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

hotel.py > Hotel > modify_hotel

```python
"""Activity 6.2 - Hotel Class
   Aarón Cortés García - A01730451
"""

import json
import os


class Hotel:
    """Class representing a hotel with room management capabilities."""
    FILE_NAME = "hotels.json"  # File where hotel data is stored

    def __init__(self, name, location, rooms):
        """Initializes a hotel with a name, location, and a set of room IDs."""
        self.name = name  # Hotel name
        self.location = location  # Hotel location
        self.rooms = {i: False for i in range(1, rooms+1)}  # Rooms with status

    def to_dict(self):
        """Converts the hotel object into a dictionary for JSON storage."""
        return {
            "name": self.name,
            "location": self.location,
            "rooms": self.rooms  # Dictionary of room statuses
        }

    @classmethod
    def save_hotels(cls, hotels):
        """Saves a list of hotel objects to a JSON file."""
        with open(cls.FILE_NAME, 'w', encoding='utf-8') as file:
            json.dump([hotel.to_dict() for hotel in hotels], file, indent=4)

```

```python
    def reserve_room(self, room_id):
        """Reserves a specific room by ID if available."""
        if room_id in self.rooms and not self.rooms[room_id]:
            self.rooms[room_id] = True  # Mark room as reserved
            self.modify_hotel()
            return True
        return False  # Room is already reserved or does not exist

    def cancel_reservation(self, room_id):
        """Cancels a reservation for a specific room by ID."""
        if room_id in self.rooms and self.rooms[room_id]:
            self.rooms[room_id] = False  # Mark room as available
            self.modify_hotel()
            return True
        return False  # Room is not reserved or does not exist

```

Customer.py

customer.py

```python
"""Activity 6.2 - Customer Class
   Aarón Cortés García - A01730451
"""

import json
import os


class Customer:
    """Class representing a customer in the hotel system."""
    FILE_NAME = "customers.json"  # File where customer data is stored

    def __init__(self, customer_id, name, email, phone):
        """Initializes a customer with his fields"""
        self.customer_id = customer_id  # Unique identifier for the customer
        self.name = name  # Full name of the customer
        self.email = email  # Contact email address
        self.phone = phone  # Contact phone number

    def to_dict(self):
        """Converts the customer object into a dictionary for JSON storage."""
        return {
            "customer_id": self.customer_id,
            "name": self.name,
            "email": self.email,
            "phone": self.phone
        }
```

```python

    @classmethod
    def get_customer(cls, customer_id):
        """Retrieves a customer by ID if it exists."""
        for customer in cls.load_customers():
            if customer.customer_id == customer_id:
                return customer  # Return the matching customer
        return None  # Return None if not found

    def modify_customer(self, name=None, email=None, phone=None):
        """Updates customer details such as name, email, or phone number."""
        if name:
            self.name = name  # Update name if provided
        if email:
            self.email = email  # Update email if provided
        if phone:
            self.phone = phone  # Update phone if provided
        customers = self.load_customers()
        for i, customer in enumerate(customers):
            if customer.customer_id == self.customer_id:
                customers[i] = self  # Replace old customer with updated data
                break
        self.save_customers(customers)  # Save the updated list to file
```

