SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT 162
Object Oriented Programming

Tutor-Marked Assignment

Jan 2023 Presentation

TUTOR-MARKED ASSIGNMENT (TMA)

This assignment is worth 24% of the final mark for ICT162, Object Oriented Programming.

The cut-off date for this assignment is **Wednesday, 26 April 2023, 2355 hrs.**

Note to Students:

Submit your solution document in the form of a single MS Word file. You are to include the following particulars in your submission: Course Code, Title of the TMA, SUSS PI No., Your Name, and Submission Date. Put this information in the first page of your solution document. Use the template word document provided – **SUSS_PI_No-FullName_162TMA.docx.** Rename the file with your SUSS_PI_No, FullName, and read the submission details under Module\TMA01 in the ICT162 L01**. Do NOT submit as a pdf document**.

You should make only one submission for TMA.

You are to copy and paste Python source code into your solution document as text. If you submit source code as image, no marks will be awarded to your program. Submit screenshots for **only** output of your program, where applicable.

Assignment Requirements:

- Unless specified in the question, you CANNOT use packages not covered in this module, e.g., re, collections, numpy, pandas etc.
- All classes must be documented. Provide sufficient comments to your code and ensure that your program adheres to good programming practices such as not using global variables.
- Failing to do so can incur a penalty of as much as 50% of the mark allotted.

Read the following introduction before attempting this TMA.

After more than three years of living with heightened health alerts through the national disease response system, Singapore started major easing of COVID 19 rules from Apr 26, 2022. Since then, the recovery of tourism is accelerating faster than expected, partly due to China's reopening as well as many countries' borders opening up.

As Singapore (along with New York City) have been named the most expensive cities in the world in 2022 by the Economist Intelligence Unit, the Tourism Board felt the need to market Singapore as an affordable destination to travellers.

One of the proposals or concepts, is to convert the Community Isolation Facilities (built during Covid-19 pandemic) into "minimalist" hotels, providing safe (sanitised daily) and affordable accommodation for budget travellers.

SAMI (Safe-Minimalist) Hotel, is the first of such proposed project and you are assigned to help develop an application to help enable booking of the hotel's room, with option to add amenities. A simplified "Check-In" should be included in your application, while check-out, blacklisting, housekeeping tasks will be manual for now.

To start, SAMI will rollout 2 types of room – Standard (2m x 2.1m) and Deluxe (2.3m x 2.1m). During the booking, guests will first select a room type, then they will decide on their preferred bed type – Single (1.9m x 0.91m) or Super (1.9m x 1.07m). By default, the bed includes a pillow and a blanket. SAMI will also offer the following type of amenities, as "option" for guests to add-on.

- General: Wi-Fi, Gym Pass, Pool Pass, Pillow, Blanket
- Toiletries: Shampoo, Shower Gel, Bath Towel, Bathrobe, Hand Towel
- Personal care: Shaving Cream, Razor, Shower Cap, Hair Dryer, Slippers
- Furniture: Walled-TV, Fridge, Bedside Table, Coffee Kit, Writing Desk, Chair

Price per day will be the sum of room price, bed price and the selected amenities' prices.

Question 1 (5 marks)

Let's start with the Guest. Construct the Guest class according to the specification in the class diagram shown below in Figure Q1.

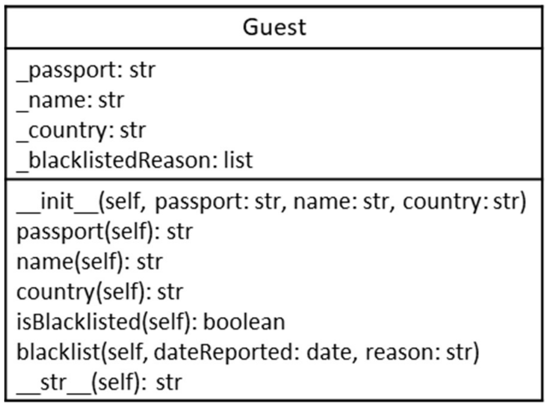


Figure Q1

Implement the class – Guest

The Guest class has:

- Four instance variables:
  - _ passport (str): the passport number of the guest.
  - _name (str): name of guest.
  - _country (str): guest's country of citizenship.
  - _blacklistedReason (list): like any other hotels, restaurants and pubs, SAMI also adopted the blacklisting strategy where misbehaving guests are prevented from (future) booking with SAMI. If guest is blacklisted, the list will contain pairings of date and reason for each offence.
- Constructor performs the following:
  - Initialises these 3 instance variables: _passport, _name and _country, with the given parameters.
- Getter methods for the instance variables: _ passport, _name and _country. Use the property decorator.
- The method isBlacklisted returns True if the _blacklistedReason list is not empty. Otherwise, the method returns False.
- As it is possible that a guest can be blacklisted more than once, the method blacklist will add the given parameters "dateReported" and "reason" as a list, into the _blacklistedReason (list of lists).
- The __str__ method returns a string representation of a Guest object, in the following order: passport number, name, country and blacklisting details (if any)

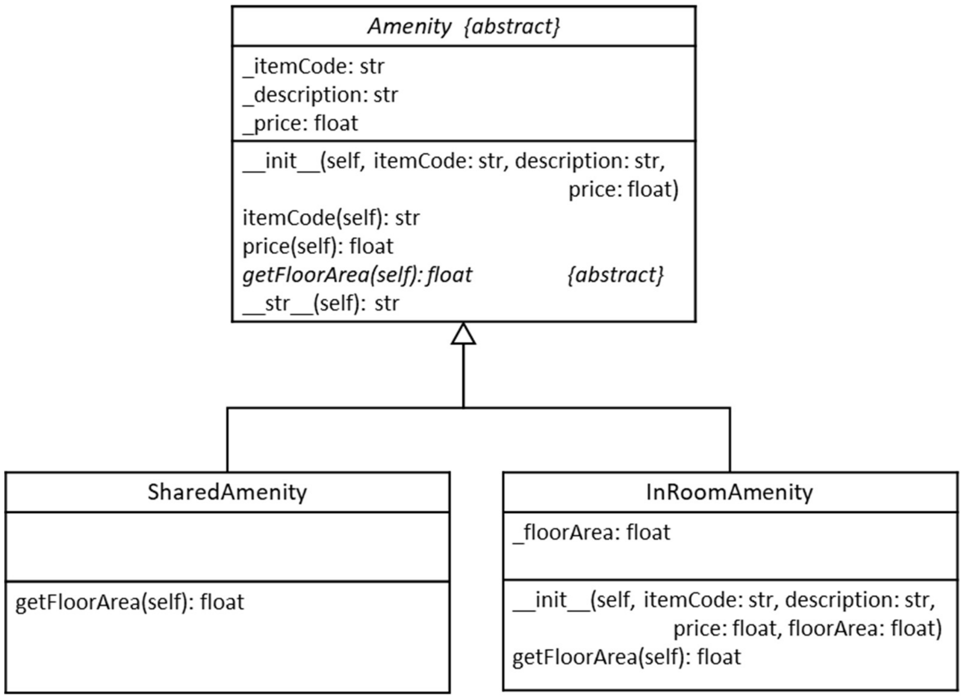
```
Passport Number: 123445678
Name: Kyrie Irving
Country: USA
<< Blacklisted on date, reason >>
> 12-Feb-2023   Drunk and disorderly
> 12-Feb-2023   Assault employees
```

(5 marks)

Question 2 (15 marks)

Demonstrate your understanding of how class hierarchy and association are used to organize information, and then construct the class hierarchy and association according to specification.

Study the class diagram in Figure Q2, which focuses on the Amenity class and its subclasses.



