Alejandro Juárez Corona

A01168444

Pruebas de software y aseguramiento de la calidad

Actividad 6.2

Ejercicio de programación 3. Pruebas unitarias

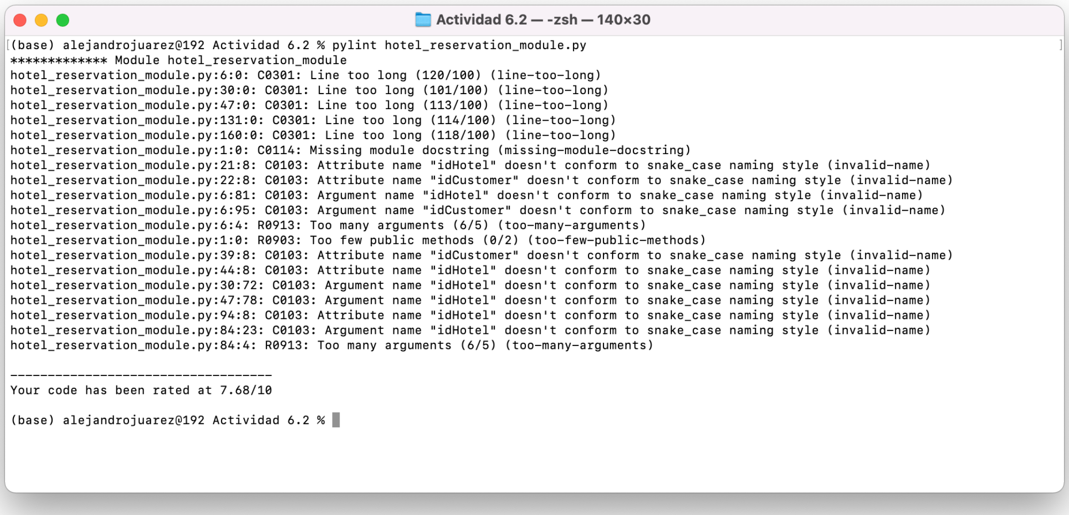
Modulo encargado de definir Hotel, Reservation y Customer:

```
"""

Module to handle a Hotel Reservation System

"""

# pylint: disable=R0903


class Reservation:
    """Class representing a reservation in a hotel."""

    reservation_counter: int = 1

    def __init__(self,
                 check_in_date: str,
                 check_out_date: str,
                 id_hotel: int,
                 id_customer: int) -> None:
        """Initialize a Reservation object.

        Args:
            check_in_date (str): Check-in date in the format 'YYYY-MM-
DD'.
            check_out_date (str): Check-out date in the format 'YYYY-
MM-DD'.
            id_hotel (int): The ID of the hotel.
            id_customer (int): The ID of the customer.
        """
        self.reservation_id: int = Reservation.reservation_counter
        Reservation.reservation_counter += 1
        self.check_in_date: str = check_in_date
        self.check_out_date: str = check_out_date
        self.id_hotel: int = id_hotel
        self.id_customer: int = id_customer


class Customer:
    """Class representing a customer in a hotel."""

    customer_counter: int = 1

    def __init__(self, name: str,
                 age: int,
                 elite_status: bool = False,
                 id_hotel: int = None) -> None:
        """Initialize a Customer object.

        Args:
            name (str): The name of the customer.
            age (int): The age of the customer.
            elite_status (bool): Whether the customer has elite
status.
            id_hotel (int): The ID of the hotel.
        """
        self.id_customer: int = Customer.customer_counter
        Customer.customer_counter += 1
        self.name: str = name
        self.age: int = age
        self.elite_status: bool = elite_status
        self.id_hotel: int = id_hotel

    @classmethod
    def create_customer(cls, name: str,
                        age: int,
                        elite_status: bool = False,
                        id_hotel: int = None) -> 'Customer':
        """Create a new Customer instance.

        Args:
            name (str): The name of the customer.
            age (int): The age of the customer.
            elite_status (bool): Whether the customer has elite
status.
            id_hotel (int): The ID of the hotel. Defaults to None.

        Returns:
            Customer: A new Customer instance.
        """
        return cls(name, age, elite_status, id_hotel)

    def delete_customer(self, customer_list: list) -> None:
        """Delete the customer from the provided list.

        Args:
            customer_list:List of customers to delete the customer.
        """
        customer_list.remove(self)

    def display_info(self) -> None:
        """Display information about the customer."""
        print(f"Customer ID: {self.id_customer}")
        print(f"Customer: {self.name}")
        print(f"Age: {self.age}")
        print(f"Elite Status: {'Yes' if self.elite_status else 'No'}")

    def display_customer_info(self) -> None:
        """Display information about the customer."""
        self.display_info()

    def update_elite_status(self, new_status: bool) -> None:
        """Update the elite status of the customer.

        Args:
            new_status (bool): The new elite status.
        """
        self.elite_status = new_status


class Hotel:
    """Class representing a hotel."""

    def __init__(self, id_hotel: int,
                 name: str,
                 address: str,