Reservation.py

reservation.py

```python
"""Activity 6.2 - Reservation Class
   Aarón Cortés García - A01730451
"""

import json
import os
from hotel import Hotel
from customer import Customer


class Reservation:
    """Class representing a reservation in the hotel system."""
    FILE_NAME = "reservations.json"  # File where reservation data is stored

    def __init__(self, reserv_id, customer, hotel, room_id):
        """Initializes a reservation with customer, hotel, and room ID."""
        self.reserv_id = reserv_id  # Unique ID for reservation
        self.customer = customer  # Customer object who made the reservation
        self.hotel = hotel  # Hotel object where the reservation is made
        self.room_id = room_id  # ID of the reserved room

    def to_dict(self):
        """Converts the reservation object into a dict. for JSON storage."""
        return {
            "reserv_id": self.reserv_id,
            "customer_id": self.customer.customer_id,
            "hotel_name": self.hotel.name,
            "room_id": self.room_id
        }

    @classmethod
    def save_reservations(cls, reservations):
        """Saves a list of reservation objects to a JSON file."""
        with open(cls.FILE_NAME, 'w', encoding='utf-8') as file:
            json.dump([reservation.to_dict() for reservation in reservations],
                      file, indent=4)
```

```python
    @classmethod
    def cancel_reservation(cls, reserv_id):
        """Cancels an existing reservation and frees the room."""
        reservations = cls.load_reservations()
        reserv_to_cancel = None

        for reservation in reservations:
            if reservation.reserv_id == reserv_id:
                reserv_to_cancel = reservation
                break

        if reserv_to_cancel:
            reserv_to_cancel.hotel.cancel_reservation(reserv_to_cancel.room_id)
            reservations.remove(reserv_to_cancel)
            cls.save_reservations(reservations)
            print(f"Reservation {reserv_id} cancelled successfully.")
        else:
            print(f"Reservation with ID {reserv_id} not found.")

    @classmethod
    def get_reservation(cls, reserv_id):
        """Retrieves a reservation by ID if it exists."""
        for reservation in cls.load_reservations():
            if reservation.reserv_id == reserv_id:
                return reservation
        return None  # Return None if not found
```

Main.py

main.py

```python
"""Activity 6.2 - Main menu for managing hotels,
   customers, and reservations. Aarón Cortés
   García - A01730451
"""

from hotel import Hotel
from customer import Customer
from reservation import Reservation


def display_menu():
    """Displays the main menu options."""
    print("\nHotel Management System")
    print("1. Create Hotel")
    print("2. Modify Hotel")
    print("3. Delete Hotel")
    print("4. Create Customer")
    print("5. Modify Customer")
    print("6. Delete Customer")
    print("7. Make Reservation")
    print("8. Cancel Reservation")
    print("0. Exit")


def handle_create_hotel():
    """Handles the creation of a new hotel."""
    name = input("Enter hotel name: ")
    location = input("Enter hotel location: ")
    rooms = int(input("Enter number of rooms: "))
    Hotel.create_hotel(name, location, rooms)
    print("Hotel created successfully.")
```

```python
def handle_delete_customer():
    """Handles customer deletion."""
    customer_id = int(input("Enter customer ID to delete: "))
    Customer.delete_customer(customer_id)
    print("Customer deleted successfully.")


def handle_make_reservation():
    """Handles making a reservation."""
    reserv_id = int(input("Enter reservation ID: "))
    customer_id = int(input("Enter customer ID: "))
    hotel_name = input("Enter hotel name: ")
    room_id = int(input("Enter room ID: "))
    Reservation.create_reservation(reserv_id, customer_id, hotel_name, room_id)
    print("Reservation created successfully.")


def handle_cancel_reservation():
    """Handles reservation cancellation."""
    reserv_id = int(input("Enter reservation ID to cancel: "))
    Reservation.cancel_reservation(reserv_id)
    print("Reservation canceled successfully.")