Amenity {abstract}
_itemCode: str
_description: str
_price: float
__init__(self, itemCode: str, description: str, price: float)
itemCode(self): str
price(self): float
getFloorArea(self): float {abstract}
__str__(self): str
SharedAmenity
getFloorArea(self): float
InRoomAmenity
_floorArea: float
__init__(self, itemCode: str, description: str, price: float, floorArea: float)
getFloorArea(self): float

Figure Q2

(a) Implement the abstract class – Amenity.

The Amenity class has:

- THREE instance variables:
  - _itemCode (str): unique code for this amenity item, e.g., “GYM_PEP”.
  - _description (str): description of the amenity, e.g., “Per entry pass to gym (Level 4-01)”.
  - _price (float): the price for this amenity, e.g., $1 for the one-day pass to the gym.
- Constructor initialises these 3 instance variables: _itemCode, _description and _price, using the given parameters.
- Getter methods for the instance variables: _itemCode and _price. Use the property decorator.
- One abstract method, getFloorArea which returns the floor area occupied by this amenity.
- The __str__ method returns a string representation of an Amenity object, in the following order – itemCode, description and price, separated by commas:

```
GYM-PEP, Per entry pass to gym (Level 4-01), $1.00
```

(5 marks)

(b) Write the Python class definitions for the SharedAmenity and InRoomAmenity.

The InRoomAmenity class is a subclass of Amenity, and it has:

- One additional instance variable, _floorArea (float) representing the floor area occupied by this amenity.
- Implement the method getFloorArea that returns the floor area occupied by this amenity.

The SharedAmenity class is also a subclass of Amenity.

- No additional instance variable.
- For shared amenities, the method getFloorArea returns 0, as these amenities are mostly in shared/common area, or not occupying any floor area of the room. E.g., gym, swimming pool, Wi-Fi etc.

(5 marks)

(c) Write a main() function to do the following:

(i) Assemble 4 Amenity objects into a list.
  - From Appendix A.1, select and create 2 SharedAmenity objects.
  - From Appendix A.2, select and create 2 InRoomAmenity objects with floor area > 0.1 m^2

(ii) Print the total floor area and total price of the Amenity objects created in Q2(c)(i).

(5 marks)

Question 3 (20 marks)

As illustrated in the introduction, SAMI follows the minimalist design style – simple, clean, and functional, by providing each room with an open floor plan, allowing guest to select their preferred bed type, and option to add amenities. In Figure Q3, the class diagram is updated with Room and Bed classes, to mimic the hotel room setup.

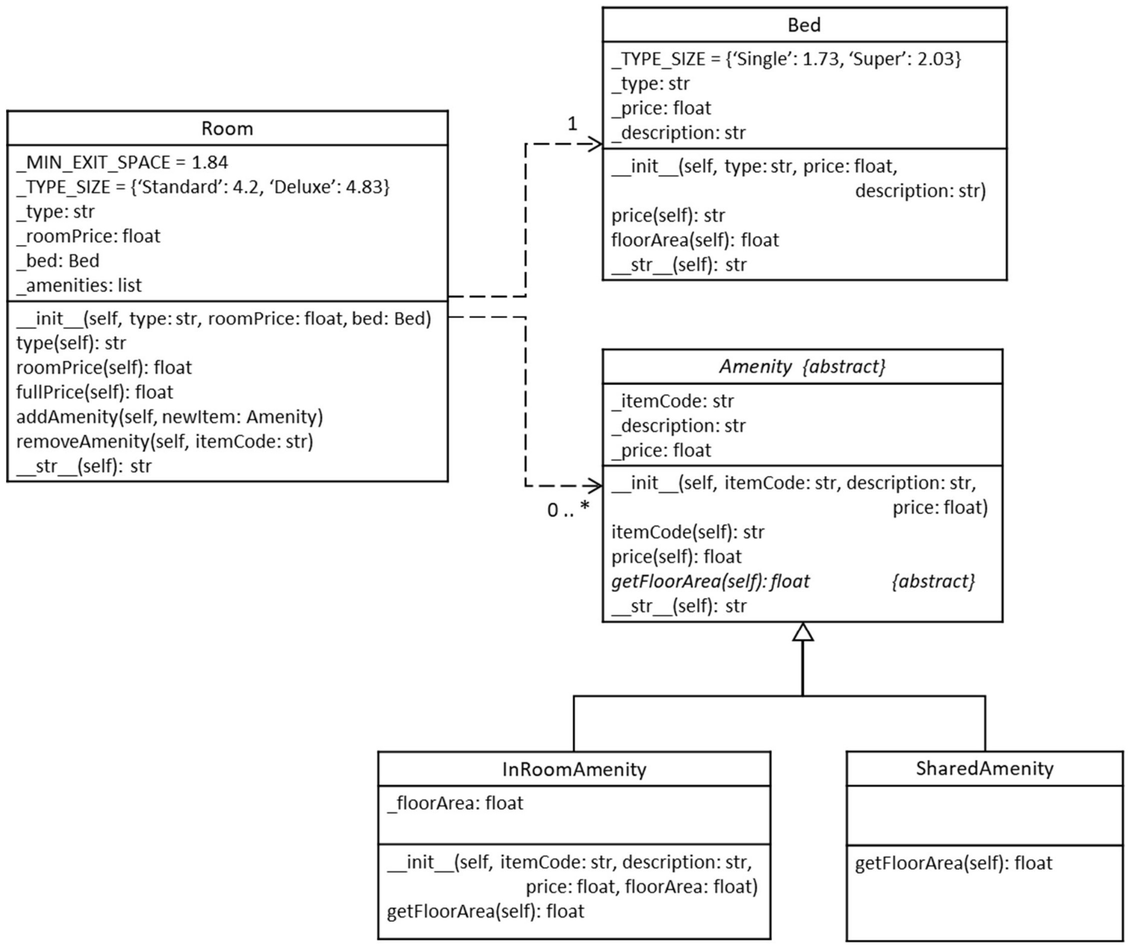

Bed
_TYPE_SIZE = {'Single': 1.73, 'Super': 2.03}
_type: str
_price: float
_description: str
__init__(self, type: str, price: float, description: str)
price(self): str
floorArea(self): float
__str__(self): str
Room
_MIN_EXIT_SPACE = 1.84
_TYPE_SIZE = {'Standard': 4.2, 'Deluxe': 4.83}
_type: str
_roomPrice: float
_bed: Bed
_amenities: list
__init__(self, type: str, roomPrice: float, bed: Bed)
type(self): str
roomPrice(self): float
fullPrice(self): float
addAmenity(self, newItem: Amenity)
removeAmenity(self, itemCode: str)
__str__(self): str
1
0 .. *
Amenity {abstract}
_itemCode: str
_description: str
_price: float
__init__(self, itemCode: str, description: str, price: float)
itemCode(self): str
price(self): float
getFloorArea(self): float {abstract}
__str__(self): str
InRoomAmenity
_floorArea: float
__init__(self, itemCode: str, description: str, price: float, floorArea: float)
getFloorArea(self): float
SharedAmenity
getFloorArea(self): float

Figure Q3

(a) Implement the Bed class.

The Bed class has:

- It has a class variable: _TYPE_SIZE that is a Dictionary containing the 2 bed types and their floor area, as key and value respectively.
- THREE instance variables:
  - _type (str): only "Single" or "Super" bed types currently offered by SAMI.
  - _price(float): the cost this bed type.
  - _description (str): description of the bed, e.g., "Super single with one pillow and one blanket".
- Constructor initialises the 3 instance variables using the given parameters.
- Getter method for instance variables: _price. Use the property decorator.
- Define getter method for floorArea. Use the property decorator.

  - floorArea is obtained through the class variable _TYPE_SIZE, using the instance variable _type.
- The __str__ method returns a string representation of a Bed object, in the following order – description and price, separated by commas:

```
Super single bed with one pillow and one blanket, $12.99
```

(4 marks)

(b) Create the following exception classes.

