

# Java For Industry: Final coursework

The coursework is out of 50, question 1, the Airline booking system is worth 35 marks. There are an additional 5 marks for coding style and organisation. A final extension section is worth 10 marks. You can achieve an excellent mark without attempting the extension (up to 80%). Don't attempt the extension until you have finished the rest of the coursework.

You shouldn't need imports other than those found in `java.util` or [java.io](#) or `java.lang`. If you want to use anything outside of these.















When you have completed your answers. Upload the completed IntelliJ project to learn.gold as a zip file.

This coursework is out of 50 marks. 35 marks for Question 1 10 for Question 2 and 5 marks for style, commenting and code organisation.

## Question 1: Airline booking

In this coursework you are going to prepare a simple airline booking system. The system will involve using inheritance and abstract classes, along with reading data in from a file. If you haven't already, please download the template files..

The files are structured as follows, with descriptions to the right:

 AirlineBooking	
 out	
 src	
 Q1Main.java	Launches the program and outputs flight data
 GeneratePassengers.java	creates the txt files, you don't need to edit or use this!
 293Seats.txt	seats layout for aircraft type
 387Seats.txt	seats layout for aircraft type
 AirlineBooking.iml	intelliJ project file
 flight1.txt	passenger details and crew for a specific flight
 flight2.txt	passenger details and crew for a specific flight
 flight3.txt	passenger details and crew for a specific flight
 flight4.txt	passenger details and crew for a specific flight
 flight5.txt	passenger details and crew for a specific flight
 flight6.txt	passenger details and crew for a specific flight



flight7.txt

passenger details and crew for a specific flight



flight8.txt

passenger details and crew for a specific flight

You are going to be adding new classes to this existing template to complete the system. You will be able to tell how effective your implementation is if the provided Main class executes successfully, producing the same output as shown in Appendix A of this coursework description. You shouldn't need to edit the Main class or GeneratePassengers.java. When marking we will verify your code against these files.

Appendix B is a class diagram representing the completed programme, you can use this to clarify the descriptions below and validate your implementation.

## Part 1 Person (10 marks)

Create a person class in the project. Person should have the following private instance variables. Create a getter and setter method for each variable.

- A string **name**
- an integer **passportNumber**

Also, create a constructor for the class with arguments for **name** and **passportNumber**.

Add a public abstract method **calculatePersonWeight** to the class it should return a double.

## Concrete classes

You are now going to create 2 concrete classes, **CrewMember** and **Passenger** That inherit from Person.

**CrewMember** should have a constructor that takes a string **name** and a passport number. It should call the super constructor you created in person.

**Passenger** needs 2 additional private instance variables, which also require getters and setters.

- An integer **holdBags** (to store the number of hold bags they have booked)
- A string **flightClass** (whether they are travelling economy or first class.)

Both CrewMember and Passenger need to implement the **calculatePersonWeight** method from person. Complete these methods using the following table as guidance.

This isn't an accurate weight for each passenger but represents an estimate of the weight the airline needs to allow for when calculating the planes total weight and fuel. First class passengers have additional meal services hence the larger weight requirement.

Person	class	weight
crewMember	-	75
passenger	first	87.5 plus 25 for each bag

passenger	economy	80 plus 25 for each bag
-----------	---------	-------------------------

## Part 2 Aircraft (6 marks)

Create an aircraft class that has the following private members and create getters and setters for each one.

- A string **make**
- A string **model**
- A String **tailNumber**
- A double **craftWeight**
- A double **maximumTakeoffWeight**
- A File **layoutFile** (you will need to import *java.io.File*)

Also write a constructor for the class that populates the above variables in the order shown above.

Override the **toString** method for the class printing out the aircrafts, make, model, tail number, weight and maximum takeoff weight.

## Part 3 Seat (4 marks)

Create a new class called seat. This is going to represent the seats available on the plane. Create the following private members with a getters and setters:

- an integer **row**
- an integer **seat** (location in the row)
- a String **flyingClass**
- a Passenger **allocatedTo**

Create a constructor that sets the **row** the **seat** and the **flyingClass**. Don't set the allocation yet. This will be done using the setter when a passenger is allocated to the seat.

## Part 4 Flight (15 marks)

The flight class brings together all the components and completes the system. Flight should have the following private members

- an integer **flightNumber**
- an Aircraft **craft**
- a String **startLocation**
- a String **endLocation**
- a double **distance**

Create getters and setters for each variable. The class should also include the following public members:

- An ArrayList **seats** that stores elements of type Seat
- An ArrayList **crew** that stores elements of the type Crew

Write a constructor for Flight. that takes as arguments. flightNumber, craft, startLocation, endLocation and distance. It should use these values to initialise the private members. It should also initialise the 2 ArrayLists.

Finally the constructor should populate the **seats** ArrayList. Using the layoutFile from the craft as input create a new scanner. Each row of input represents one row of seats. Each seat in the row is separated by a comma. A first class seat is represented by an F an economy class seat by a E. You will need to read a line of input, split the string, then create a new seat object and add it to the seats arrayList. Make sure you handle the exception generated by the scanner carefully and it doesn't terminate the programme.

Also, add the following methods to Flight:

### calculateTakeOffWeight:

**CalculateTakeOffWeight** should take no arguments and return a double. Using the **calculatePersonWeight** method from **Passenger** and **CrewMember** and the weight of the craft calculate the total weight of the plane at takeoff. Return the weight unless it is larger than the maximum takeoff weight, in which case return -1.

