Assignment 2

Academic Integrity:

- This is an individual assessment.
- You are permitted to refer to the Internet for Kotlin syntax. However, following tutorials, homework help websites, or using full/partial solutions from the internet is not permitted.
- You are *not* permitted to share code/references with other learners or discuss solutions/approaches.

Submission Requirements:

- Once you are done with the task, perform the following steps:
    - Click on the File menu in IntelliJ, select Export à Project to Zip file.
- Upload the **.zip file** named as **A2_YOUR_FIRST_NAME** to the submission folder.

GRADING CRITERIA

Grading is based on functional correctness and the learner's object oriented design. Besides implementing the required functionality, submissions are also evaluated based on usage of coding conventions demonstrated in class, professional organization of the code, alignment, and clarity of variable/function naming. Comments for code snippets are recommended but not required.

ACADEMIC INTEGRITY

1. Learners are responsible for familiarizing themselves with the college's Academic Integrity policies.
2. Learners may use the Internet for researching Kotlin syntax.
3. This is an individual assessment and collaboration with others (either inside or outside the class) is **not** permitted. Collaboration includes: working with others to develop a solution, showing your code to others as a 'reference', or requesting others to provide you a 'reference'.
4. You are **not** permitted to search the Internet for full or partial solutions, follow online tutorials/videos/websites that contain similar tasks/activities, or to seek assistance on homework sharing websites.

Problem Description

You were hired by a local travel agency to build an application that manages flight reservations. Create IntelliJ project using Kotlin named as **A2_YOUR_FIRST_NAME** (such as **A2_JOHN**) to work on the following:

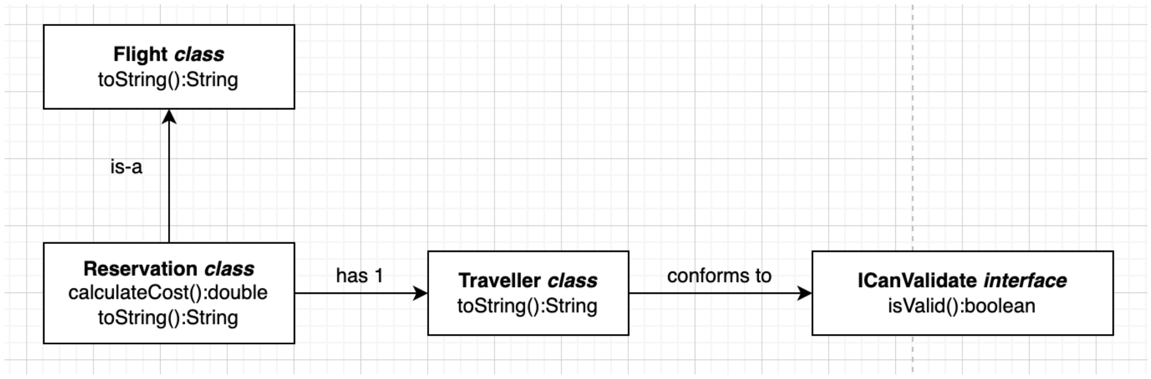
Based on the following class diagram and descriptions, create necessary classes and interfaces.

- For each class you should provide appropriate constructors and toString() functions.
- The toString() function must output the relevant details of the object.
- You are responsible for deciding the most appropriate data type of your class properties.

Class diagram

In the diagram below:

- **Is-a** refers to inheritance
- **has** refers to composition
- **conforms to** refers to implementation of an interface



ICanValidate Interface

This interface contains a function called **isValid()**. The function returns a boolean.

Traveller Class:

1. Create a ***Traveller*** class that implements the **ICanValidate** interface.

2. The Traveller class has the following properties

- Name
- Passport number
- Email address

3. Where appropriate, create required constructors and properties. You are responsible for making the best decision about the data type for each property.

4. A passport number is a 5 digit number between 10000 and 99999, inclusive. All other values are invalid.

5. The Traveller class' implementation of the isValid() function must determine whether the Traveller's passport number is valid. If yes, then return true, otherwise return false.

6. Create ***toString*()** method which returns a string comprising all the information about the passenger.

Flight Class:

1. Create a class called ***Flight*** to have following members:

- Flight Number:
- Departure airport
- Arrival airport
- Flight Distance (between the arrival and departure cities in **Miles**)

2. Create the required constructors and properties. You are responsible for making the best decision about the data type for each property.

3. Create ***toString*()** method which returns a string comprising all the information about the flight.

Reservation Class:

1. Create a class called **Reservation** which will represent an air ticket reservation. The Reservation class inherits all the **Flight** class attributes and methods.

2. In addition to the inherited properties, the Reservation class must also contains the following properties:
- Reservation Id
- Passenger (Traveller class object)
- Seat number

3. Create required constructors and properties. You are responsible for making the best decision about the data type for each property.

4. The reservation id must be automatically generated at the time of object creation (not via constructor parameters). The reservation id is of the following format: XXXX, where XXXX is a randomly generated 4 digit number between 1000-9999 inclusive. After a reservation id is *set*, it can be *retrieved,* but *not* modified.