```

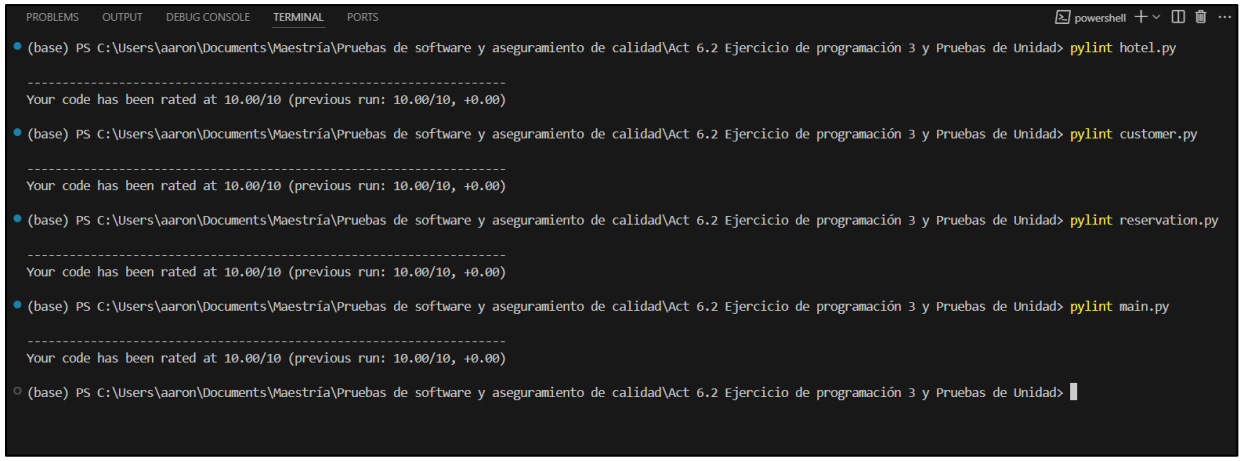
Unit_tests.py

unit_tests.py

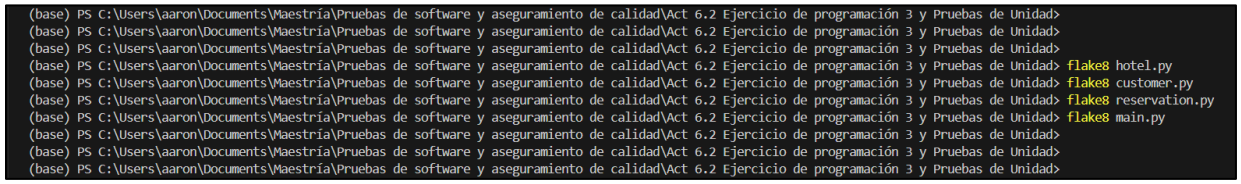
```python
"""Activity 6.2 - Unit test cases
   Aarón Cortés García - A01730451
"""

import unittest
import os
from hotel import Hotel
from customer import Customer
from reservation import Reservation

class TestHotelSystem(unittest.TestCase):
    """Unit tests for the Hotel, Customer, and Reservation classes."""

    def setUp(self):
        """Setup test environment by clearing JSON files."""
        for file in [Hotel.FILE_NAME, Customer.FILE_NAME, Reservation.FILE_NAME]:
            if os.path.exists(file):
                os.remove(file)

    def test_create_hotel(self):
        """Test creating and retrieving a hotel."""
        Hotel.create_hotel("FiestaInn", "Puebla", 10)
        hotel = Hotel.get_hotel("FiestaInn")
        self.assertIsNotNone(hotel)
        self.assertEqual(hotel.name, "FiestaInn")
        self.assertEqual(hotel.location, "Puebla")
        self.assertEqual(len(hotel.rooms), 10)

    def test_modify_hotel(self):
        """Test modifying a hotel's details."""
        Hotel.create_hotel("Grand Americana", "CDMX", 8)
        hotel = Hotel.get_hotel("Grand Americana")
        hotel.modify_hotel(location="Monterrey")
        hotel = Hotel.get_hotel("Grand Americana")
        self.assertEqual(hotel.location, "Monterrey")
```

```python
        Reservation.create_reservation(2, 99, "NonExistentHotel", 1)
        reservation = Reservation.get_reservation(2)
        self.assertIsNone(reservation)  # No reservation should be created

    def test_get_reservation(self):
        """Test retrieving an existing reservation by its ID."""
        # Create a hotel and a customer
        Hotel.create_hotel("Grand Plaza", "Veracruz", 12)
        hotel = Hotel.get_hotel("Grand Plaza")
        Customer.create_customer(40, "Raúl Sánchez", "raul@webmail.com", "2215667788")
        customer = Customer.get_customer(40)

        # Create a reservation
        Reservation.create_reservation(4, customer.customer_id, hotel.name, 10)
        reservation = Reservation.get_reservation(4)
        self.assertIsNotNone(reservation)
        self.assertEqual(reservation.reserv_id, 4)

    def tearDown(self):
        """Clean up JSON files after tests."""
        for file in [Hotel.FILE_NAME, Customer.FILE_NAME, Reservation.FILE_NAME]:
            if os.path.exists(file):
                os.remove(file)

if __name__ == "__main__":
    unittest.main()
```

