ROBERT GORDON UNIVERSITY ABERDEEN

Mock Lab Assessment Brief (2023-2024)

Academic Year	**2023-24**
Semester	**SEM 1**
Module Number	**CMM024**
Module Title	**Object-Orientated Programming**
Assessment Method	**Mock Lab Assessment on Campus**
Deadline (time and date)	**22/11/2023**
Submission	**CampusMoodle.**
Word Limit	**N/A**
Use of Generative Artificial Intelligence (AI) text	**NOT authorised**
Module Co-ordinator	**Jean Claude Golovine**

What knowledge and/or skills will I develop by undertaking the assessment?

- Interpret a set of requirements to develop procedural code.
- Interpret a set of requirements and design and implement Java classes based on those requirements and interpret class behaviour presented in UML Class diagrams.
- Critically appraise an object-oriented design from the perspectives of: Satisfying requirements by:
    - Achieving encapsulation and data hiding
    - Reusing existing classes where appropriate
    - Designing for future re-use and ease of code maintenance.
- Critically employ the principles of object-oriented design in the context of your class design and implementation. Including use of:
    - Instance variables and class variables
    - Constructor, accessor, and transformer methods
    - Object and class methods
    - Abstraction and interfacing

What knowledge and/or skills will I develop by undertaking the assessment?

On successful completion of the assessment students will be able to achieve the following Learning Outcomes:

1. Produce a software solution to a problem using key constructs and mechanisms in a procedural language.
2. Analyse a set of requirements to create an object-Orientated Design.
3. Create a working solution from a design using an object-Orientated programming language.
4. Effectively communicate an Object-Orientated design using an appropriate notation.

Please also refer to the Module Descriptor, available from the module Moodle study area.

What is expected of me in this assessment?

Background

This assessment has two domains. The first is focussed on procedural programming , which focusses on in the first 4 sessions. The second is about Object Orientated programming, which is covered in the session 5 onward. You will develop a simple booking system that focusses storing both Hotel and Flight bookings.

Procedural Coding Tasks (12 subgrades)

This series of tasks related to Learning Outcome (1).

Task 1: (1 subgrades)

Ability to create and java project and java file, getting user input to thereafter displaying the information entered in a formatted manner. (Lecture 1)

- Create a Java project following the information mentioned below.

Before completed the tasks below, create a Visual Studio project following these steps:

AL_[StudentID]_[Seat]

What is expected of me in this assessment?

So, if you student ID is 05571 and you are seated in the lab at seat 51, your project folder should be:

AL_05571_51

- Create a java file named Task1.java, and create a main method
- Write code to input your first name, surname, (strings) and ID (integer) using the Scanner class.
- Display this information in a well formatted message in the console as shown below.

```
226/bin Task1
Enter your first name
Jean CLaude
Enter your surname
Golovine
Enter your ID
123456
First Name: Jean CLaude Surname: Golovine ID: 123456
jean-claude@MacBook-Pro-2 Procedural %
```

Task 2: (1 subgrades)

Create a java file and be able to obtain user input to enter numerical values and conduct some operations on these values, and finally to display the result in a formatted fashion. (Lecture 1)

- Create a java file named Task2.java, and create a main method
- Create some code to:
    - Create code for user input to obtain two decimal numbers and store them in two declared variables named dividend and divisor.
    - Divide dividend by divisor and store the quotient and the remainder in appropriate variables using appropriate data types (quotient & remainder).
    - Subtract divisor from quotient and store the result in an appropriate variable using an appropriate data type.
    - Display the result as integers in the console as shown below

```
.java/jdt_ws/Procedural_e01bc226/bin Task2
Enter Divident
5
Enter Divisor
2
Quotient: 2 Remainder:  1 res = 0
jean-claude@MacBook-Pro-2 Procedural %
```

Task 3: (1 subgrades)

Create a java file and declare a constant numerical variable and assign a value to it, which is then cast into a float variable, and finally to display the result in a formatted fashion for both values.(Lecture 1 & 2)

- Create a java file named Task3.java, and create a main method
- Create some code to:
    - Declare a constant double value names **number1** and assign the value of 3.141592 to it.
    - Declare a double variable named **number2** and assign the same value as for *number1*.
    - Display both variable values in a formatted fashion in the console first using the println method and using the printf method with only 2 decimal values displayed as shown below.

```
jean-claude@MacBook-Pro-2 Procedural %  cd
in/env /Library/Java/JavaVirtualMachines/l
/Users/jean-claude/Library/Application\ Su
rocedural_e01bc226/bin Task3
Number1: 3.141592 Number 2: 3.141592
Number1: 3.14 -  Number2:  3.14
jean-claude@MacBook-Pro-2 Procedural %
```

