

ACTIVIDAD 6.2 EJERCICIO DE PROGRAMACIÓN 3

PRUEBAS DE SOFTWARE Y ASEGURAMIENTO DE LA CALIDAD

Aarón Cortés García A01730451

Programa: Manage abstractions of Hotel, Customer and Reservation classes.

EVIDENCIA FRAGMENTOS DE LOS CÓDIGOS

Hotel.py

```
hotel.py x
hotel.py > Hotel > modify_hotel
1  """Activity 6.2 - Hotel Class
2  Aarón Cortés García - A01730451
3  """
4
5  import json
6  import os
7
8
9  class Hotel:
10     """Class representing a hotel with room management capabilities."""
11     FILE_NAME = "hotels.json" # File where hotel data is stored
12
13     def __init__(self, name, location, rooms):
14         """Initializes a hotel with a name, location, and a set of room IDs."""
15         self.name = name # Hotel name
16         self.location = location # Hotel location
17         self.rooms = {i: False for i in range(1, rooms+1)} # Rooms with status
18
19     def to_dict(self):
20         """Converts the hotel object into a dictionary for JSON storage."""
21         return {
22             "name": self.name,
23             "location": self.location,
24             "rooms": self.rooms # Dictionary of room statuses
25         }
26
27     @classmethod
28     def save_hotels(cls, hotels):
29         """Saves a list of hotel objects to a JSON file."""
30         with open(cls.FILE_NAME, 'w', encoding='utf-8') as file:
31             json.dump([hotel.to_dict() for hotel in hotels], file, indent=4)
32
```

```
80
81     def reserve_room(self, room_id):
82         """Reserves a specific room by ID if available."""
83         if room_id in self.rooms and not self.rooms[room_id]:
84             self.rooms[room_id] = True # Mark room as reserved
85             self.modify_hotel()
86             return True
87         return False # Room is already reserved or does not exist
88
89     def cancel_reservation(self, room_id):
90         """Cancels a reservation for a specific room by ID."""
91         if room_id in self.rooms and self.rooms[room_id]:
92             self.rooms[room_id] = False # Mark room as available
93             self.modify_hotel()
94             return True
95         return False # Room is not reserved or does not exist
96
```

Customer.py

customer.py X

customer.py > Customer > to_dict

```
1  """Activity 6.2 - Customer Class
2  |   Aarón Cortés García - A01730451
3  |   """
4
5  import json
6  import os
7
8
9  class Customer:
10     """Class representing a customer in the hotel system."""
11     FILE_NAME = "customers.json" # File where customer data is stored
12
13     def __init__(self, customer_id, name, email, phone):
14         """Initializes a customer with his fields"""
15         self.customer_id = customer_id # Unique identifier for the customer
16         self.name = name # Full name of the customer
17         self.email = email # Contact email address
18         self.phone = phone # Contact phone number
19
20     def to_dict(self):
21         """Converts the customer object into a dictionary for JSON storage."""
22         return {
23             "customer_id": self.customer_id,
24             "name": self.name,
25             "email": self.email,
26             "phone": self.phone
27         }
28
```

```
61
62     @classmethod
63     def get_customer(cls, customer_id):
64         """Retrieves a customer by ID if it exists."""
65         for customer in cls.load_customers():
66             if customer.customer_id == customer_id:
67                 return customer # Return the matching customer
68         return None # Return None if not found
69
70     def modify_customer(self, name=None, email=None, phone=None):
71         """Updates customer details such as name, email, or phone number."""
72         if name:
73             self.name = name # Update name if provided
74         if email:
75             self.email = email # Update email if provided
76         if phone:
77             self.phone = phone # Update phone if provided
78         customers = self.load_customers()
79         for i, customer in enumerate(customers):
80             if customer.customer_id == self.customer_id:
81                 customers[i] = self # Replace old customer with updated data
82                 break
83         self.save_customers(customers) # Save the updated list to file
84
```