                 capacity: int) -> None:
        """Initialize a Hotel object.

        Args:
            id_hotel (int): The ID of the hotel.
            name (str): The name of the hotel.
            address (str): The address of the hotel.
            capacity (int): The capacity of the hotel.
        """
        self.id_hotel: int = id_hotel
        self.name: str = name
        self.address: str = address
        self.stars: int = 0
        self.capacity: int = capacity
        self.reservations: list = []

    def create_hotel(self, name: str,
                     address: str,
                     stars: int,
                     capacity: int) -> None:
        """Create or modify a hotel with the provided information.

        Args:
            name (str): The name of the hotel.
            address (str): The address of the hotel.
            stars (int): The star rating of the hotel.
            capacity (int): The capacity of the hotel.
        """
        self.name = name
        self.address = address
        self.stars = stars
        self.capacity = capacity

    def delete_hotel(self) -> None:
        """Delete the hotel."""
        del self

    def display_info(self) -> None:
        """Display information about the hotel."""
        print(f"Hotel ID: {self.id_hotel}")
        print(f"Hotel: {self.name}")
        print(f"Address: {self.address}")
        print(f"Stars: {self.stars}")
        print(f"Capacity: {self.capacity} guests")
        print("Reservations:")
        for reservation in self.reservations:
            one =
f"{reservation.reservation_id},{reservation.guest_name}, "
            two =
f"{reservation.check_in_date},{reservation.check_out_date}"
            print(one + two)

    def modify_info(self, name: str = None,
                    address: str = None,
                    stars: int = None,
                    capacity: int = None) -> None:
        """Modify the information of the hotel.

        Args:
            name (str): The new name of the hotel. Defaults to None.
            address (str): The new address of the hotel. Defaults to
None.
            stars (int): The new star rating of the hotel. Defaults to
None.
            capacity (int): The new capacity of the hotel. Defaults to
None.
        """
        if name:
            self.name = name
        if address:
            self.address = address
        if stars:
            self.stars = stars
        if capacity:
            self.capacity = capacity

    def reserve_room(self, check_in_date: str, check_out_date: str) ->
int:
        """Reserve a room in the hotel.

        Args:
            check_in_date (str): Check-in date in the format 'YYYY-MM-
DD'.
            check_out_date (str): Check-out date in the format 'YYYY-
MM-DD'.

        Returns:
            int: The ID of the reservation.
        """
        reservation = Reservation(check_in_date,
                                  check_out_date,
                                  self.id_hotel,
                                  len(self.reservations) + 1)
        self.reservations.append(reservation)
        return reservation.reservation_id

    def cancel_reservation(self, reservation_id: int) -> bool:
        """Cancel a reservation in the hotel.
        Args:
            reservation_id (int): The ID of the reservation to be
canceled.

        Returns:
            bool: True if the reservation is canceled, False
otherwise.
        """
        for reservation in self.reservations:
            if reservation.reservation_id == reservation_id:
                self.reservations.remove(reservation)
                return True
        return False
```