Task 4: (2 subgrades)

Create a java file and declare two double variables to hold the height and weight of a person. Then compute the bmi and display these values. (Lecture 1 & 2)

BMR Definition: Your Basal Metabolic Rate (BMR) is the number of calories you burn as your body performs basic (basal) life-sustaining function. Commonly also termed as Resting Metabolic Rate . For men it follows the function shown below.

- Create a java file named Task4.java, and create a main method
- Create some code to:
    - Create code for user input of three decimal numbers and store these inputs in three declared variables named appropriately for height (cm), weight (kilograms) and age (years). E.g., height = 1.78 (meter), weight = 81 (kg), age (63).

What is expected of me in this assessment?

- Declare a variable called bmr.
- Calculate the bmr of a person following this formula shown below and assign the value to the bmr variable.

$$bmr = 88.362 + (13.397 * weight) + (4.799 * height) - (5.677 * age)$$

So, for the example above:

$$bmr = 88.362 + (13.397 * 81) + (4.799 * 178) - (5.677 * 63)$$

$$bmr = 88.362 + (1085.157) + (854.222) - (357.651)$$

$$bmr = 1670.09 = \approx 1670\ calories$$

- Display all in values in a formatted manner in the console as shown below.

```
Enter your weight
81
Enter your height (cm)
178
Enter your age
63
Weight: 81.00  Height:  178.00  Age: 63  BMR:  1670.09
jean-claude@MacBook-Pro-2 Procedural %
```

Task 5: (2 subgrades)

Create a java file and declare one double variable to hold the BMI of an individual. Then display the information related to the BMI following the information shown in the tables. ((Lecture 1 & 2)

- Create a java file named Task5.java, and create a main method
- Copy code from Task4.java (BMR input and display)
- Use the BMR value you obtain to display to:
  - And a **println** line to display information (shown in figure below) i.e., Normal or Overweight.
  -

Age	Max BMI value	Status	
		MBR value =< max	BMR value > max
20-29	1400	Normal	Overweight
30 - over	1250	Normal	Overweight

What is expected of me in this assessment?

```
00242a08c4a2b80b5470e0ff0/redhat.java/jdt_ws/Procedural_e0

Enter your weight
81
Enter your height (cm)
178
Enter your age
63
Weight: 81.00  Height:  178.00  Age: 63  BMR:  1670.09
Overweight
jean-claude@MacBook-Pro-2 Procedural %
```

Task 6: (1 subgrades)

This task is about demonstrating the use user input which is used in a while loop to output the output shown below (Lecture 1 & 2).

- Create a java file named Task6.java and create a main method.
- Use the Scanner class to input the number of times the loop should run.
- Create some code using a while loop so that its output mirrors what is shown in the figure below.

```
rocedural_e01bc226/bin Task6

Number of time to run the loop
10
Count = 1
Count = 2
Count = 3
Count = 4
Count = 5
Count = 6
Count = 7
Count = 8
Count = 9
Count = 10

jean-claude@MacBook-Pro-2 Procedural %
```

Task 7: (2 subgrades)

This task is about demonstrating the use user input which is used to calculate the third side of a right-angle triangle when two sides are known. (Lecture 3)

- Create a java file named Task7.java, and create a main method
- Create some code to:
    - Create one constant with a double datatype called radius of 4 for this radius.
    - The task is to compute the area of a circle given this radius and the formula shown below, using *Math.PI* constant to obtain the PI value.

$$area = 2 * \pi * radius$$

So as an example, if radius = 4 the result is 50.27

$$\text{area} = 2 * 3.14159 * 4$$

```
Circle Radius: 4.00 Area:  50.27
jean-claude@MacBook-Pro-2 Procedural %
```

- Display all in values in a formatted manner (only 2 decimal values - use printf method) in the console as shown above.

Task 8: (2 subgrades)

This task is about demonstrating knowledge in using ArrayLists etc. (lecture 4)

- Create a java file named Task8.java, and create a main method
- Create some code to:
  - Declare and create an ArrayList named "cars" to contain string objects
  - Add to the array list the elements shown in the table below and make sure they are added in the same order as shown in this table.