- BookingException – subclass of the Exception class. This class has no added attribute or method. When the application encounters booking rules violation, your application will raise an exception from this class.
- MinFloorAreaException – subclass of the Exception class. This class has no added attribute or method.
  - In Singapore, there is a requirement to maintain a minimum floor area and passageway to facilitate safe and swift exit in event of emergency.
  - When adding amenities to room, the minimum floor area rule may be violated. Your application will raise an exception from this class.

(1 mark)

(c) Construct and Implement the Room class.

The Room class has:

- TWO class variables:
  - _MIN_EXIT_SPACE: minimum floor area (1.84m^2) that should be kept clear.
  - _TYPE_SIZE that is a Dictionary containing the 2 room types and their floor area, as key and value respectively.
- FOUR instance variables:
  - _type (str): the room type, either “Standard” or “Deluxe”.
  - _roomPrice (float): representing the price for the room, excluding the bed.
  - _bed (Bed): the Bed object, which is the preferred bed of the guest.
  - _amenities (list): a collection (list) containing the “add-on” amenities.
- Constructor initialises these 3 instance variables: _type, _roomPrice and _bed using the given parameters. The constructor should set _amenities to an empty list.
- Getter method for instance variables: _type and _roomPrice. Use the property decorator.
- Define getter method for fullPrice – the sum of room price, bed price and all amenities’ prices. Use the property decorator
- The addAmenity method has newItem (Amenity object) as parameter. The following checks need to be performed before adding this newItem into _amenities:
  - The minimum floor area of 1.84m^2 must not be compromised after adding this newItem. If the above condition is not met, raise MinFloorAreaException with appropriate message.
  - In SAMI, there should not be duplicate amenity in each room, i.e., not more than one, raise BookingException with appropriate message if duplicate amenity is added.

- With itemCode as parameter, the removeAmenity method will find the matching Amenity object with the same item code and remove it from the collection _amenities.
    - If there is no amenity matching the given itemCode, then method should raise BookingException with message stating there is no such amenity in this room.
- The __str__ method returns the details of the room, follow by the string representations of bed, amenities, and the full price. Below are two examples:

```
Standard room, $16.99
Super single bed with one pillow and one blanket, $12.99
Full Price: $29.98


Deluxe room, $19.99
Single bed with one pillow and one blanket, $10.99
One hand towel (to return when check-out), $1.00
Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
Full Price: $40.97
```

(10 marks)

(d) Write a main() function with exception handling for the following:

(i) Create **TWO** rooms using the information below.

- Deluxe room with super single bed, and add the following amenities in this order:
    - Mini fridge
    - Chair
    - Writing desk
    - Iron and ironing Board

- Standard room with single bed, and add the following amenities in this order:
    - Gym pass
    - Mini fridge
    - Wi-Fi
    - Gym pass

Print the string representation of the created Room objects.

(ii) Remove these amenities from both rooms, in this order::

- Mini fridge
- Gym pass

Print the string representation of the Room objects, after removal.

(5 marks)

Question 4 (40 marks)

This question is a continuation of Question 1, 2 & 3, hence will make use of the classes in Question 1, 2 & 3.

To stay in SAMI Hotel, you will need to make a booking. As SAMI only offer rooms for individuals, a booking is made up of ONE guest and ONE room. However, a booking can only be submitted and confirmed if there is room available from the check-in to check-out dates.

As SAMI Hotel only offer 2 room types, the room availability is provided through a file and released monthly (see Appendix C). If selected room is available on those dates, a booking ID will be generated once the booking is submitted. On the check-in date, the guest will quote the booking ID when arriving at SAMI Hotel. Room allocation and the rest of hotel operations will be manual.

In this question, you will create a prototype application for the booking of rooms in SAMI Hotel. SAMI management will observe the demand and decide later if the application is to enhance to support the rest of Hotel operations.

Refer to the Booking class diagram in Figure Q4a and the Hotel class diagram in Figure Q4b.

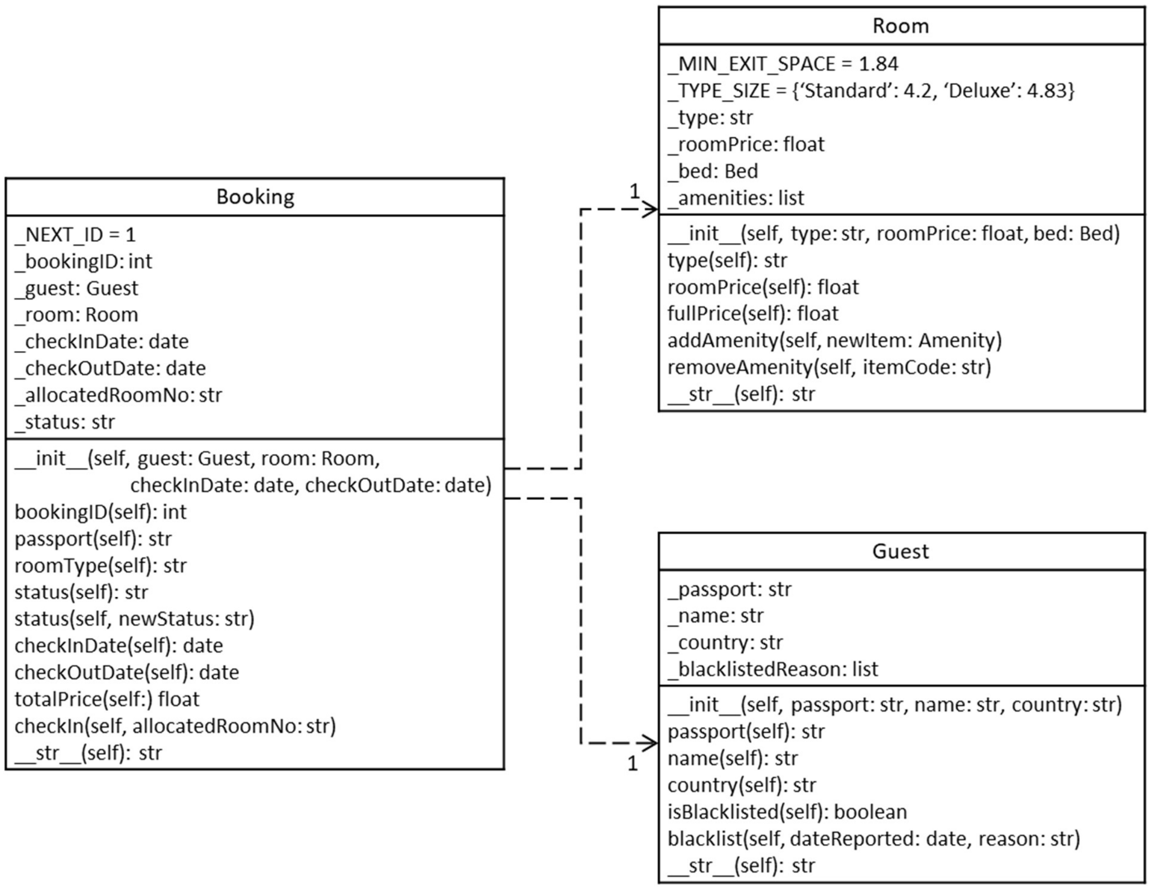


Figure Q4a

(a) Implement the Booking class.