Módulo encargado de las pruebas unitarias

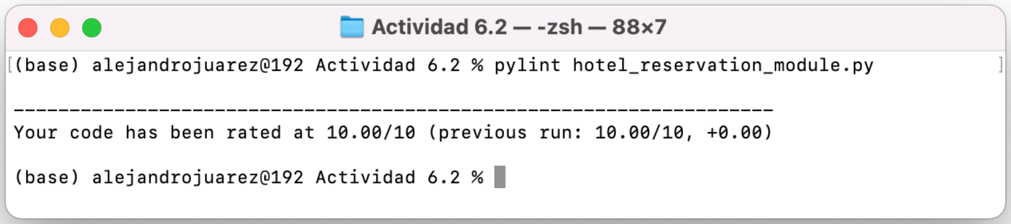
```
import unittest
from datetime import date

from hotel_reservation_module import Hotel, Customer, Reservation

class TestHotelReservationSystem(unittest.TestCase):

    def setUp(self):
        # Create instances of the necessary classes for the tests
        self.hotel = Hotel(1, "Example Hotel", "123 Main St", 100)
        self.customer = Customer.create_customer("John Doe", 30)
        self.reservation = Reservation("2024-03-01", "2024-03-05", 1,
1)

    def test_reserve_room(self):
        # Ensure a room can be reserved successfully
        reservation_id = self.hotel.reserve_room("2024-03-01", "2024-
03-05")
        self.assertEqual(len(self.hotel.reservations), 1)

    def test_cancel_reservation(self):
        # Ensure a reservation can be canceled successfully
        self.hotel.reservations.append(self.reservation)
        result =
self.hotel.cancel_reservation(self.reservation.reservation_id)
        self.assertTrue(result)
        self.assertEqual(len(self.hotel.reservations), 0)

    def test_create_customer(self):
        # Ensure a customer can be created successfully
        self.assertEqual(self.customer.name, "John Doe")
        self.assertEqual(self.customer.age, 30)

    def test_update_elite_status(self):
        # Ensure the customer's elite status can be updated
successfully
        self.customer.update_elite_status(True)
        self.assertTrue(self.customer.elite_status)

    def test_display_info(self):
        # Ensure information can be displayed correctly
        self.customer.display_info()

    def test_display_customer_info(self):
        # Ensure customer information can be displayed correctly
        self.customer.display_customer_info()

    def test_modify_hotel_info(self):
        # Ensure hotel information can be modified correctly
        self.hotel.modify_info(name="New Name", stars=5)
        self.assertEqual(self.hotel.name, "New Name")
        self.assertEqual(self.hotel.stars, 5)

    def test_delete_customer(self):
        # Ensure a customer can be deleted correctly from the list of
customers
        customer_list = [self.customer]
        self.customer.delete_customer(customer_list)
        self.assertEqual(len(customer_list), 0)


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

Chequeo con Pylint



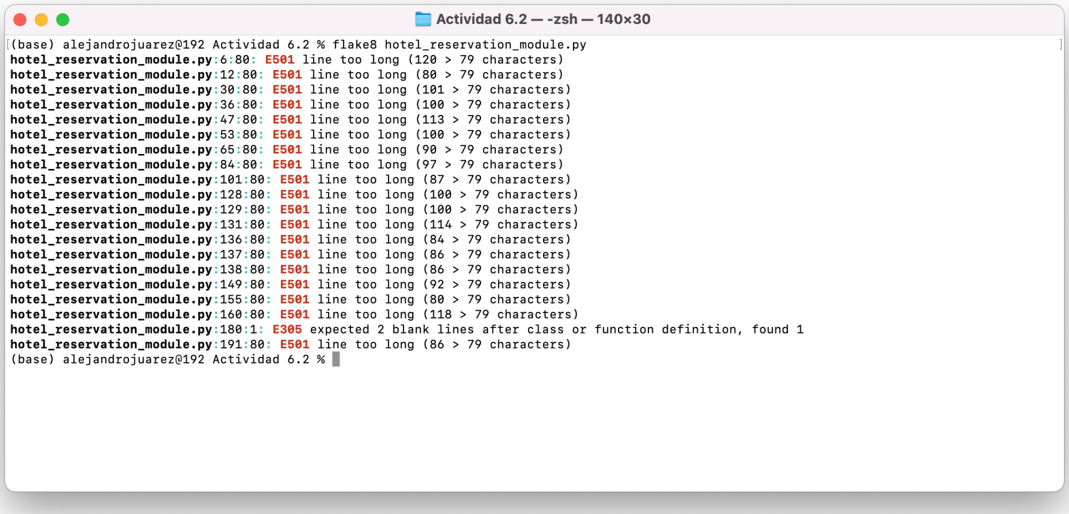
```
(base) alejandrojuarez@192 Actividad 6.2 % pylint hotel_reservation_module.py
************* Module hotel_reservation_module
hotel_reservation_module.py:6:0: C0301: Line too long (120/100) (line-too-long)
hotel_reservation_module.py:30:0: C0301: Line too long (101/100) (line-too-long)
hotel_reservation_module.py:47:0: C0301: Line too long (113/100) (line-too-long)
hotel_reservation_module.py:131:0: C0301: Line too long (114/100) (line-too-long)
hotel_reservation_module.py:160:0: C0301: Line too long (118/100) (line-too-long)
hotel_reservation_module.py:1:0: C0114: Missing module docstring (missing-module-docstring)
hotel_reservation_module.py:21:8: C0103: Attribute name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:22:8: C0103: Attribute name "idCustomer" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:6:81: C0103: Argument name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:6:95: C0103: Argument name "idCustomer" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:6:4: R0913: Too many arguments (6/5) (too-many-arguments)
hotel_reservation_module.py:1:0: R0903: Too few public methods (0/2) (too-few-public-methods)
hotel_reservation_module.py:39:8: C0103: Attribute name "idCustomer" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:44:8: C0103: Attribute name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:30:72: C0103: Argument name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:47:78: C0103: Argument name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:94:8: C0103: Attribute name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:84:23: C0103: Argument name "idHotel" doesn't conform to snake_case naming style (invalid-name)
hotel_reservation_module.py:84:4: R0913: Too many arguments (6/5) (too-many-arguments)