Element	Value
1	Toyota
2	Honda
3	BMW
4	Ford

  - Display the content of the ArrayList in a formatted fashion

```
.java/jdt_ws/Procedural_e01bc226/bin Task8
Toyota
Honda
BMW
Ford
jean-claude@MacBook-Pro-2 Procedural %
```

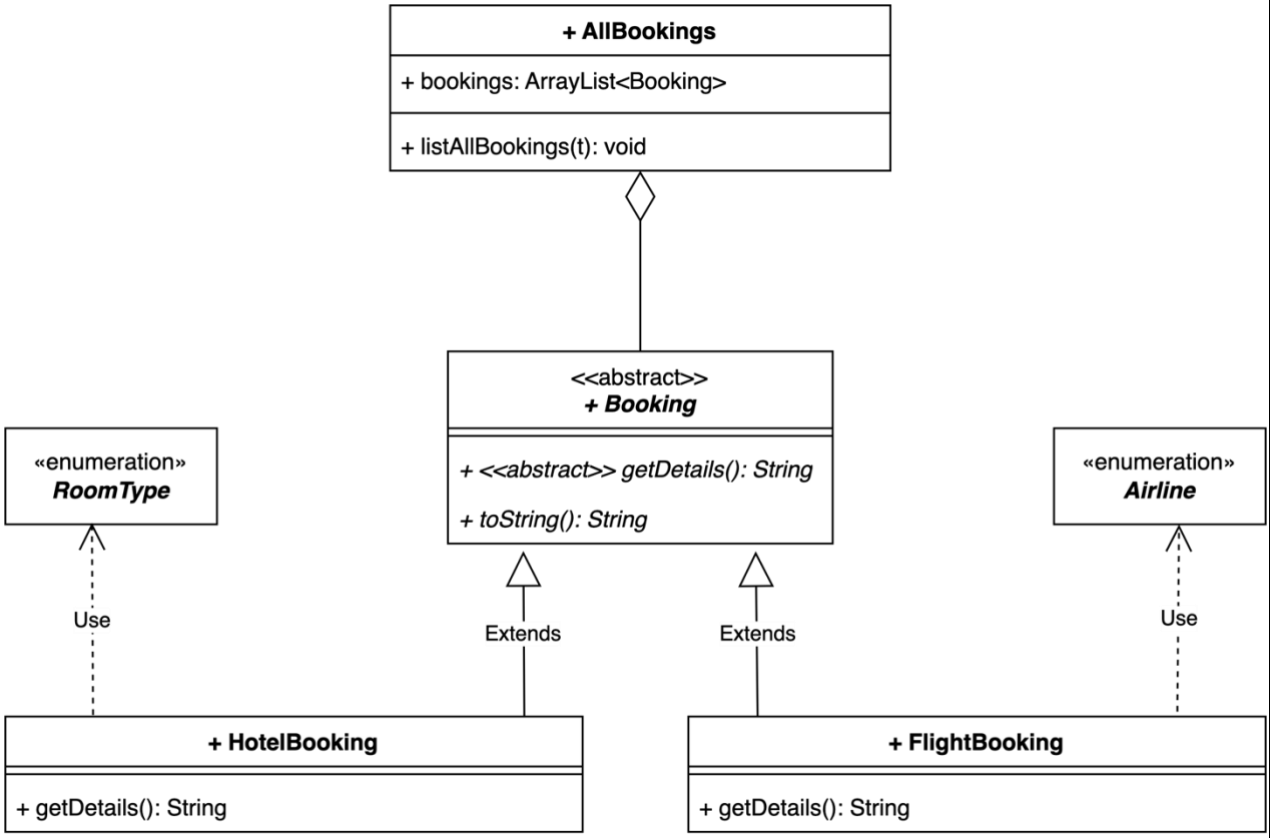
Object Orientated Coding Tasks (16 subgrades)

This series of tasks related to Learning Outcome (2,3,4).

The object Orientated model is composed of two enum and 4 classes, one of which is an abstract class and the rest (i.e., 3) are concrete classes.

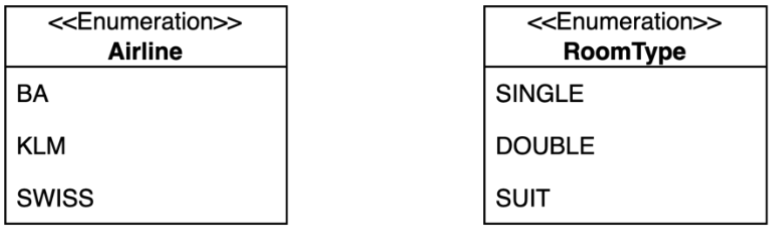
What is expected of me in this assessment?

