Instructions

1. All your personal devices (laptop, cell phone, tablet...) must be turned off.
2. The only programs you are allowed to use during the exam are: eclipse, notepad, command line and chrome browser.
3. Only pages you allowed to visit during the exam are LMS and links from the course page on LMS.
4. Speaking or communicating with anyone during the exam is considered a cheating attempt and will result in the grade of zero.

Question

1. Implement the classes: Passenger, Flight, TestFlight.

Class Passenger

2. This class represents a person who intends to board the flight. The attributes, constructors and methods are given below:

```
Passenger
-------------------------------------------
- id: integer
- name: string
- age: integer
- gender: char
-------------------------------------------
+Passenger(integer, string, integer, char)
+Passenger(Passenger)
+getId():integer
+getName():string
+getAge():integer
+getGender(): char
+equals(Passenger):boolean
+toString():string
```

3. Attributes:
   - `id`: an integer attribute that represents the person's id
   - `name`: a string attribute that represents the person's name
   - `age`: an integer attribute that represents the person's age
   - `gender`: a character attribute, stores 'm' for males and 'f' for females
4. Constructors and Methods
   - A constructor that takes four parameters to initialize the four attributes.
   - A copy constructor that takes a parameter of type `Passenger`.
   - Getter methods, one for each of the four attributes

- `equals()`: a method that takes a parameter of type `Passenger` and compares it to this object and returns `true` if they have the same `id` and `false` otherwise.
- `toString()`: a method that returns the name of the passenger in the format:
  `<title> <name>`
  where title is either (Mr.) for males who are 18 years or older, (Master) for males below 18 years, (Mrs.) for females whoe are 18 years or older and (Miss) for females below 18 years.

Class Flight

5. This class represents an air trip (a flight). The attributes, constructors and methods are given below:

```
Flight
------------------------------------------------
- flight_no: string
- passengers: Passenger[]
- capacity: integer
- nPassengers: integer
- baseFare: double
------------------------------------------------
+Flight(string, integer, double)
+findPassenger(Passenger):integer
+findPassenger(integer):integer
+reserveTicket(Passenger):boolean
+cancelTicket(integer):boolean
+displayManifest()
+getFlightRevenue(): double
+getFlightWithHigherRevenue(Flight):Flight
```

6. Attributes:
   - `flight_no`: a string attribute that represents the flight number.
   - `passengers`: an array that contains the `Passenger` objects of passengers who are boarding the flight.
   - `capacity`: integer attribute represents the number of seats of the flight.
   - `nPassengers`: integer attribute represents the actual passengers who are boarding the flight.
   - `baseFare`: double attribute that represents the amount in Saudi Riyals that should be paid by each passenger who is 12 years or older. Passengers between 2 years and 12 years will pay half the baseFare, while infants below 2 years travel for free with accompanying adult.
7. Constructor and Methods
   - A constructor that takes the flight number, the maximum capacity and the base fare as parameters, and initializes the attributes of the flight.
   - `findPassenger(Passenger)`: a method that takes an object of class `Passenger` and returns the index of this object in the `passengers` array if it exists, and `-1` otherwise.

- `findPassenger(integer)`: a method that takes an `id` of a `passenger` and returns the index of the passenger object in the `passengers` array who has such id, if found, and `-1` otherwise.
- `reserveTicket(Passenger)`: a method that takes a parameter of class Passenger and tries to add the `passengers` of the flight if possible. A passenger cannot be added twice to the same flight. If the method succeeds it returns `true`, and `false` otherwise.
- `cancelTicket(integer)`: a method that takes a passenger's `id` as a parameter and tries to remove the passenger with this `id` from the passengers of the flight. If the method succeeds it returns `true`, and `false` otherwise.
- `displayManifest()`: a method that displays information about the flight and its passengers: it first prints the flight number, then list of all of the passengers and their seat (same as the index of the passenger), then summary of the number of passenger and empty seats.
- `getFlightRevenue()`: a method that returns the sum of all the fares paid by all passenger who are boarding the flight. Again: full fare should be paid by each passenger who is 12 years or older. Passengers between 2 years and 12 years will pay half the baseFare, while infants below 2 years travel for free with accompanying adult.
- `getFlightWithHigherRevenue(Flight)`: a method the takes another flight object as parameter, compares its revenue to that of this flight, and returns the flight object that has the higher revenue.

Class TestFlight

8. The class has a main method that should do the following:

- Create two flights with the following information:

Flight number	**Number of seats**	**Base fare**
SV780	100	550
AR212	50	320

- Create five passenger objects and add them to the flights as follows:

Flight	Passenger id	Name	Age	Gender
SV780	30213	Ali	17	male
	74362	Noura	55	female
	53621	Fahad	1	male
AR212	84735	Sarah	30	female
	89763	Saleh	42	male

- Remove Noura from flight SV780 and add her to flight AR212

- Print the manifest of the flight with higher revenue.
-

- Here is a sample run:

```
Flight#: AR212
====================
Seat  Passenger
--------------------
1     Mrs. Sarah
2     Mr. Saleh
3     Mrs. Noura
--------------------
Total number of passengers: 3
Number of empty seats: 47
```