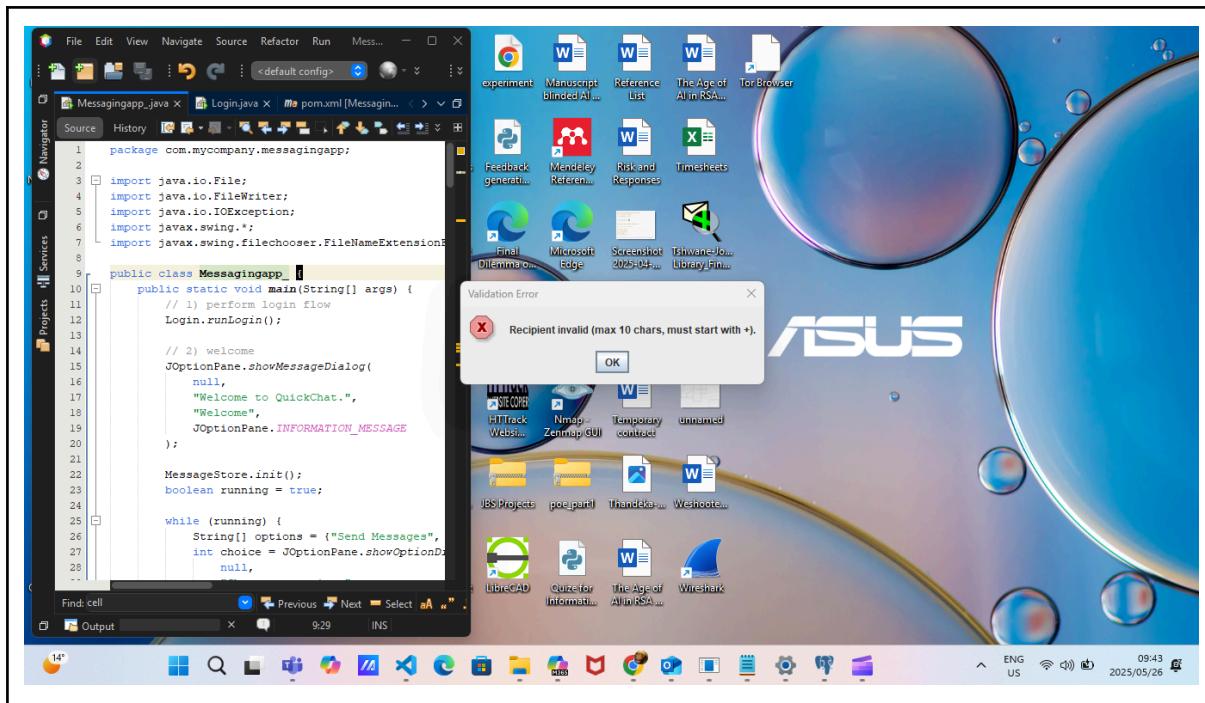
14° ENG US 09:33 2025/05/26











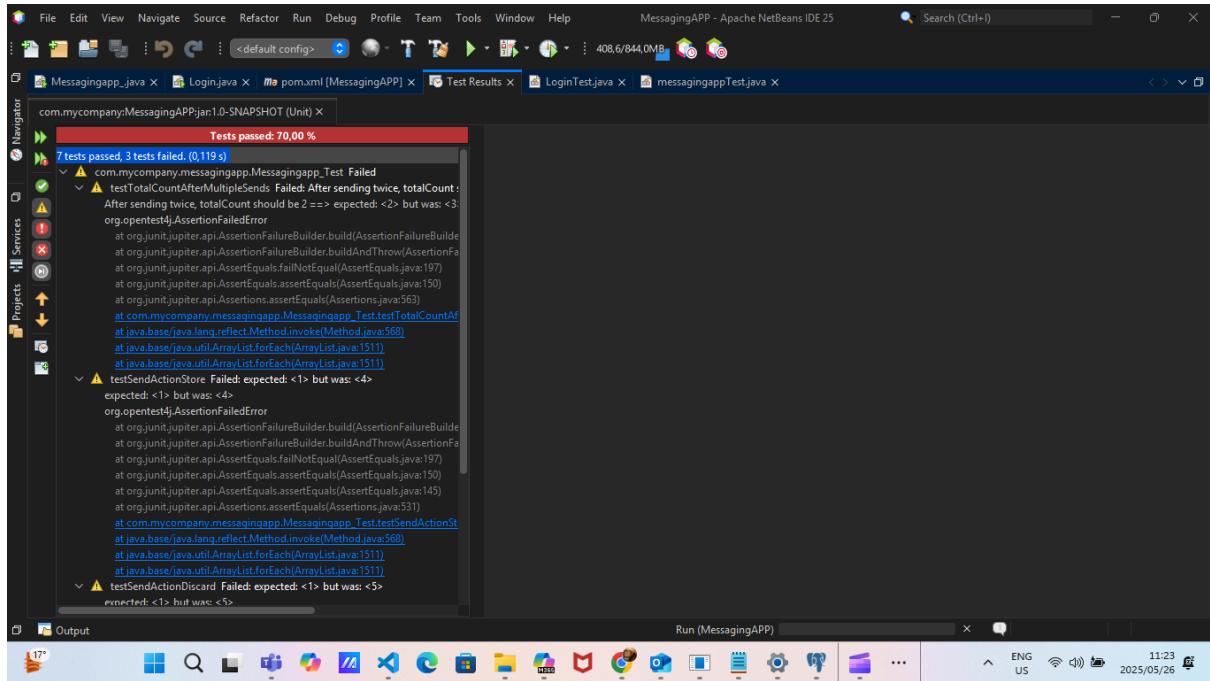
The screenshot shows the Apache NetBeans IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages like Groupwork, JavaApplication10, JavaModularApplication1, Methods practice, POE\_Part1, POE\_Part1\_Version\_1, and Poefortesting.
- Source Editor:** Displays Java code for a class named Poefortestingfinal. The code includes methods for credentials and cellphonenumbers\_input, which handle user input for first name, surname, and phone number, validating them against specific rules.
- Output Window:** Shows the terminal output of the application running. It prompts for a new username, checks if it contains an underscore and is at least 5 characters, and then successfully logs in the user "beautiful human" with the message "You have successfully Logged you in beautiful humanbeing!".
- System Bar:** Shows system information including battery level (433/96600MB), time (17:60), and date (2025/04/22).

This screenshot is identical to the one above, showing the same Java code in the source editor and the same terminal output in the output window. Both screenshots demonstrate the functionality of a Java application for user registration and password length validation.

## SECTION D – SCREENSHOTS AND OUTCOME OF TESTING of full code

### 1. Expanded overall Unit Test



### 2. Individual methods of unit tests, non-failed.

**Screenshot 1 (Top):** NetBeans IDE 25 showing the `messagingappTest.java` file. The code contains test methods for sending messages. The `testTotalCountAfterMultipleSends()` method is highlighted.

```