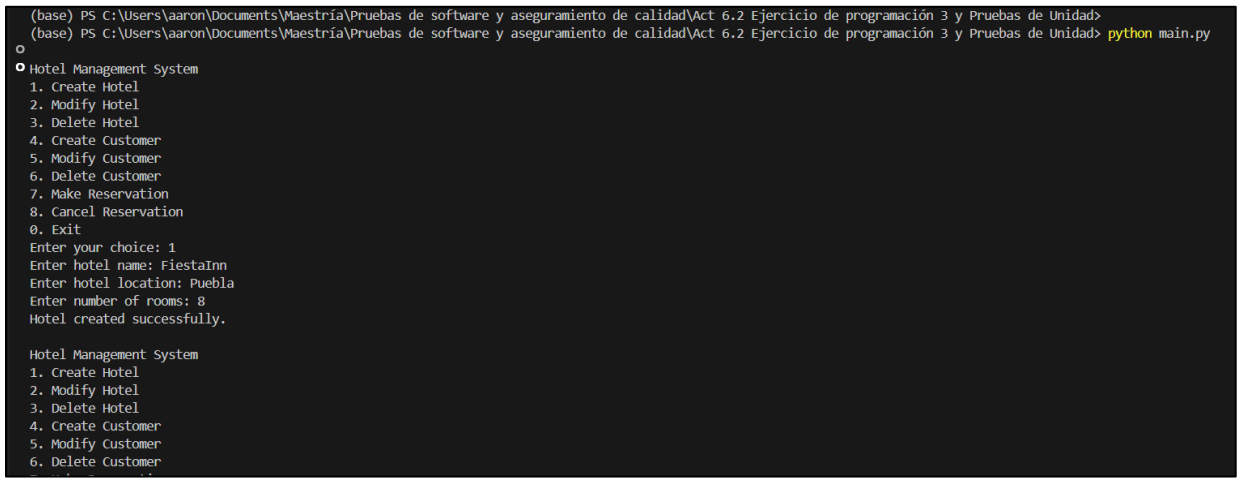
EVIDENCIA EVALUACIONES CON PYLINT



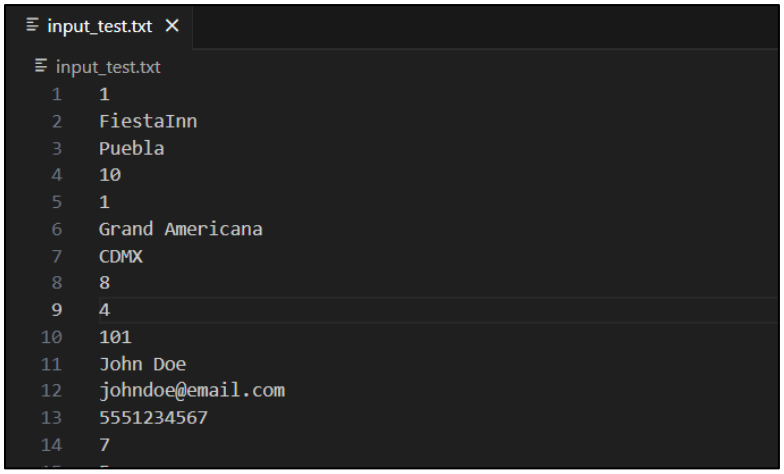
EVIDENCIA EVALUACIONES CON FLAKE8



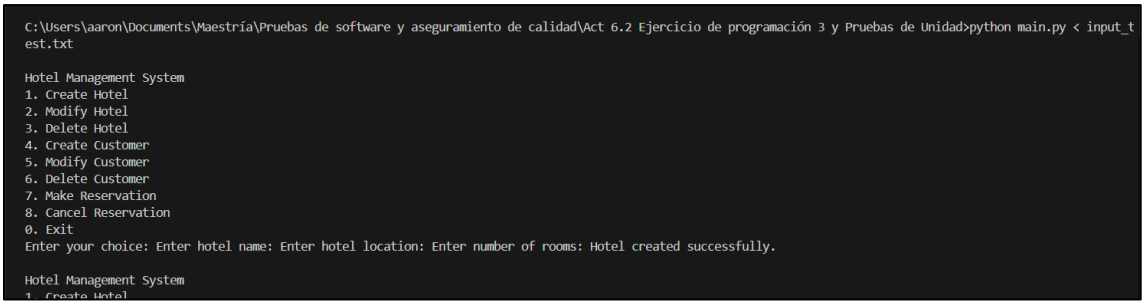
EVIDENCIA DE EJECUCIÓN DE EJEMPLO EN CONSOLA (Inputs manuales)