- It has a class variable: _NEXT_ID, starting from 1, used for generating running numbers to uniquely identify the booking.
- There are SEVEN instance variables:
  - _bookingID (str): the booking ID to uniquely identify this booking
  - _guest (Guest): the Guest object, represent the guest who make this booking.
  - _room (Room): the Room object, representing the room, including the bed and amenities selected by the guest.
  - _checkInDate (date): check-in date.
  - _checkOutDate (date) : check-out date.
  - _allocatedRoomNo (str): this field is empty when booking is made, and will be updated by SAMI when the guest check-in.
  - _status (str): status of the booking.
- Constructor performs the following:
  - Initialises these 4 instance variables: _guest, _room, _checkInDate and _checkOutDate, using the given parameters.
  - The _bookingID is generated using the class variable _NEXT_ID, which starts from 1, and increment by 1 for every new booking.
  - Set _allocatedRoomNo to None, and _status to "Pending".
  - Ensure the guest is not backlisted, or check-out date is at least 1 day after check-in date. Otherwise, raise BookingException with appropriate message.
- Define the following getter methods, using the property decorator:
  - Instance variables: _bookingID, _checkInDate, _checkOutDate and _status.
  - For the following:
    - passport – returns the passport of the guest.
    - roomType – returns the room's type.
    - totalPrice – returns the product of fullPrice of room and number of nights.
- Setter methods for instance variable: _status. Use the property decorator.
- Method checkIn will first check if the following rules are not violated:
  - Current booking status must be "Confirmed".
  - The check-in must be done on the _checkInDate itself.
  - Check that the guest is not blacklisted.
  - If any of the above condition is not met, raise BookingException with appropriate message.

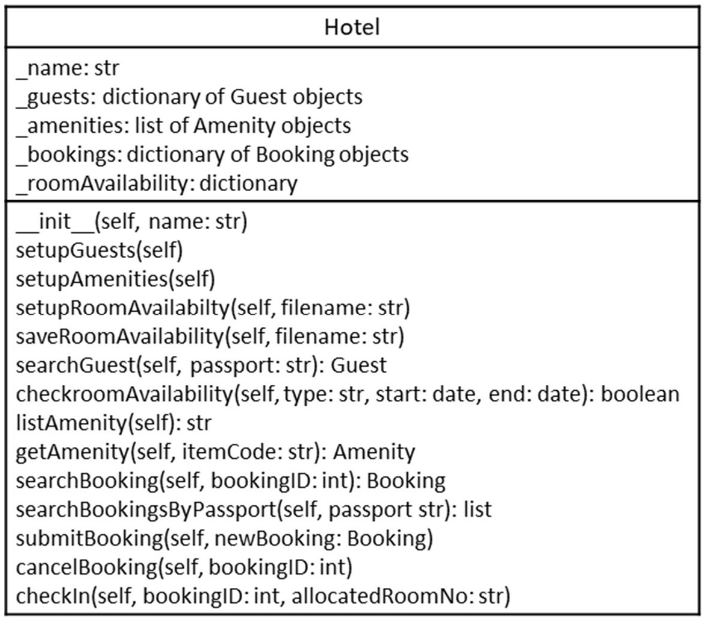
  The method then proceed to perform the following:
  - Update _status to "Checked-In".
  - Update _allocatedRoomNo with the given parameter.
- The __str__ method returns a string representation of a Booking object, which consists of booking ID, guest's passport and name, check-in, checkout dates, booking status, follow by the string representation of Room object, and finally the total price. Below is a sample:

```
Booking ID: 999
Passport Number: 123445678
Name: Kyrie Irving
Check-In/Out dates: 12-Feb-2023 / 15-Feb-2023
Booking Status: Confirmed

Deluxe room, $19.99
Single bed with one pillow and one blanket, $10.99
```

```
One hand towel (to return when check-out), $1.00
Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
Full Price: $40.97

Total Price: $40.97 x 3 nights = $122.91
```

(8 marks)

(b) Figure Q4b highlight the class diagram for Hotel class.



```
Hotel
-----------------------------------------------
_name: str
_guests: dictionary of Guest objects
_amenities: list of Amenity objects
_bookings: dictionary of Booking objects
_roomAvailability: dictionary
-----------------------------------------------
__init__(self, name: str)
setupGuests(self)
setupAmenities(self)
setupRoomAvailabilty(self, filename: str)
saveRoomAvailability(self, filename: str)
searchGuest(self, passport: str): Guest
checkroomAvailability(self, type: str, start: date, end: date): boolean
listAmenity(self): str
getAmenity(self, itemCode: str): Amenity
searchBooking(self, bookingID: int): Booking
searchBookingsByPassport(self, passport str): list
submitBooking(self, newBooking: Booking)
cancelBooking(self, bookingID: int)
checkIn(self, bookingID: int, allocatedRoomNo: str)
```

Figure Q4b

You are to complete a partially written Hotel class, given in Appendix E. The Python codes for Hotel class only have constructor and setup methods for guests, amenities, and room availability.

Description of the instance variables and constructor of the Hotel class:

- It has FIVE instance variables:
  - _name (str): the name of the hotel.
  - _guests (dictionary): collection of guests. The key for this dictionary is the passport number, and the value is the Guest object.
  - _amenities (list): the list of amenities that SAMI is providing.
  - _bookings (dictionary): collection of bookings. The key for this dictionary is the booking ID, and the value is the Booking object.
  - _roomAvailability (dictionary): the room availability as shown in Appendix C. The key for this dictionary is the date, and the value is a list holding the standard room count and deluxe room count.
- Given name and roomFilename as parameters, the constructor performs the following:
  - Initialises the instance variable: _name, using the given parameter "name".
  - Invoke the setupGuest method – to setup Guest objects into the dictionary by reading the information from "Guests.txt" and "Blacklist.txt".

  - Invoke the setupAmenities method – to setup the current set of Amenity objects into a list using the information provided in Appendix A.1 & A.2.
  - Invoke the saveRoomAvailability method, with the roomFilename, to setup the room availability as shown in Appendix C.
  - Setup _bookings as empty dictionary.

You are to write the following methods of the Hotel class:

- Given a passport (number) as parameter, the method searchGuest searches and returns the Guest object matching this passport number. If not found, the method returns None.
- checkRoomAvailability method will return True if there is room (type) available from parameter start (date) to parameter end (date). The method returns False, if:
  - start date is later than end date.
  - start or end dates are not in the _roomAvailability dictionary.
- listAmenity method returns the string representation of all Amenity objects, concatenated and one Amenity object per line.
- Given an itemCode as parameter, the getAmenity method searches and returns the Amenity object that matches the parameter itemCode. If there is no such amenity, the method returns None.
- searchBooking method has 1 parameter: bookingID (int). This method searches and returns the Booking object that matches the parameter bookingID. If there is no such booking, the method returns None.
- Given passport (str) as parameter, the searchBookingByPassport method searches and returns a list containing all the Booking objects with guest's passport matching the parameter passport. An empty list is returned if there is no booking found.
- The submitBooking method also has a newBooking (Booking object) as parameter. This method is invoked when the booking is ready for submission. The following validation tasks need to be performed:
  - Check that the newBooking status is "Pending".
  - Check that there should be room available for this booking (from check-in to check-out date, for the room type).
  - If any of the above checks fail, this method should raise BookingException with appropriate messages.

  If there is no exception, then proceed to:
  - Update room availability by deducting the room type count from check-in to check-out dates.
  - Update the booking status to "Confirmed".
  - Add this newBooking object into _bookings dictionary.
- cancelBooking method has bookingID as parameter and need to perform the following checks:
  - If there is no booking with this booking ID, then method should raise BookingException with message saying there is no such booking.
  - If this booking ID can locate a booking, but the booking status is not "Confirmed", then method should raise BookingException with message stating this booking cannot be cancelled.

  - Update the booking status to "Cancelled" and update the room availability data (increment room type count by 1 from check-in to check-out dates)
- The checkIn method has 2 parameters: bookingID and allocatedRoomNo. This method will perform the following:
  - Search for this booking using the given bookingID.
  - If booking is found, invoke the checkIn method of the Booking object.
  - If no such booking, then method should raise BookingException with proper message.

(18 marks)

(c) Write an application that allow SAMI to manage the room booking processes, focusing on the creation, submission, and cancellation of bookings. As this is an experimental project, the actual room allocation and check-in procedure will be done manually. Hence, your application will provide a shortened "Check-In" function to just update the status and room allocated for the bookings.

```
@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option:
```

Before presenting the below above menu, your application needs to do the following:
  - Create the Hotel object for SAMI
  - Prompt user to enter filename containing room availability information.
  - Invoke the appropriate Hotel method to perform the room availability setup.