    Message m = new Message("ID", 1, "+1", "Hi");
    m.sendMessage("Store");
    assertEquals("Store", m.getStatus());
    assertEquals(1, m.returnTotalMessages());
}

@Test
void testTotalCountAfterMultipleSends() {
    Message m1 = new Message("ID1", 1, "+1", "A");
    Message m2 = new Message("ID2", 2, "+1", "B");
    m1.sendMessage("Send");
    m2.sendMessage("Send");
    assertEquals(2, m2.returnTotalMessages(),
        "After sending twice, totalCount should be 2");
}

```

The `Test Results` panel shows "Tests passed: 100.00 %".

**Screenshot 2 (Bottom):** NetBeans IDE 25 showing the `messagingappTest.java` file. The code contains test methods for sending messages. The `testSendActionDiscard()` and `testSendActionStore()` methods are highlighted.

```

void testSendActionDiscard() {
    Message m = new Message("ID", 1, "+1", "Hi");
    m.sendMessage("Discard");
    assertEquals("Discard", m.getStatus());
    assertEquals(1, m.returnTotalMessages());
}

@Test
void testSendActionStore() {
    Message m = new Message("ID", 1, "+1", "Hi");
    m.sendMessage("Store");
    assertEquals("Store", m.getStatus());
    assertEquals(1, m.returnTotalMessages());
}

@Test
void testTotalCountAfterMultipleSends() {
    Message m1 = new Message("ID1", 1, "+1", "A");
    Message m2 = new Message("ID2", 2, "+1", "B");
}

```

The `Test Results` panel shows "Tests passed: 100.00 %".