The proposed model simulates a small and simple booking system to demonstrate object orientated programming skills.



Task 9: RoomType & Airline enum (2 subgrades)

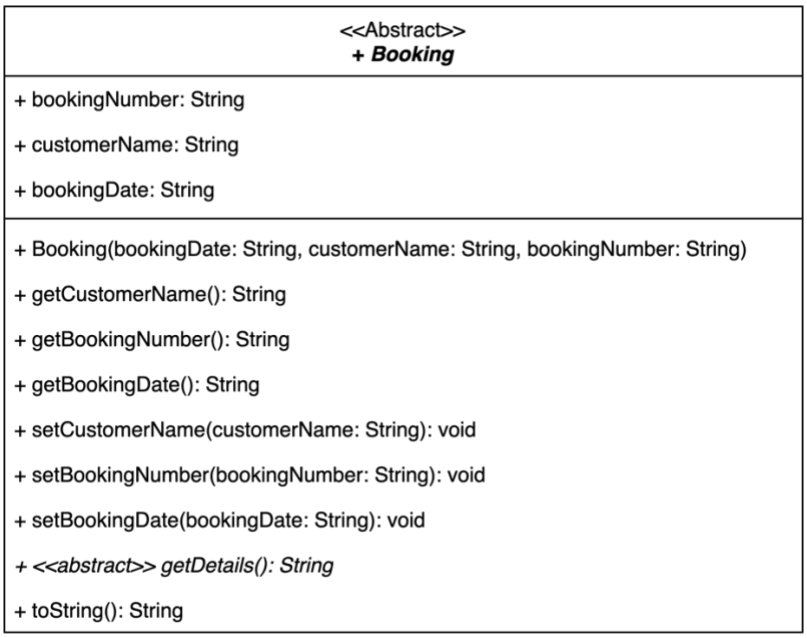
Ability to create an enum give the UML diagram.

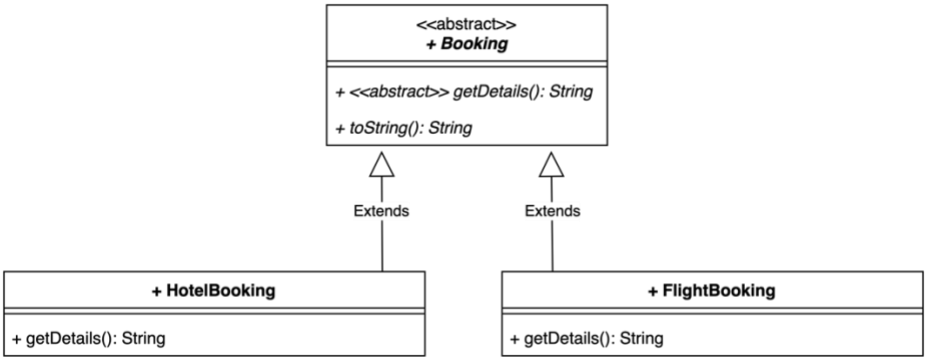


- Create two java files appropriately named for the two enum shown below.
- Create some code to:
    - Complete the enumeration for each of them as shown in the UML diagrams below.
    - **Note**: The order at which the constants are listed is very important.

Task 10: Creating the Vehicle class (4 subgrades)

Ability to create a simple abstract superclass that models a generic vehicle like a car or a bus.