4. Add a ***calculateCost()*** method that calculates and returns the cost of the reservation. The formula is: $100 + (Flight Distance * $0.12)

5. Create ***toString*()** method which returns a string comprising all the information about the reservation. The output of the reservation should look like this:

```
Passenger: Carmen Pope
Reservation id: 7572
Flight Information: Flight: AC306
```

```
Departure Airport: PHX
Arrival Airport: YUL
Distance: 1832.8000000000002
Seat number: 999Q
```

Main.kt

After implementing your classes, create a Main.kt file. In your Main.kt file, write the code to perform the operations described below. In your code, ensure your output is **easily readable** by using new lines/whitespacing and - - - - symbols to delimit the sections of your program output. Refer to the *sample* output for an example.

Your main program should output appropriate and obvious messages that indicate the results of each operation. For example, when a reservation is created, the program should output "Reservation created".

When outputting data about objects to the screen, you must use that object's **toString()** function.

LIST OF OPERATIONS

In your Main.kt class, write the code to perform the following tasks:

1. Create the following variables:
   - An empty list of Traveller objects
   - An empty list of Flight objects
   - An empty list of Flight Reservation objects.

Add the following entities to the list of Travellers:

Name	Email address	Passport number
Bob Smith	bob@gmail.com	Assign a VALID passport number to Bob
Abigail Diaz	abigail@gmail.com	Assign an **INVALID** passport number to Abigail
Carmen Pope	carmen@gmail.com	Assign a VALID passport number of Carmen.

Add the following entities to the list of Flights:

Flight Number	Departure Airport	Arrival Airport	Distance
AA281	DFW	ICN	6835.70 miles
AC306	PHX	YUL	2291.00 miles
WN3855	PHX	STL	1261.38 miles

2. After you have created and populated your lists, write the code to perform the following operations:

Section 1: Searching for a Traveller and determining if a passport is valid

- Using a loop, search the list of travellers to *search* for a customer whose name includes the word "Smith". If a matching object can be found.
    - If the result of your loop was that a matching traveller was found, output "Traveller found", and output the information about the traveller to the screen.
    - If the result of your loop was that *no* matching traveller was found, output "No matching travellers found".

Section 2: Is the passport valid?

- Retrieve Abigail's object from the Traveller's array using Abigail's position in the array. Programmatically determine if Abigail has a valid passport number. **You should use the Traveller class' *isValid* function.** If valid, then output a message indicating that the number is valid. If invalid, then output a warning message and update their passport number to a valid number.

Section 3: Creating Reservations

- Create the following reservations for Bob and Carmen. The information about flights and travellers must be directly retrieved from the *appropriate arrays* using the object's position number in the array:
    - A reservation for Bob on WN3855
    - A reservation for Bob on AA281
    - A reservation for Carmen on Flight AC306

- Add the reservations to the reservations array.

Section 4: Outputting Information about Reservations

- Programmatically calculate and output the total number of reservations containing flights departing PHX. Your implementation must work for a reservations list of *any size*, not just the reservations array created in the previous step.
- Programmatically calculate and output the total revenue generated from all reservations. The total revenue is the *sum* of all the all reservation costs (ie: total revenue = reservation 1 cost + reservation 2 cost + reservation 3 cost + reservation 4 cost + ….). Your implementation must work for a reservations list of *any size*, not just the reservations array created in the previous step.

Section 5: Modifying a Reservation

- Retrieve Carmen's reservation from the reservations array using Carmen's position number in the array. Using the retrieved reservation:
    - Output Carmen's current reservation details to the screen.
    - Change Carmen's seat number to a different value
    - Reduce the flight distance on Carmen's reservation by 10%

- Using Carmen's reservation object,
    - Output Carmen's updated reservation details to the screen
    - Output the cost of the reservation *before* the flight distance changed
    - Output the cost of the reservation *after* the flight distance changed.

Sample output

Your output and values may vary depending on your implementation.

```
------------ Searching for a Traveller ----------
Traveller found.
Name: Bob Smith
Email: bob455@gmail.com
Passport Number: 10000

------------ CHECKING PASSPORT ----------
Abigail's passport is invalid

------------ RESERVATIONS INFORMATION ----------
Number of reservations departing PHX: 2
Total revenue from reservations: $1545

------------ CARMEN'S RESERVATION ----------
Passenger: Carmen Pope
Reservation id: 2808
Flight Information: Flight: AC306
Departure Airport: PHX
Arrival Airport: YUL
Distance: 2291.0
Seat number: 8A

------------ UPDATED RESERVATION ----------
Passenger: Carmen Pope
Reservation id: 2808
Flight Information: Flight: AC306
Departure Airport: PHX
Arrival Airport: YUL
Distance: 1832.8000000000002
Seat number: 999Q

Old reservation cost: $374.92
Updated reservation cost: $319.93600000000004
```