The same filename will be used to save the latest room availability information before the application exits.

The next few pages will explain each menu option in detail.

Option 1: Submit Booking

To submit a "booking", the following steps and data elements should be collected:
- Prompt for guest passport to help locate the Guest object.
  - If guest is registered in SAMI, print the details of the guest for verification.
  - If there is no such guest, display appropriate message and return to menu.
- Prompt user to enter the room and bed type. Create the Room object with the selected Bed object.
- List the hotel's amenities for guest to include in the booking:
  - Prompt the guest for the item code to add into the room.
  - If the item code is valid, add the Amenity object to the room.
  - Repeat until <enter> is pressed to indicate completion.
- Ask for the Check-In and Check-Out date.

- Create the Booking object using data/objects collected above.
- Display the booking details for verification and prompt the user for confirmation to submit the booking.
  - If yes, proceed to submit this new booking.
  - If no, display message that this booking is aborted.
- Below is an illustration of successfully submitted booking:

```
@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 1
Enter Passport Number to start: a9001022
Guest located. Please verify

Passport: A9001022
Name: Steve Tan
Country: Malaysia

Select preferred room type (S)tandard or (D)eluxe): S
Select preferred bed type (S)ingle or s(U)per: S

List of Amenities
=================
WI-FI, One-day Wi-Fi access, $1.00
GYM-PEP, Per entry pass to gym (Level 4-01), $1.00
SWIM-PEP, Per entry pass to swimming pool (Level 2 outdoor), $1.50
BIZ-PEP, Per entry pass to business centre (Level 3-03), $2.00
BREAKFAST, Breakfast buffet at Sun café (Level 1-01, 6am to 10am), $8.99
Hi-TEA, Hi-Tea buffet at Sun café (Level 1-01 2pm to 4pm), $11.99
SHAMPOO-1, One sachet of conditioning shampoo, $0.29
SHOWER-1, One sachet of shower gel, $0.25
TOWEL-B, One bath towel (to return when check-out), $1.50
TOWEL-H, One hand towel (to return when check-out), $1.00
SHOW-CAP, One shower cap, $0.49
SHA-RAZOR, One disposable shaving razor (men’s), $2.99
SHA-CREAM, One sachet of shaving cream (men’s), $0.29
HAIR-DRY, Hair Dryer (to return when check-out), $2.00
SLIPPERS, One pair of disposable slippers , $4.59
WALLED-TV, Walled-TV, $3.99
FRIDGE, Mini Fridge (50L), $4.59
TABLE-B, Bedside table (60cm x 50cm), $2.59
DESK-W, Writing desk (80cm x 55cm), $3.99
CHAIR, Foldable Chair (42cm x 38cm), $2.59
IRON-B, Iron and ironing board (128cm x 30cm), $2.99

Enter item code to add or <enter> to stop: wi-fi
WI-FI added...

List of Amenities
=================
WI-FI, One-day Wi-Fi access, $1.00
GYM-PEP, Per entry pass to gym (Level 4-01), $1.00
SWIM-PEP, Per entry pass to swimming pool (Level 2 outdoor), $1.50
BIZ-PEP, Per entry pass to business centre (Level 3-03), $2.00
BREAKFAST, Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
```

```
.
.
.
WALLED-TV, Walled-TV, $3.99
FRIDGE, Mini Fridge (50L), $4.59
TABLE-B, Bedside table (60cm x 50cm), $2.59
DESK-W, Writing desk (80cm x 55cm), $3.99
CHAIR, Foldable Chair (42cm x 38cm), $2.59
IRON-B, Iron and ironing board (128cm x 30cm), $2.99

Enter item code to add or <enter> to stop: <enter>

Enter Check-In date in DD-MON-YYYY: 10-Apr-2023
Enter Check-Out date in DD-MON-YYYY: 12-apr-2023

Please verify Guest and Room details....

Booking ID: 6
Passport: A9001022
Name: Steve Tan
Check-In/Out dates: 10-Apr-2023 / 12-Apr-2023
Booking status: Pending

Standard room, $16.99
Single bed with one pillow and one blanket, $10.99
WI-FI, One-day Wi-Fi access, $1.00
Full Price: $28.98

Total price: $28.98 x 2 nights = $57.96

Proceed to submit booking? (Y/N): y
Booking is submitted.
```

- Below are examples of unsuccessful submissions:

```
@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 2
Enter Passport Number to start: M2375489
Guest located. Please verify

Passport: M2375489
Name: Mary Tan
Country: Malaysia

Select preferred room type (S)tandard or (D)eluxe): S
Select preferred bed type (S)ingle or s(U)per: S

List of Amenities
=================
WI-FI, One-day Wi-Fi access, $1.00
GYM-PEP, Per entry pass to gym (Level 4-01), $1.00
SWIM-PEP, Per entry pass to swimming pool (Level 2 outdoor), $1.50
BIZ-PEP, Per entry pass to business centre (Level 3-03), $2.00
BREAKFAST, Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
.
```

```
.
.
WALLED-TV, Walled-TV, $3.99
FRIDGE, Mini Fridge (50L), $4.59
TABLE-B, Bedside table (60cm x 50cm), $2.59
DESK-W, Writing desk (80cm x 55cm), $3.99
CHAIR, Foldable Chair (42cm x 38cm), $2.59
IRON-B, Iron and ironing board (128cm x 30cm), $2.99

Enter item code to add or <enter> to stop: breakfast
BREAKFAST added...
.
.
.
<< and many more amenities added… >>
.
.
.
Enter item code to add or <enter> to stop: <enter>

Enter Check-In date in DD-MON-YYYY: 11-Apr-2023
Enter Check-Out date in DD-MON-YYYY: 12-apr-2023

Please verify Guest and Room details....

Booking ID: 7
Passport: M2375489
Name: Mary Tan
Check-In/Out dates: 11-Apr-2023 / 12-Apr-2023
Booking status: Pending

Deluxe room, $19.99
Single bed with one pillow and one blanket, $10.99
BREAKFAST, Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
FRIDGE, Mini Fridge (50L), $4.59
TOWEL-B, One bath towel (to return when check-out), $1.50
Full Price: $46.06

Total price: $46.06 x 1 night = $46.06

Confirm submit booking? (Y/N): y
There is no Deluxe room available from 11-Apr-2023 to 12-Apr-2023
```

Option 2: Cancel Booking

To cancel a “booking”,

- Prompt for the booking ID.
- Invoke the appropriate Hotel method to perform the booking cancellation.
- Below are examples of successful and unsuccessful cancellations:

```
@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 2
Enter booking ID to cancel: 2
Booking is cancelled...

@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 2
Enter booking ID to cancel: 2
The booking status is 'Cancelled'. Cannot cancelled!!

@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 2
Enter booking ID to cancel: 7777
There is no such booking...
```