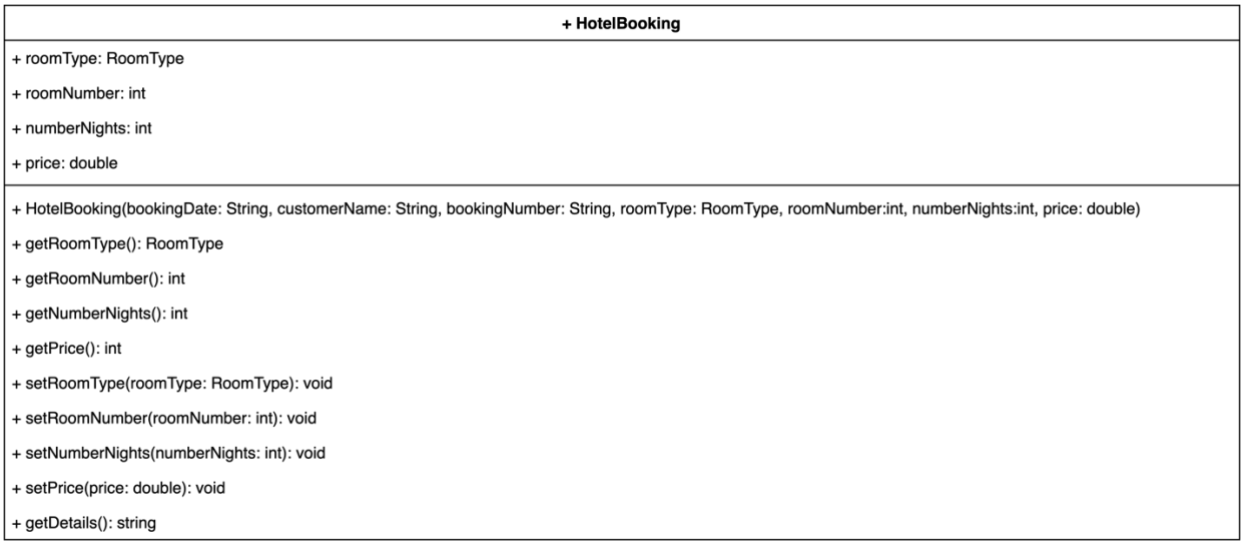
- Create a java file appropriately named this class.
- Refer to the UML diagram above to implement code below.
- Create a parameterised constructor with three parameters to initialise all object attributes.
- Create the getter and setter methods for this class.
- Create the **toString** method, which should first details of the object attributes values in a well formatted manner, and then invoke the **getDetails** method to include the

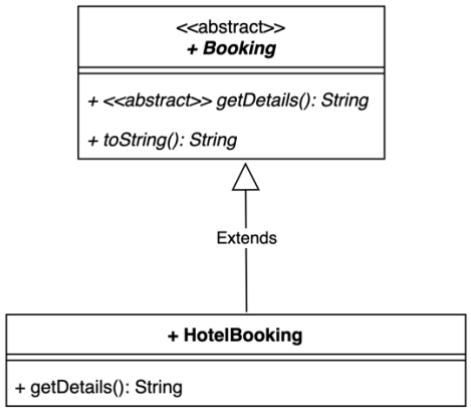
subclasses information returned by the getDetails method which should be implemented but the subclasses. See later tasks.

- Create the abstract method ***getDetails***
    - ***Note:*** *the* ***getDetails*** *is an abstract method that will be implemented by both* ***HotelBooking*** *and* ***FlightBooking*** *subclasses discussed in later tasks in which relevant information held in the object attributes is formatted into a String.*

Task 11: Creating the HotelBooking class (3 subgrades)

Ability to create a simple subclass that models a generic car.





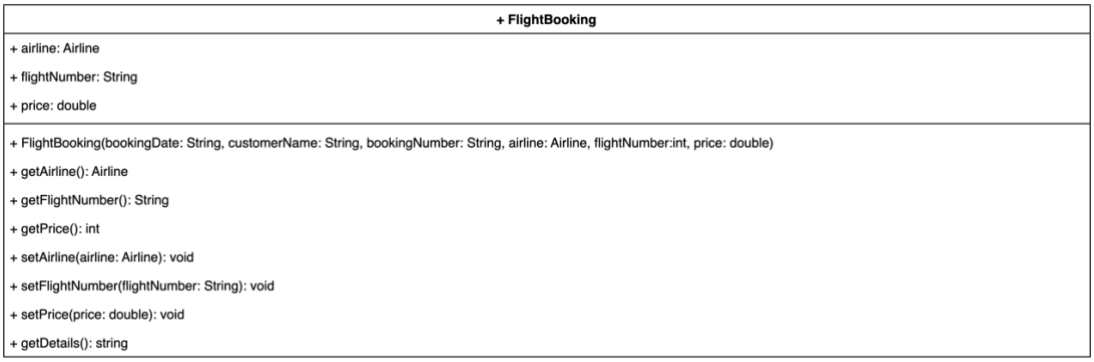
- Create a java file appropriately named this class.