Reservation.py

```
reservation.py X
reservation.py > Reservation > _init_
1  """Activity 6.2 - Reservation Class
2      Aarón Cortés García - A01730451
3  """
4
5  import json
6  import os
7  from hotel import Hotel
8  from customer import Customer
9
10
11 class Reservation:
12     """Class representing a reservation in the hotel system."""
13     FILE_NAME = "reservations.json" # File where reservation data is stored
14
15     def __init__(self, reserv_id, customer, hotel, room_id):
16         """Initializes a reservation with customer, hotel, and room ID."""
17         self.reserv_id = reserv_id # Unique ID for reservation
18         self.customer = customer # Customer object who made the reservation
19         self.hotel = hotel # Hotel object where the reservation is made
20         self.room_id = room_id # ID of the reserved room
21
22     def to_dict(self):
23         """Converts the reservation object into a dict. for JSON storage."""
24         return {
25             "reserv_id": self.reserv_id,
26             "customer_id": self.customer.customer_id,
27             "hotel_name": self.hotel.name,
28             "room_id": self.room_id
29         }
30
31     @classmethod
32     def save_reservations(cls, reservations):
33         """Saves a list of reservation objects to a JSON file."""
34         with open(cls.FILE_NAME, 'w', encoding='utf-8') as file:
35             json.dump([reservation.to_dict() for reservation in reservations],
36                       file, indent=4)
37
```

```
74
75     @classmethod
76     def cancel_reservation(cls, reserv_id):
77         """Cancels an existing reservation and frees the room."""
78         reservations = cls.load_reservations()
79         reserv_to_cancel = None
80
81         for reservation in reservations:
82             if reservation.reserv_id == reserv_id:
83                 reserv_to_cancel = reservation
84                 break
85
86         if reserv_to_cancel:
87             reserv_to_cancel.hotel.cancel_reservation(reserv_to_cancel.room_id)
88             reservations.remove(reserv_to_cancel)
89             cls.save_reservations(reservations)
90             print(f"Reservation {reserv_id} cancelled successfully.")
91         else:
92             print(f"Reservation with ID {reserv_id} not found.")
93
94     @classmethod
95     def get_reservation(cls, reserv_id):
96         """Retrieves a reservation by ID if it exists."""
97         for reservation in cls.load_reservations():
98             if reservation.reserv_id == reserv_id:
99                 return reservation
100         return None # Return None if not found
```

Main.py

```
main.py X
main.py > ...
1  """Activity 6.2 - Main menu for managing hotels,
2      customers, and reservations. Aarón Cortés
3      García - A01730451
4      """
5
6  from hotel import Hotel
7  from customer import Customer
8  from reservation import Reservation
9
10
11  def display_menu():
12      """Displays the main menu options."""
13      print("\nHotel Management System")
14      print("1. Create Hotel")
15      print("2. Modify Hotel")
16      print("3. Delete Hotel")
17      print("4. Create Customer")
18      print("5. Modify Customer")
19      print("6. Delete Customer")
20      print("7. Make Reservation")
21      print("8. Cancel Reservation")
22      print("0. Exit")
23
24
25  def handle_create_hotel():
26      """Handles the creation of a new hotel."""
27      name = input("Enter hotel name: ")
28      location = input("Enter hotel location: ")
29      rooms = int(input("Enter number of rooms: "))
30      Hotel.create_hotel(name, location, rooms)
31      print("Hotel created successfully.")
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81  def handle_delete_customer():
82      """Handles customer deletion."""
83      customer_id = int(input("Enter customer ID to delete: "))
84      Customer.delete_customer(customer_id)
85      print("Customer deleted successfully.")
86
87
88  def handle_make_reservation():
89      """Handles making a reservation."""
90      reserv_id = int(input("Enter reservation ID: "))
91      customer_id = int(input("Enter customer ID: "))
92      hotel_name = input("Enter hotel name: ")
93      room_id = int(input("Enter room ID: "))
94      Reservation.create_reservation(reserv_id, customer_id, hotel_name, room_id)
95      print("Reservation created successfully.")
96
97
98  def handle_cancel_reservation():
99      """Handles reservation cancellation."""
100     reserv_id = int(input("Enter reservation ID to cancel: "))
101     Reservation.cancel_reservation(reserv_id)
102     print("Reservation canceled successfully.")
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
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Unit_tests.py

```
unit_tests.py X
unit_tests.py > TestHotelSystem
1  """Activity 6.2 - Unit test cases
2  |   Aarón Cortés García - A01730451
3  |   """
4
5  import unittest
6  import os
7  from hotel import Hotel
8  from customer import Customer
9  from reservation import Reservation
10
11 class TestHotelSystem(unittest.TestCase):
12     """Unit tests for the Hotel, Customer, and Reservation classes."""
13
14     def setUp(self):
15         """Setup test environment by clearing JSON files."""
16         for file in [Hotel.FILE_NAME, Customer.FILE_NAME, Reservation.FILE_NAME]:
17             if os.path.exists(file):
18                 os.remove(file)
19
20     def test_create_hotel(self):
21         """Test creating and retrieving a hotel."""
22         Hotel.create_hotel("FiestaInn", "Puebla", 10)
23         hotel = Hotel.get_hotel("FiestaInn")
24         self.assertIsNotNone(hotel)
25         self.assertEqual(hotel.name, "FiestaInn")
26         self.assertEqual(hotel.location, "Puebla")
27         self.assertEqual(len(hotel.rooms), 10)
28
29     def test_modify_hotel(self):
30         """Test modifying a hotel's details."""
31         Hotel.create_hotel("Grand Americana", "CDMX", 8)
32         hotel = Hotel.get_hotel("Grand Americana")
33         hotel.modify_hotel(location="Monterrey")
34         hotel = Hotel.get_hotel("Grand Americana")
35         self.assertEqual(hotel.location, "Monterrey")
36
37     def test_create_reservation(self):
38         """Test creating a reservation with a non-existent hotel ID."""
39         Reservation.create_reservation(2, 99, "NonExistentHotel", 1)
40         reservation = Reservation.get_reservation(2)
41         self.assertIsNone(reservation) # No reservation should be created
42
43     def test_get_reservation(self):
44         """Test retrieving an existing reservation by its ID."""
45         # Create a hotel and a customer
46         Hotel.create_hotel("Grand Plaza", "Veracruz", 12)
47         hotel = Hotel.get_hotel("Grand Plaza")
48         Customer.create_customer(40, "Raúl Sánchez", "raul@webmail.com", "2215667788")
49         customer = Customer.get_customer(40)
50
51         # Create a reservation
52         Reservation.create_reservation(4, customer.customer_id, hotel.name, 10)
53         reservation = Reservation.get_reservation(4)
54         self.assertIsNotNone(reservation)
55         self.assertEqual(reservation.reserv_id, 4)
56
57     def tearDown(self):
58         """Clean up JSON files after tests."""
59         for file in [Hotel.FILE_NAME, Customer.FILE_NAME, Reservation.FILE_NAME]:
60             if os.path.exists(file):
61                 os.remove(file)
62
63 if __name__ == "__main__":
64     unittest.main()
65
```

EVIDENCIA EVALUACIONES CON PYLINT

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> pylint hotel.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> pylint customer.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> pylint reservation.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> pylint main.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> |
```