<u>Option 3: Search Booking(s)</u>

The are 2 ways to search for booking(s), using booking ID or passport. Below are illustrations of search booking(s):

```
@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 3
Search by Booking ID or Passport? (B/P): b
Enter Booking ID to search: 7777
No booking found


@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 3
Search by Booking ID or Passport? (B/P): b
Enter Booking ID to search: 3

Booking ID: 3
Passport: M2375489
Name: Mary Tan
Check-In/Out dates: 12-Apr-2023 / 14-Apr-2023
Booking status: Confirmed
```

```
Deluxe room, $19.99
Single bed with one pillow and one blanket, $10.99
BREAKFAST, Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
FRIDGE, Mini Fridge (50L), $4.59
TOWEL-B, One bath towel (to return when check-out), $1.50
Full Price: $46.06

Total price: $46.06 x 2 nights = $92.12


@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 3
Search by Booking ID or Passport? (B/P): p
Enter Passport Number to search: a9001022

Booking ID: 14
Passport: A9001022
Name: Steve Tan
Check-In/Out dates: 22-Apr-2023 / 24-Apr-2023
Booking status: Confirmed

Deluxe room, $19.99
Super Single bed with one pillow and one blanket, $12.99
HAIR-DRY, Hair Dryer (to return when check-out), $2.00
TABLE-B, Bedside table (60cm x 50cm), $2.59
TOWEL-H, One hand towel (to return when check-out), $1.00
Full Price: $38.57

Total price: $38.57 x 2 nights = $77.14

Booking ID: 15
Passport: A9001022
Name: Steve Tan
Check-In/Out dates: 3-Apr-2023 / 4-Apr-2023
Booking status: Confirmed

Deluxe room, $19.99
Single bed with one pillow and one blanket, $10.99
BREAKFAST, Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
FRIDGE, Mini Fridge (50L), $4.59
TOWEL-B, One bath towel (to return when check-out), $1.50
Full Price: $46.06

Total price: $46.06 x 1 night = $46.06
```

Option 4: Check-In

To perform a "Check-In",

- Prompt for the booking ID.
- Locate the Booking object with the provided booking ID.
    - If there is no such booking, display appropriate message and return to menu.

- Display the booking details for verification.
- Prompt for the allocated room number, or <enter> to cancel the “Check-In”.
- If room number is entered, invoke the appropriate Hotel method to perform the “Check-In”.
- Below is an illustration of successful “Check-In”:

```
<assuming today is 12-Apr-2023>

   @@@@ SAMI Hotel @@@@
   ====================
   1. Submit Booking
   2. Cancel Booking
   3. Search Booking(s)
   4. Check-In
   0. Exit
   Enter option: 4
   Enter Booking ID to start Check-In: 1

   Please verify Guest and Room details....

   Booking ID: 1
   Passport: E234567Z
   Name: Lee Ah Seng
   Check-In/Out dates: 12-Apr-2023 / 15-Apr-2023
   Booking status: Confirmed

   Standard room, $16.99
   Single bed with one pillow and one blanket, $10.99
   WI-FI, One-day Wi-Fi access, $1.00
   BREAKFAST, Breakfast buffet at Sun café (Level 1-01 6am to 10am), $8.99
   FRIDGE, Mini Fridge (50L), $4.59
   Full Price: $42.56

   Total price: $42.56 x 3 nights = $127.68

   Enter allocated room number or <enter> to cancel Check-In: 1133
   Checked-In. Enjoy your stay in SAMI.
```

- Below are illustrations of unsuccessful “Check-In”:

```
   @@@@ SAMI Hotel @@@@
   ====================
   1. Submit Booking
   2. Cancel Booking
   3. Search Booking(s)
   4. Check-In
   0. Exit

   Enter option: 4
   Enter Booking ID to start Check-In: 2

   Please verify Guest and Room details....

   Booking ID: 2
   Passport: C1899345
   Name: Lau Hong Gui
   Check-In/Out dates: 13-Apr-2023 / 15-Apr-2023
   Booking status: Confirmed
```

```
Standard room, $16.99
Super Single bed with one pillow and one blanket, $12.99
GYM-PEP, Per entry pass to gym (Level 4-01), $1.00
HAIR-DRY, Hair Dryer (to return when check-out), $2.00
TABLE-B, Bedside table (60cm x 50cm), $2.59
Full Price: $35.57

Total price: $35.57 x 2 nights = $71.14

Enter allocated room number or <enter> to cancel Check-In: <enter>
Check-In aborted


@@@@ SAMI Hotel @@@@
====================
1. Submit Booking
2. Cancel Booking
3. Search Booking(s)
4. Check-In
0. Exit
Enter option: 4
Enter Booking ID to start Check-In: 2

Please verify Guest and Room details....

Booking ID: 2
Passport: C1899345
Name: Lau Hong Gui
Check-In/Out dates: 13-Apr-2023 / 15-Apr-2023
Booking status: Confirmed

Standard room, $16.99
Super Single bed with one pillow and one blanket, $12.99
GYM-PEP, Per entry pass to gym (Level 4-01), $1.00
HAIR-DRY, Hair Dryer (to return when check-out), $2.00
TABLE-B, Bedside table (60cm x 50cm), $2.59
Full Price: $35.57

Total price: $35.57 x 2 nights = $71.14

Enter allocated room number or <enter> to cancel Check-In: 1122
Cannot check-in today!!
```

Errors and exceptions handling

- Assume user will enter valid input. However, you are encouraged to handle type/value error.
- For all menu options, handle any raised exceptions by printing the error messages.

(14 marks)

Question 5 (20 marks)

Upon testing of your application developed in Q4, SAMI is requesting for a separate application to manage their blacklisting process. After understanding SAMI requirements, you have decided to showcase your Python tkinter knowledge and develop a GUI application to help SAMI record the date of incident and reason for blacklisting the guest.

(a) Develop and implement the SAMI Blacklisting GUI as shown in Figure Q5a.

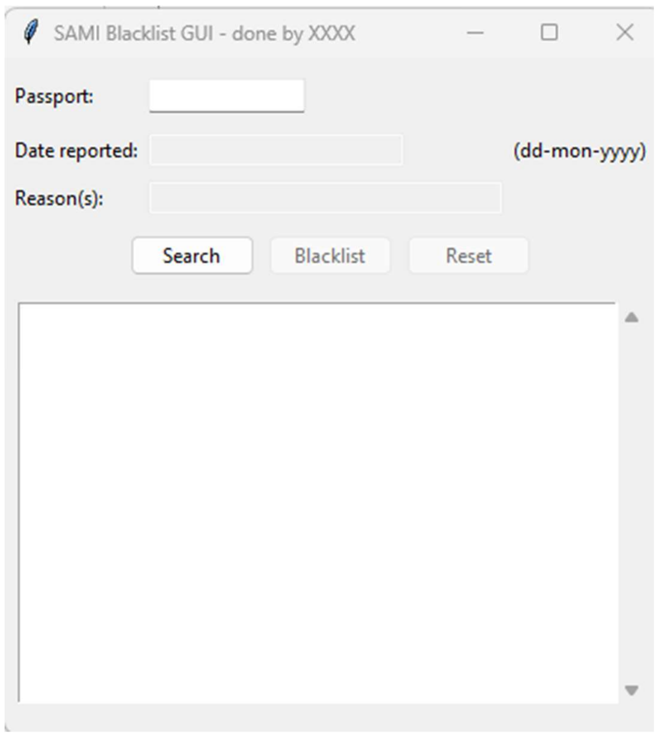


Figure Q5a

- Title of the GUI **MUST** include your name (replace XXXX with your name).
- The layout is recommended, but you can define any suitable design.
- These are the proposed widgets:
  - Label "Passport:" to guide user input (Entry) to begin guest search with Passport Number.
  - Label "Date Reported:" & "(dd-mon-yyyy)" to guide user input (Entry) for the date of incident.
  - Label "Reason(s):" to guide user input (Entry) to record reason(s) for blacklisting this guest.
  - 3 Buttons "Search" (enabled), "Blacklist" (disabled) and "Reset" (disabled).
  - Scrolled Text (disabled) to display search results, blacklisting outcome, or any messages.