-----------------------------------
Your code has been rated at 7.68/10

(base) alejandrojuarez@192 Actividad 6.2 %
```

Corrección de errores de Pylint



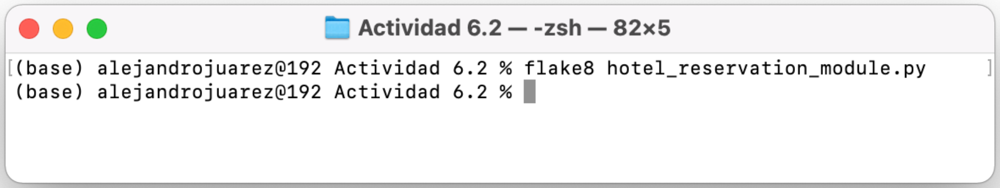
```
(base) alejandrojuarez@192 Actividad 6.2 % pylint hotel_reservation_module.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

(base) alejandrojuarez@192 Actividad 6.2 %
```

Chequeo de errores con Flake8



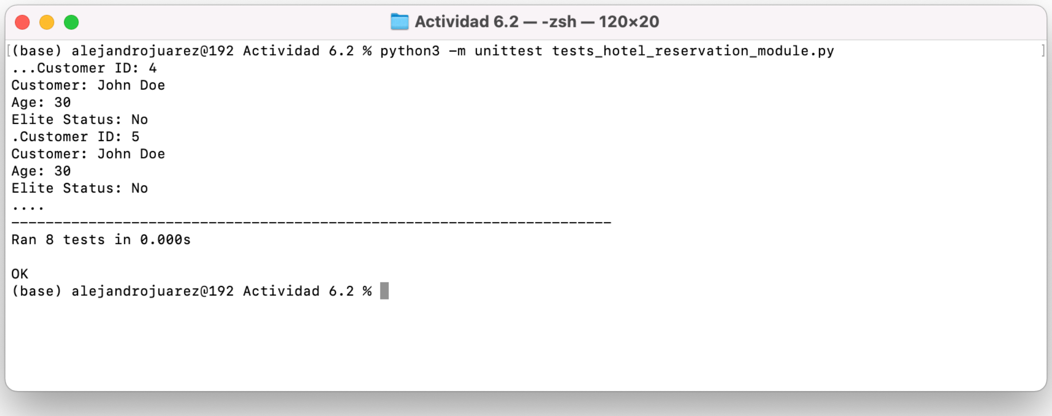
```
(base) alejandrojuarez@192 Actividad 6.2 % flake8 hotel_reservation_module.py
hotel_reservation_module.py:6:80: E501 line too long (120 > 79 characters)
hotel_reservation_module.py:12:80: E501 line too long (80 > 79 characters)
hotel_reservation_module.py:30:80: E501 line too long (101 > 79 characters)
hotel_reservation_module.py:36:80: E501 line too long (100 > 79 characters)
hotel_reservation_module.py:47:80: E501 line too long (113 > 79 characters)
hotel_reservation_module.py:53:80: E501 line too long (100 > 79 characters)
hotel_reservation_module.py:65:80: E501 line too long (90 > 79 characters)
hotel_reservation_module.py:84:80: E501 line too long (97 > 79 characters)
hotel_reservation_module.py:101:80: E501 line too long (87 > 79 characters)
hotel_reservation_module.py:128:80: E501 line too long (100 > 79 characters)
hotel_reservation_module.py:129:80: E501 line too long (100 > 79 characters)
hotel_reservation_module.py:131:80: E501 line too long (114 > 79 characters)
hotel_reservation_module.py:136:80: E501 line too long (84 > 79 characters)
hotel_reservation_module.py:137:80: E501 line too long (86 > 79 characters)
hotel_reservation_module.py:138:80: E501 line too long (86 > 79 characters)
hotel_reservation_module.py:149:80: E501 line too long (92 > 79 characters)
hotel_reservation_module.py:155:80: E501 line too long (80 > 79 characters)
hotel_reservation_module.py:160:80: E501 line too long (118 > 79 characters)
hotel_reservation_module.py:180:1: E305 expected 2 blank lines after class or function definition, found 1
hotel_reservation_module.py:191:80: E501 line too long (86 > 79 characters)
(base) alejandrojuarez@192 Actividad 6.2 %
```

Corrección de errores con Flake8



```
(base) alejandrojuarez@192 Actividad 6.2 % flake8 hotel_reservation_module.py
(base) alejandrojuarez@192 Actividad 6.2 %
```

Ejecución de Pruebas Unitarias


Actividad 6.2 — -zsh — 120×20

```
(base) alejandrojuarez@192 Actividad 6.2 % python3 -m unittest tests_hotel_reservation_module.py
...Customer ID: 4
Customer: John Doe
Age: 30
Elite Status: No
.Customer ID: 5
Customer: John Doe
Age: 30
Elite Status: No
....
----------------------------------------------------------------------
Ran 8 tests in 0.000s

OK
(base) alejandrojuarez@192 Actividad 6.2 %
```