EVIDENCIA DE ARCHIVO DE ENTRADA DE EJEMPLO GENERADO

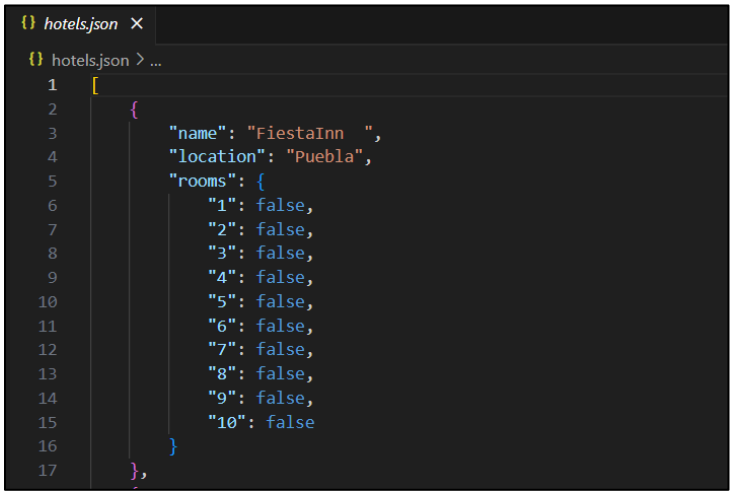


EVIDENCIA DE EJECUCIÓN DE EJEMPLO EN CONSOLA (Redireccionando archivo de entrada para los inputs)

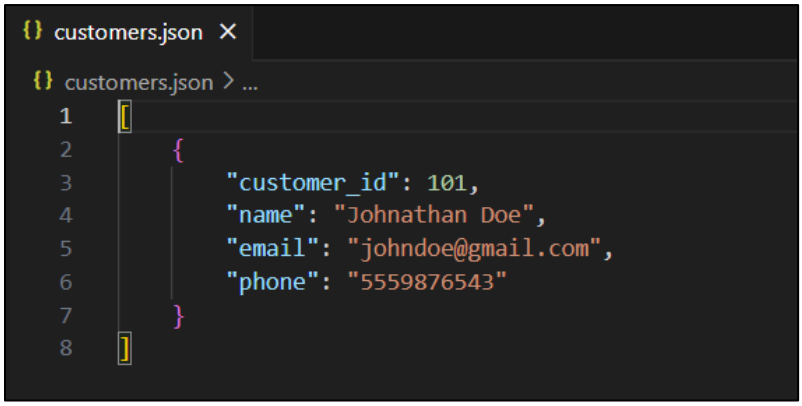


EVIDENCIA DE ARCHIVOS JSON FINALES GENERADOS

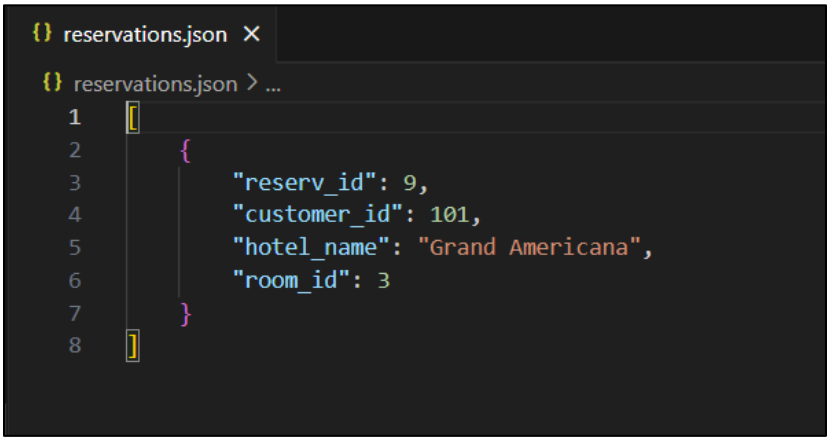
Hotels.json



Customers.json



Reservations.json



EVIDENCIA DE UNIT TESTS EN CONSOLA