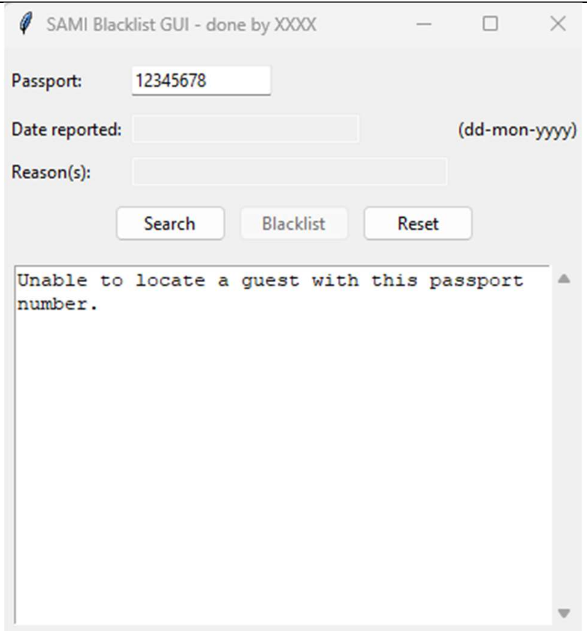
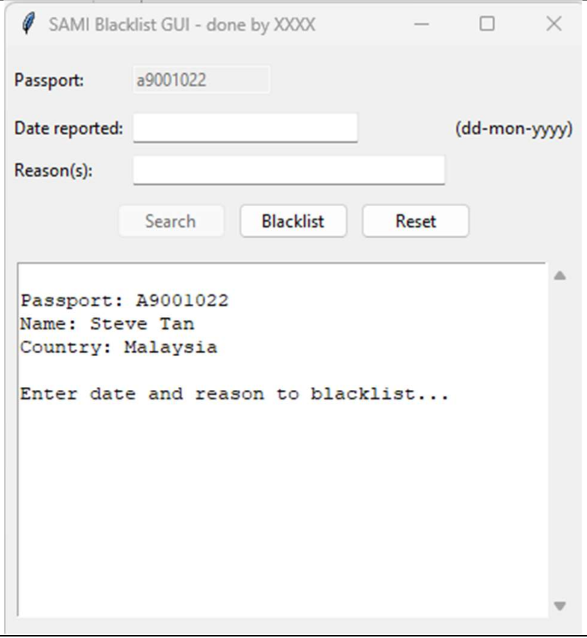
In this prototype GUI, re-instating blacklisted guests or removal of incidents are not in scope.

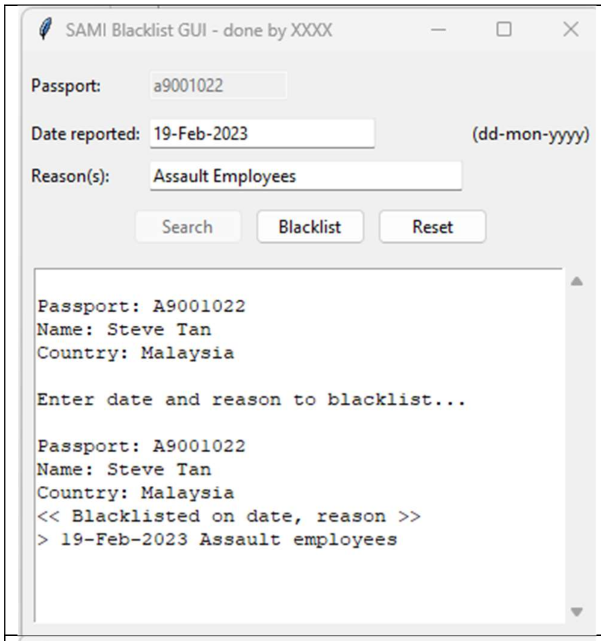
(7 marks)

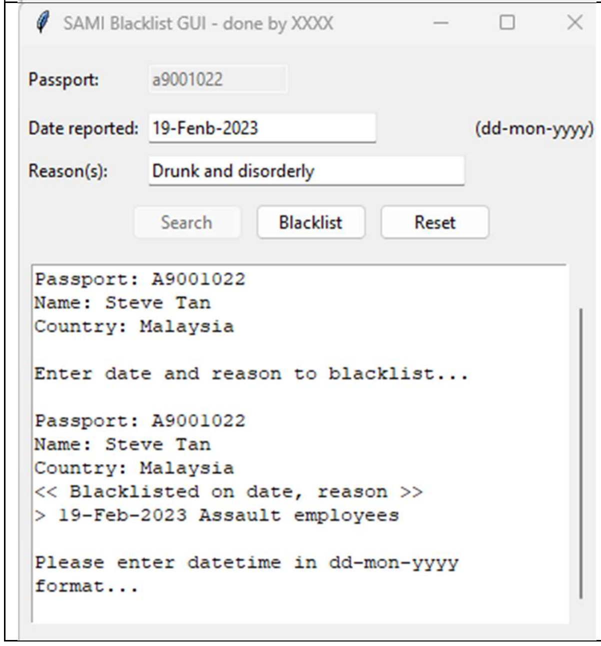
(b) Implement event handling so that the SAMI Blacklisting GUI can respond to left button clicks on these buttons.

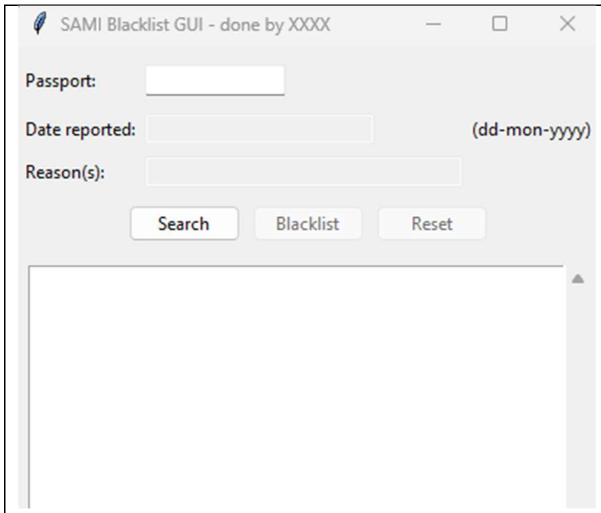
Before the GUI is presented to the user, the program must load the guests from Appendix D.1 (Guests.txt), follow by the date of incidents and blacklisted reasons for guests (Appendix D.2 Blacklist.txt). It is not mandatory to integrate with Hotel class of question 4.

Below are demonstrations of the SAMI Blacklisting GUI application:

 SAMI Blacklist GUI - done by XXXX Passport: 12345678 Date reported: (dd-mon-yyyy) Reason(s): Search Blacklist Reset Unable to locate a guest with this passport number.	To start using the GUI, user need enter the passport number to search for the guest. If the passport number cannot find a guest, your program should display appropriate error messages. Ensure “Reset” button is enabled whenever the ScrolledText is populated with any text.
 SAMI Blacklist GUI - done by XXXX Passport: a9001022 Date reported: (dd-mon-yyyy) Reason(s): Search Blacklist Reset Passport: A9001022 Name: Steve Tan Country: Malaysia Enter date and reason to blacklist...	When the passport number located a guest, information for this guest should be shown in the ScrolledText, and your program should perform the following: - enable text entries for “Date reported” & “Reason(s)” - disable passport entry field - disable “Search” button Ensure “Reset” button is enabled whenever the ScrolledText is populated with any text. At this point, the guest is identified, and user is ready to use your GUI application to blacklist this guest.