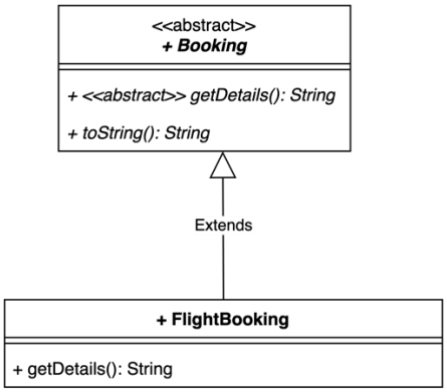
What is expected of me in this assessment?

- Refer to the UML diagram above to implement code below.
- The **HotelBooking** class is a subclass of **Booking**.
- Create a parameterised constructor with the relevant parameters (7) to initialise the superclass **Booking (3)** and the object attributes of **HotelBooking (4)**.
- Code the getter and setter methods for all attributes
- Code the ***getDetails*** method.
    - This was already discussed in Task 11 above. The method returns as a String that contains values of the object attributes properly formatted into a string.

Task 12: Creating the FlightBooking class (3 subgrades)

Ability to create a simple subclass that models a generic bus.





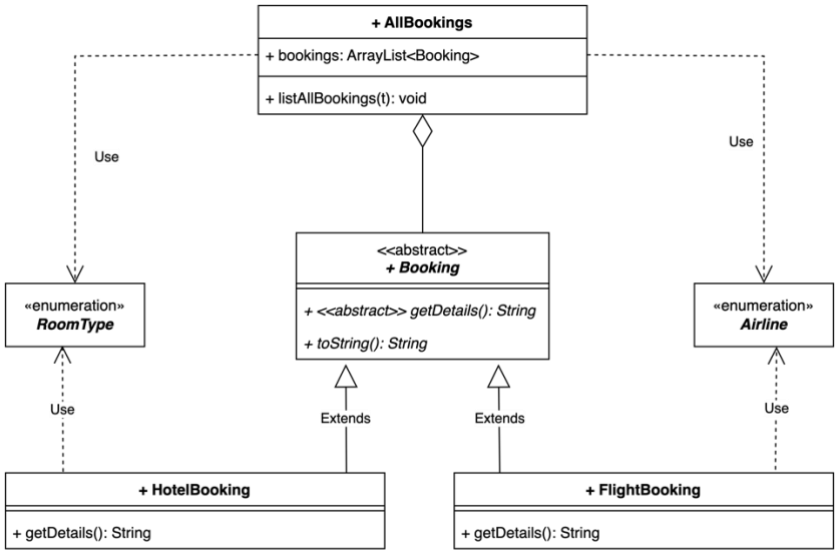
- Create a java file appropriately named this class.
- Refer to the UML diagram above to implement code below.
- The **FlightBooking** class is a subclass of **Booking**.

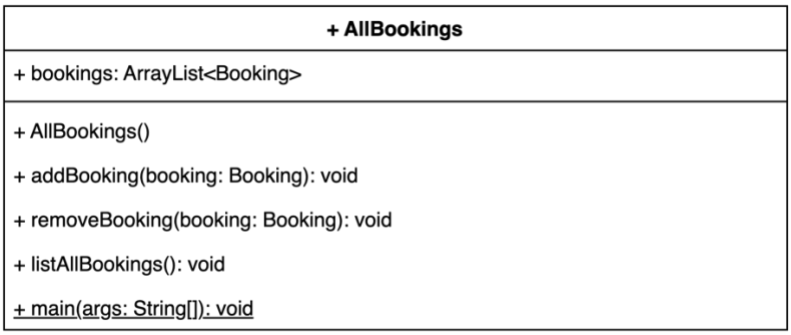
What is expected of me in this assessment?

- Create a parameterised constructor with the relevant parameters (7) to initialise the superclass **Booking (3)** and the object attributes of **FlightBooking (3)**.
- Code the ***getDetails*** method.
    - This was already discussed in Task 11 above. The method returns as a String that contains values of the object attributes properly formatted into a string.

Task 13: Creating the AllBookings (4 subgrades)

Ability to create a simple concreate that models a generic storage class.





- Create a java file appropriately named this class.
- Refer to the UML diagram above to implement code below.
- The **AllBookings** class is a simple concrete class.

What is expected of me in this assessment?