**Screenshot 1:** Apache NetBeans IDE 25 showing the first set of test cases for the MessagingApp project. The code editor displays three test methods: `testSendActionSend()`, `testSendActionDiscard()`, and `testSendActionStore()`. The `testSendActionDiscard()` method is currently selected. The output window shows "Tests passed: 100.00 %".

```

    @Test
    void testSendActionSend() {
        Message m = new Message("ID", 1, "+1", "Hi");
        m.sendMessage("Send");
        assertEquals("Send", m.getStatus());
        assertEquals(1, m.returnTotalMessages());
    }

    @Test
    void testSendActionDiscard() {
        Message m = new Message("ID", 1, "+1", "Hi");
        m.sendMessage("Discard");
        assertEquals("Discard", m.getStatus());
        assertEquals(1, m.returnTotalMessages());
    }

    @Test
    void testSendActionStore() {
        Message m = new Message("ID", 1, "+1", "Hi");
    }

```

**Screenshot 2:** Apache NetBeans IDE 25 showing the second set of test cases for the MessagingApp project. The code editor displays three test methods: `testSendActionSend()`, `testSendActionDiscard()`, and `testSendActionStore()`. The `testSendActionSend()` method is currently selected. The output window shows "Tests passed: 100.00 %".

```

    assertEquals(251, longText.length());
    Message m = new Message("ID", 1, "+1", longText);
    assertTrue(m.getContent().length() > 250,
               "Content exceeding 250 chars should be flagged");

    @Test
    void testSendActionSend() {
        Message m = new Message("ID", 1, "+1", "Hi");
        m.sendMessage("Send");
        assertEquals("Send", m.getStatus());
        assertEquals(1, m.returnTotalMessages());
    }

    @Test
    void testSendActionDiscard() {
        Message m = new Message("ID", 1, "+1", "Hi");
    }

```

**Screenshot 1 (Top):** Apache NetBeans IDE 25 showing the code for `messagingappTest.java`. The code contains three test methods: `testContentLengthExceedsLimit()`, `testSendActionSend()`, and `testContentLengthWithinLimit()`. The test for `testContentLengthExceedsLimit()` fails with the message "Content exceeding 250 chars should be flagged".

```

54     Message m = new Message("ID", 1, "+1", shortText);
55     assertTrue(m.getContent().length() <= 250,
56                 "Content of length 250 should be allowed");
57 }
58
59
60
61
62 @Test
63 void testContentLengthExceedsLimit() {
64     String longText = "A".repeat(251);
65     assertEquals(251, longText.length());
66     Message m = new Message("ID", 1, "+1", longText);
67     assertTrue(m.getContent().length() > 250,
68                 "Content exceeding 250 chars should be flagged");
69 }
70
71
72
73
74 @Test
75 void testSendActionSend() {
76     Message m = new Message("ID", 1, "+1", "Hi");
77 }
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
545
546
547
548
549
549
550
551
552
553
554
555
556
557
557
558
559
559
560
561
562
563
564
565
566
567
567
568
569
569
570
571
572
573
574
575
576
577
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
607
608
609
609
610
611
612
613
614
615
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
629
629
630
631
632
633
634
635
635
636
637
637
638
639
639
640
641
642
643
644
644
645
646
646
647
648
648
649
649
650
651
652
653
654
655
655
656
657
657
658
659
659
660
661
662
663
664
665
665
666
667
667
668
669
669
670
671
672
673
674
675
675
676
677
677
678
679
679
680
681
682
683
684
685
685
686
687
687
688
689
689
690
691
692
693
694
695
695
696
697
697
698
699
699
700
701
702
703
704
705
705
706
707
707
708
709
709
710
711
712
713
714
715
715
716
717
717
718
719
719
720
721
722
723
724
725
725
726
727
727
728
729
729
730
731
732
733
734
735
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
785
785
786
787
787
788
789
789
790
791
792
793
794
795
795
796
797
797
798
799
799
800
801
802
803
804
805
805
806
807
807
808
809
809
810
811
812
813
814
815
815
816
817
817
818
819
819
820
821
822
823
824
825
825
826
827
827
828
829
829
830
831
832
833
834
835
835
836
837
837
838
839
839
840
841
842
843
844
845
845
846
847
847
848
849
849
850
851
852
853
854
855
855
856
857
857
858
859
859
860
861
862
863
864
865
865
866
867
867
868
869
869
870
871
872
873
874
875
875
876
877
877
878
879
879
880
881
882
883
884
885
885
886
887
887
888
889
889
890
891
892
893
894
895
895
896
897
897
898
899
899
900
901
902
903
904
905
905
906
907
907
908
909
909
910
911
912
913
914
915
915
916
917
917
918
919
919
920
921
922
923
924
925
925
926
927
927
928
929
929
930
931
932
933
934
935
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
954
955
955
956
957
957
958
959
959
960
961
962
963
964
965
965
966
967
967
968
969
969
970
971
972
973
974
975
975
976
977
977
978
979
979
980
981
982
983
984
985
985
986
987
987
988
989
989
990
991
992
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1002
1002
1003
1004
1004
1005
1006
1006
1007
1008
1008
1009
1009
1010
1011
1011
1012
1013
1013
1014
1015
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1905
1906
1907
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
199
```