 SAMI Blacklist GUI - done by XXXX
Passport: a9001022
Date reported: 19-Feb-2023 (dd-mon-yyyy)
Reason(s): Assault Employees
```
Passport: A9001022
Name: Steve Tan
Country: Malaysia

Enter date and reason to blacklist...

Passport: A9001022
Name: Steve Tan
Country: Malaysia
<< Blacklisted on date, reason >>
> 19-Feb-2023 Assault employees
```
	When user entered valid date with reason and clicked “Blacklist”, your program should:
	- invoke the correct method of the Guest object to blacklist this guest
	- display the “updated” guest information
	- save the guest’s passport number, date of incident, blacklisted reason into “Blacklist.txt”

 SAMI Blacklist GUI - done by XXXX
Passport: a9001022
Date reported: 19-Fenb-2023 (dd-mon-yyyy)
Reason(s): Drunk and disorderly
```
Passport: A9001022
Name: Steve Tan
Country: Malaysia

Enter date and reason to blacklist...

Passport: A9001022
Name: Steve Tan
Country: Malaysia
<< Blacklisted on date, reason >>
> 19-Feb-2023 Assault employees

Please enter datetime in dd-mon-yyyy
format...
```
	Your program should display appropriate error messages in the ScrolledText when:
	- “Date reported” is empty or in wrong date format
	- “Reason(s)” is empty or no data value

 SAMI Blacklist GUI - done by XXXX Passport: Date reported: (dd-mon-yyyy) Reason(s): Search Blacklist Reset	When “Reset” button is clicked, your program should: - clear the 3 text entries - clear the ScrolledText - enable the “Search” button - disable the “Blacklist” and “Reset” button - set focus on the “Passport” text entry field

Apply Object oriented principles when writing the GUI program. The program must be written as a class, and you can submit your codes for Q5(a) together with Q5(b).
Submit at least 4 set of screenshots: displaying successful/unsuccessful search and blacklisting results.

(13 marks)

Appendix A.1 – SharedAmenity,txt

This file contains the shared amenities provided by SAMI.

<u>Format of the file</u>:

Each line of the file contains information in this order:

```
<itemCode>,<description>,<price>

WI-FI,One-day Wi-Fi access,1.00
GYM-PEP,Per entry pass to gym (Level 4-01),1.00
SWIM-PEP,Per entry pass to swimming pool (Level 2 outdoor),1.50
BIZ-PEP,Per entry pass to business centre (Level 3-03),2.00
BREAKFAST,Breakfast buffet at Sun café (Level 1-01 6am to 10am),8.99
HI-TEA,Hi-Tea buffet at Sun café (Level 1-01 2pm to 4pm),11.99
```

Appendix A.2 – InRoomAmenity.txt

This file contains the In-Room amenities provided by SAMI.

<u>Format of the file</u>:

Each line of the file contains information in this order:

```
<itemCode>,<description>,<price>,<floorArea>

SHAMPOO-1,One sachet of conditioning shampoo,0.29,0
SHOWER-1,One sachet of shower gel,0.25,0
TOWEL-B,One bath towel (to return when check-out),1.50,0
TOWEL-H,One hand towel (to return when check-out),1.00,0
SHOW-CAP,One shower cap,0.49,0
SHA-RAZOR,One disposable shaving razor (men’s),2.99,0
SHA-CREAM,One sachet of shaving cream (men’s),0.29,0
HAIR-DRY,Hair Dryer (to return when check-out),2.00,0.1
SLIPPERS,One pair of disposable slippers,4.59,0.1
WALLED-TV,Walled-TV,3.99,0
FRIDGE,Mini Fridge (50L),4.59,0.25
TABLE-B,Bedside table (60cm x 50cm),2.59,0.3
DESK-W,Writing desk (80cm x 55cm),3.99,0.44
CHAIR,Foldable Chair (42cm x 38cm),2.59,0.16
IRON-B,Iron and ironing board (128cm x 30cm),2.99,0.4
```

Appendix B.1

The following are the 2 room types provided by SAMI. *(floor area in m^2)*

Type	Description	Price	Floor Area
Standard	Standard room (2m x 2.1m)	$16.99	4.2
Deluxe	Deluxe room (2.3m x 2.1m)	$19.99	4.83

Appendix B.2
The following are the 2 bed types provided by SAMI. *(floor area in m^2)*

Type	Description	Price	Floor Area
Single	Single bed with one pillow and one blanket	$10.99	1.73
Super	Super single bed with one pillow and one blanket	$12.99	2.03

Appendix C – Rooms_April2023.txt
This file represents the room availability for month of April, 2023.

<u>Format of the file:</u>

Each line of the file contains information in this order:

```
<dd-mon-yyyy>,<standard rooms available>,<deluxe rooms available>

01-Apr-2023,6,4
02-Apr-2023,6,4
03-Apr-2023,1,1
04-Apr-2023,6,4
05-Apr-2023,1,1
06-Apr-2023,6,4
07-Apr-2023,6,4
08-Apr-2023,1,1
09-Apr-2023,6,4
10-Apr-2023,6,4
11-Apr-2023,1,1
12-Apr-2023,6,4
13-Apr-2023,6,4
14-Apr-2023,6,4
15-Apr-2023,6,4
16-Apr-2023,1,1
17-Apr-2023,6,4
18-Apr-2023,6,4
19-Apr-2023,6,4
20-Apr-2023,1,1
21-Apr-2023,6,4
22-Apr-2023,6,4
23-Apr-2023,6,4
24-Apr-2023,1,1
25-Apr-2023,6,4
26-Apr-2023,1,1
27-Apr-2023,6,4
28-Apr-2023,6,4
29-Apr-2023,6,4
30-Apr-2023,6,4
```

Note: the number of rooms for standard and deluxe are reduced in selected dates to help easier simulation of room unavailability during your testing.

Appendix D.1 – Guests.txt

Format of the file:

Each line of the file contains information in this order:

```
<passport>,<name>,<country>

123445678, Kyrie Irving, USA
E234567Z, Lee Ah Seng, Singapore
A9001022, Steve Tan, Malaysia
C1899345, Lau Hong Gui, Malaysia
PA0010023, Andrew Hewitt, Australia
M2375489, Mary Tan, Malaysia
```

Appendix D.2 – Blacklist.txt

Format of the file:

Each line of the file contains information in this order:

```
<passport>,<date of incident>,<blacklisted reason>

123445678, 12-Feb-2023, Drunk and disorderly
123445678, 12-Feb-2023, Assault employees
PA0010023, 12-Mar-2023, Disputing credit card charges
PA0010023, 12-Mar-2023, Causes damage to facilities
```

Appendix E – class Hotel

Python codes for constructor, including the setup of guests, amenities, and room availability.

```python
class Hotel:

    def __init__(self, name, roomFilename):
        self._name = name
        self._guests = self.setupGuests()
        self._amenities = self.setupAmenities()
        self._roomAvailability = self.setupRoomAvailability(roomFilename)
        self._bookings = {}

    def setupGuests(self):
        guests = { }
        infile = open("Guests.txt", "r")
        for line in infile:
            pp,name,country = line.split(",")
            guests[pp.strip()] = Guest(pp.strip(),name.strip(),country.strip())
        infile.close()

        infile = open("Blacklist.txt", "r")
        for line in infile:
            pp, dateReported, reason = line.split(",")
            g = guests.get(pp.strip())
            if g is not None:
                g.blacklist(datetime.strptime(dateReported.strip(), "%d-%b-%Y").date(), 
reason.strip())

        infile.close()
        return guests

    def setupAmenities(self):
        amenities = [ ]
        infile = open("SharedAmenity.txt", "r")
        for line in infile:
            itemCode,desc,price = line.split(",")
            amenities.append( SharedAmenity(itemCode, desc, float(price)))
        infile.close()

        infile = open("InRoomAmenity.txt", "r")
        for line in infile:
            itemCode,desc,price,floorArea = line.split(",")
            amenities.append( InRoomAmenity(itemCode, desc, float(price), 
float(floorArea)))
        infile.close()
        return amenities

    def setupRoomAvailability(self, filename):
        roomAvailability = { }
        infile = open(filename, "r")
        for line in infile:
            dateString, standardCount, deluxeCount = line.split(",")
            thisDate = datetime.strptime(dateString, "%d-%b-%Y").date()
            roomAvailability[thisDate] = [int(standardCount), int(deluxeCount)]

        infile.close()
        return roomAvailability

    def saveRoomAvailability(self, filename):
        outfile = open(filename, "w")
        for k, v in self._roomAvailability.items():
            print("{},{},{}".format(k.strftime("%d-%b-%Y"), v[0], v[1]), file=outfile)
        outfile.close()
```

---- END OF ASSIGNMENT ----