EVIDENCIA EVALUACIONES CON FLAKE8

```
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> flake8 hotel.py
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> flake8 customer.py
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> flake8 reservation.py
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> flake8 main.py
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
```

EVIDENCIA DE EJECUCIÓN DE EJEMPLO EN CONSOLA (Inputs manuales)

```
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
(base) PS C:\Users\aaaron\Documents\Maestria\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> python main.py
o
Hotel Management System
1. Create Hotel
2. Modify Hotel
3. Delete Hotel
4. Create Customer
5. Modify Customer
6. Delete Customer
7. Make Reservation
8. Cancel Reservation
0. Exit
Enter your choice: 1
Enter hotel name: FiestaInn
Enter hotel location: Puebla
Enter number of rooms: 8
Hotel created successfully.

Hotel Management System
1. Create Hotel
2. Modify Hotel
3. Delete Hotel
4. Create Customer
5. Modify Customer
6. Delete Customer
```

EVIDENCIA DE ARCHIVO DE ENTRADA DE EJEMPLO GENERADO

```
input_test.txt X
input_test.txt
1 1
2 FiestaInn
3 Puebla
4 10
5 1
6 Grand Americana
7 CDMX
8 8
9 4
10 101
11 John Doe
12 johndoe@email.com
13 5551234567
14 7
15 -
```

EVIDENCIA DE EJECUCIÓN DE EJEMPLO EN CONSOLA (Redireccionando archivo de entrada para los inputs)

```
C:\Users\aaaron\Documents\Maestría\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>python main.py < input_test.txt

Hotel Management System
1. Create Hotel
2. Modify Hotel
3. Delete Hotel
4. Create Customer
5. Modify Customer
6. Delete Customer
7. Make Reservation
8. Cancel Reservation
9. Exit
Enter your choice: Enter hotel name: Enter hotel location: Enter number of rooms: Hotel created successfully.

Hotel Management System
1. Create Hotel
```

EVIDENCIA DE ARCHIVOS JSON FINALES GENERADOS

Hotels.json

```
hotels.json X
hotels.json > ...
1 [
2   {
3     "name": "FiestaInn ",
4     "location": "Puebla",
5     "rooms": {
6       "1": false,
7       "2": false,
8       "3": false,
9       "4": false,
10      "5": false,
11      "6": false,
12      "7": false,
13      "8": false,
14      "9": false,
15      "10": false
16    }
17  },
18 ]
```

Customers.json

```
{ } customers.json X
{ } customers.json > ...
1  [
2      {
3          "customer_id": 101,
4          "name": "Johnathan Doe",
5          "email": "johndoe@gmail.com",
6          "phone": "5559876543"
7      }
8  ]
```

Reservations.json

```
{ } reservations.json X
{ } reservations.json > ...
1  [
2      {
3          "reserv_id": 9,
4          "customer_id": 101,
5          "hotel_name": "Grand Americana",
6          "room_id": 3
7      }
8  ]
```

EVIDENCIA DE UNIT TESTS EN CONSOLA

```
python unit_tests.p
(base) PS C:\Users\aaaron\Documents\Maestría\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad> python unit_tests.p
y
Reservation 3 created successfully.
Reservation 3 cancelled successfully.
...Reservation 1 created successfully.
.Customer or Hotel not found. Reservation not created.
...Reservation 4 created successfully.
....
-----
Ran 11 tests in 0.135s

OK
(base) PS C:\Users\aaaron\Documents\Maestría\Pruebas de software y aseguramiento de calidad\Act 6.2 Ejercicio de programación 3 y Pruebas de Unidad>
```