**Screenshot 1: Apache NetBeans IDE 25 - MessagingAPP Project**

The screenshot shows the Apache NetBeans IDE 25 interface with the "MessagingAPP" project open. The "Source" tab of the messagingappTest.java file is selected, displaying test methods for generating IDs and checking message content lengths. The code editor highlights several assertions and variable declarations. The "Test Results" panel shows a green bar indicating "Tests passed: 100.00 %". The status bar at the bottom right shows "11:37 2025/05/26".

```

38     }
39
40     @Test
41     void testGenerateIdAndCheckMessageID() {
42         String id = Message.generateId();
43         assertEquals(10, id.length(), "ID must be 10 digits");
44         assertTrue(id.matches("\\d{10}"), "ID must be numeric");
45         Message m = new Message(id, 1, "+1", "X");
46         assertTrue(m.checkMessageID(), "Generated ID should be valid");
47     }
48
49
50     @Test
51     void testContentLengthWithinLimit() {
52         String shortText = "a".repeat(250);
53         assertEquals(250, shortText.length());
54         Message m = new Message("ID", 1, "+1", shortText);
55         assertTrue(m.getContent().length() <= 250,
56             "Content of length 250 should be allowed");
57     }

```

**Screenshot 2: Apache NetBeans IDE 25 - MessagingAPP Project**

This screenshot shows the same project setup as the first one, but the "Source" tab of the messagingappTest.java file is selected. It displays different test methods, including ones for creating message hashes and checking recipient cells. The "Test Results" panel again shows 100% test coverage. The status bar at the bottom right shows "43:2 2025/05/26".

```

26     void testCreateMessageHash() {
27         Message m = new Message("AB1234567", 1, "+1234567890", "Hello World");
28
29         assertEquals("AB:0:HELLOWORLD", m.createMessageHash());
30     }
31
32     @Test
33     void testCheckRecipientCell() {
34         Message ok = new Message("ID", 1, "+27831234567", "X");
35         assertTrue(ok.checkRecipientCell());
36         Message bad = new Message("ID", 1, "27831234567", "X");
37         assertFalse(bad.checkRecipientCell());
38     }
39
40     @Test
41     void testGenerateIdAndCheckMessageID() {
42         String id = Message.generateId();
43         assertEquals(10, id.length(), "ID must be 10 digits");
44     }

```

The screenshot displays two instances of the Apache NetBeans IDE interface, each showing a different Java project structure and code editor.

**Top Window (MessagingAPP Project):**

- Project Structure:** Shows Source Packages (com.mycompany.messagingapp) containing Login.java and Messagingapp.java, and Test Packages (com.mycompany.messagingapp) containing LoginTest.java and messagingappTest.java.
- Code Editor:** Displays the content of LoginTest.java, specifically the `testCreateMessageHash()` method which is currently failing.
- Output:** Shows the build output for com.mycompany.MessagingAPP.jar:1.0-SNAPSHOT (Unit) with a green bar indicating Tests passed: 100.00 %.

**Bottom Window (Messagingapp Project):**

- Project Structure:** Shows Source Packages (com.mycompany.messagingapp) containing Login.java and Messagingapp.java, and Test Packages (com.mycompany.messagingapp) containing LoginTest.java and messagingappTest.java.
- Code Editor:** Displays the content of Messagingapp\_Test.java, specifically the `testHashForDinnerInvite()` method which is currently passing.
- Output:** Shows the build output for com.mycompany.MessagingAPP.jar:1.0-SNAPSHOT (Unit) with a green bar indicating Tests passed: 100.00 %.

Both windows have a status bar at the bottom showing system information like battery level, network, and date.

## Bibliography

Farrell, J. 2023. *Java programming*. 10th ed. Boston: Cengage.

OpenAI. 2025. Chat-GPT (Version 4.0). [Large language model]. Available at: <https://chat.openai.com/> [Accessed: 20 March 2024].