- Declare the ***bookings*** object attribute which is an ArrayList to contain **Booking** objects.
- Create a standard constructor. You should not forget to create the **bookings** ArrayList inside this constructor .
- Code the ***addBooking*** method.
    - This method takes a **Booking** parameter, which is used to append a booking to the **bookings** ArrayList.
- Code the ***removeBooking*** method.
    - This method takes a **Booking** parameter, which is used to remove a booking from the ***bookings*** ArrayList.
- Code the ***listAllBooking*** method.
    - You should iterate through all the vehicles stored in the **bookings** ArrayList and return a formatted string that contains the information for each of the booking as well as the hiring cost.
    - In addition, you should use the keyword "**instanceof**" to find if the booking is a **FlightBooking** or a **HotelBooking** and display "*Hotel Booking*" or "*Flight Booking*" depending on what the Booking is.
    - Pseudo code:

```
For each booking
```

- `Retrieve the booking information (display information about the Booking superclass)`
- `Retrieve the information from the subclasses`
- `Format the information properly`

```
Return this information as a String
```

    - Sample output to help for the formatting of the getDetails and toString methods. Note the Flight Booking before the customer's name. The Flight booking (for instance) was detected using the ***instanceof*** keyword. The rest of the information is obtained after invoking the Booking toString method.

What is expected of me in this assessment?

```
9837d842a0419a0/redhat.java/jdt_ws/OO_4ce9b169/bin
Flight Booking
Customer Name: JCG
Booking Number: B234
Booking Date: 14/11/2023
Booking Details
*****************************
Airline: KLM FlightNumber: KL1446
Price: 241.54
*****************************

Hotel Booking
Customer Name: JCG
Booking Number: B234
Booking Date: 14/11/2023
Booking Details
*****************************
Room Type SINGLE Room Number 146 Number Nights 2
Price 190.75
*****************************
```

- Code the ***static main*** method.

  You should create a static main method to test the **AllBookings** solution by creating one Hotel Booking and one Flight Booking, adding them to the booking storage by invoking the relevant method, and then displaying the information by invoking the ***listAllBooking*** method coded above.

  - Sample Data for the creation of objects:

Type	Name	Booking number	Customer	Airline	Flight Number	Price
Flight	14/11/2023	B234	"JCG"	KLM	KL1446	241.54
Type	**Name**	**Booking number**	**Customer**	**Room Type**	**Room Number**	**Price**
Hotel	17/11/2023	F246	"David"	SINGLE	146	190.75

Uploading your work to Moodle

Read the section "Important Notes" below

Important notes

Useful Resources

What is expected of me in this assessment?

On the CMM024 Moodle page you have access to copies of the CMM024 lecture notes and Lab Sheets. But you do not have access to Lab Sheet solutions etc and videos.

Uploading your work after coding on Moodle

You must zip the folder project i.e., where you created your classes into one Zip file. The name of the zip file is as follow:

YourStudentID_YouName

How will I be graded?

A grade will be provided for each criterion on the feedback grid which is specific to the assessment.

The overall grade for the assessment will be calculated using the algorithm below.

A	At least 50% of the feedback grid to be at Grade A, at least 75% of the feedback grid to be at Grade B or better, and normally 100% of the feedback grid to be at Grade C or better.
B	At least 50% of the feedback grid to be at Grade B or better, at least 75% of the feedback grid to be at Grade C or better, and normally 100% of the feedback grid to be at Grade D or better.
C	At least 50% of the feedback grid to be at Grade C or better, and at least 75% of the feedback grid to be at Grade D or better.
D	At least 50% of the feedback grid to be at Grade D or better, and at least 75% of the feedback grid to be at Grade E or better.
E	At least 50% of the feedback grid to be at Grade E or better.
F	Failing to achieve at least 50% of the feedback grid to be at Grade E or better.
NS	Non-submission.

*If the word count is above the specified word limit by more than 10% or the submission contains an excessive use of text within tables, the grade for the submission will be reduced to the next lowest grade.

Feedback grid

GRADE	A	B	C	D	E	F
DEFINITION / CRITERIA (WEIGHTING)	EXCELLENT Outstanding Performance	COMMENDABLE/VERY GOOD Meritorious Performance	GOOD Highly Competent Performance	SATISFACTORY Competent Performance	BORDERLINE FAIL	UNSATISFACTORY Fail
Procedural coding related tasks (Tasks 1 to 8) (12 subgrades)	Your coding of the procedural tasks is excellent overall. There may be some small errors but it is an outstanding effort.	Your coding of the procedural tasks is overall a solid effort but there are some small errors at times or omissions.	Overall, this is a competent effort but it is not perfect as it has minor errors as well as more serious ones.	Overall, an effort was made but there are too many coding errors; with some of them of a serious nature. This is addition of lack of implementation for some tasks.	Overall, this is a poor effort. Too many small or more serious with too much code that was not implemented, or code that was implemented but not relevant.	Minimal Attempt for all criteria. If the coding was attempted, it was not relevant or not comprehensible.
Object-Orientated related tasks (Task 9 to 13) (16 subgrades)	The coding of the classes is overall excellent. There are very few mistakes. The superclass and the two subclasses are properly coded and a very good attempt with the AllBookings class. Almost all the functionality is present.	The coding of the classes is overall a very good effort. The superclass and the two subclasses are mostly properly coded and there was a good attempt at programming the AllBookings class. There are some omission and errors though. Most of the functionality is present.	The attempt at coding the classes is commendable with most of the expected code functional. The coding of the subclasses and superclass has issues but the OO modelling is correct. Some efforts were made at coding the AllBookings class	Many classes has major errors that affects the functionality badly. An attempt was made in terms of OO design but there might also be some issues. There is also a lot of code that was not implemented or not relevant. An attempt may have been made at coding the store class i.e., AllBookings but again there are issues	A small attempt at coding was made. There are many serious issues or code not implemented or what was implemented is not relevant to the tasks. Overall, a poor attempt at completing the tasks.	Minimal Attempt for all criteria. If the coding was attempted, it was not relevant or not comprehensible

Coursework received late, without valid reason, will be regarded as a non-submission (NS) and one of your assessment opportunities will be lost.

What else is important to my assessment?

What is the Assessment Word Limit Statement?

It is important that you adhere to the Word Limit specified above. The Assessment Word Limit Statement can be found in Appendix 2 of the RGU Assessment Policy. It provides detail on the purpose, setting and implementation of wordage limits; lists what is included and excluded from the word count; and the penalty for exceeding the word count.

What's included in the word count?

The table below lists the constituent parts which are included and excluded from the word limit of a Coursework; more detail can be found in the full Assessment Word Limit Statement. Images will not be allowed as a mechanism to circumvent the word count.

Excluded	Included
Cover or Title Page	Main Text e.g. Introduction, Literature Review, Methodology, Results, Discussion, Analysis, Conclusions, and Recommendations
Executive Summary (Reports) or Abstract	Headings and subheadings
Contents Page	In-text citations
List of Abbreviations and/or List of Acronyms	Footnotes (relating to in-text footnote numbers)
List of Tables and/or List of Figures	Quotes and quotations written within "…"
Tables – mainly numeric content	Tables – mainly text content
Figures	
Reference List and/or Bibliography	
Appendices	
Glossary	

What are the penalties?

The grade for the submission will be reduced to the next lowest grade if:

- The word count of submitted work is above the specified word limit by more than 10%.
- The submission contains an excessive use of text within Tables or Footnotes.

What else is important to my assessment?

What is plagiarism?

Plagiarism is "the practice of presenting the thoughts, writings or other output of another or others as original, without acknowledgement of their source(s) at the point of their use in the student's work. All materials including text, data, diagrams or other illustrations used to support a piece of work, whether from a printed publication or from electronic media, should be appropriately identified and referenced and should not normally be copied directly unless as an acknowledged quotation. Text, opinions or ideas translated into the words of the individual student should in all cases acknowledge the original source" (RGU 2022).

What is collusion?

"Collusion is defined as two or more people working together with the intention of deceiving another. Within the academic environment this can occur when students work with others on an assignment, or part of an assignment, that is intended to be completed separately" (RGU 2022).

For further information please see Academic Integrity.

What if I'm unable to submit?

- The University operates a Fit to Sit Policy which means that if you undertake an assessment then you are declaring yourself well enough to do so.
- If you require an extension, you should complete and submit a Coursework Extension Form. This form is available on the RGU Student and Applicant Forms page.
- Further support is available from your Course Leader.

What additional support is available?

- RGU Study Skills provide advice and guidance on academic writing, study skills, maths and statistics and basic IT.
- RGU Library guidance on referencing and citing.
- The Inclusion Centre: Disability & Dyslexia.
- Your Module Coordinator, Course Leader and designated Personal Tutor can also provide support.

What are the University rules on assessment?

The University Regulation 'A4: Assessment and Recommendations of Assessment Boards' sets out important information about assessment and how it is conducted across the University.