### bookSeat:

**BookSeat** should take a Passenger as the only argument and return an int. Find the next available seat that matches the passengers **flightClass** and allocate the passenger to it by setting the **allocatedTo** variable of the available seat. If no seats are available in the class you can upgrade or downgrade the passenger, otherwise, the plane is full and the passenger can't be booked in. return the following values depending on the outcome:

value	outcome
1	booking completed
2	passenger upgraded
3	passenger downgraded
-1	plane full

### toString:

Override the **toString** method of **Flight** to display the flight information. It should output the following:

- A clear heading including the flight number
- Where the flight is flying from
- Where it is flying to,
- The distance covered
- How many first class passengers are booked on
- How many economy passengers
- How many seats are unallocated
- The name of each crew member
- Information about the aircraft, (you should use the Aircraft toString method to do this.)

When you have finished run the programme (If you are using IntelliJ. Ensure you have selected the 'Main' configuration. The output should match Appendix A.

## Question 2: Extension Sudoku Solver (10 marks)

Write a programme to solve sudoku puzzles. Your implementation should use the backtracking brute force method to solve puzzles. You can find a description of the method at the following wikipedia page: [https://en.wikipedia.org/wiki/Sudoku\\_solving\\_algorithms](https://en.wikipedia.org/wiki/Sudoku_solving_algorithms) You should have the following classes in your completed implementation:

Code your solution in the same IntelliJ Project as question 1.

**Cell:** The cells of a playing grid. You may want to give the cell a boolean property for whether it can be edited or not. Cells that are in the original puzzle definition wouldn't be editable.

**Grid:** The collection of cells arranged across a 2d array. You will want methods for getting and setting the value of cells. You will also want methods for determining if there is a duplicated value across rows, columns or an inner 3 by 3 grid. You will also want a method for displaying the grid to the console, perhaps overriding the toString method.

**Solver:** This class includes the actual solver. Make sure you include a static method called solve, that takes the initial grid as the only argument and returns the completed grid.

**Q2Main:** Loads a grid in from a file and runs the solver method including a public static void main(String[] args) method.

You should test your code using the extensionPuzzle.txt file on learn.gold. It should solve in 6015 iterations. Your code is expected to work with other puzzle definitions too.

# Appendix A

The output from your programme when running main. Should match the following.

PAP airways flight allocation system

Adding fleet to system...

Adding flights...

Testing file handling, should fail gracefully  
can't find layout file

Adding Passengers...

Entering passengers for flight #1

Entering passengers for flight #2

plane full can't book Delia Xeni  
plane full can't book Tilly Goodwin  
plane full can't book Thomas Brown

Entering passengers for flight #3

downgrading Winifred Xeni  
downgrading Quentin Walsh  
downgrading Agatha Holmes

Entering passengers for flight #4

Entering passengers for flight #5

downgrading Vinny English  
downgrading Rebecca Lingard  
downgrading William Firth  
downgrading Beau Underwood

Entering passengers for flight #6

upgrading Yves Goodwin  
upgrading William Goodwin  
upgrading Michaela Iqbal  
upgrading Thomas Chan  
upgrading Fiona Xeni  
upgrading Zara Walsh

Entering passengers for flight #7

Entering passengers for flight #8

upgrading Edward Venn  
upgrading Benny Lingard

-----  
\* Flight #1 \*

-----  
From: Heathrow  
To: JFK  
Distance: 5149.0  
First class passengers: 24  
Economy passengers: 110  
Unallocated seats: 2  
crew: Michaela Shiffman, Quentin Ainsley, Ulysses Kantt, Joanna Tolworth, Liam Tolworth, Michaela Munster, Georgina Munster,  
Make: Skycoach Model: 387 tailNumber: PAP-JX1 weight: 800000.0 maximum takeoff weight: 817000.0  
flight #1 weighs 816275.0KG

-----  
\* Flight #2 \*

-----  
From: Gatwick  
To: Indira Gandhi  
Distance: 6759.0  
First class passengers: 24  
Economy passengers: 112  
Unallocated seats: 0  
crew: Ursula English, Holly Kantt, Beau Iqbal, Olivia Iqbal, Freddy Zerdin, Colin Firth, Marvin Iqbal,  
Make: Skycoach Model: 387 tailNumber: PAP-JX2 weight: 800000.0 maximum takeoff weight: 817000.0  
flight #2 weighs 816810.0KG

-----  
\* Flight #3 \*

-----  
From: JFK  
To: Heathrow  
Distance: 5149.0  
First class passengers: 24  
Economy passengers: 112  
Unallocated seats: 0  
crew: Donny Iqbal, Kym Reagan, Colin Young, Asad Goodwin, Zara Quest, Queenie Brown, Nicholas Iqbal,  
Make: Skycoach Model: 387 tailNumber: PAP-JX1 weight: 800000.0 maximum takeoff weight: 817000.0  
flight #3 weighs 816932.5KG

-----  
\* Flight #4 \*

-----  
From: Heathrow  
To: Charles De Gaulle  
Distance: 482.0  
First class passengers: 12  
Economy passengers: 72  
Unallocated seats: 0  
crew: Violet Ainsley, Xanthe Venn, Thomas Young, Donny Brown, Nicholas Goodwin, Niamh Goodwin, Yves Xeni,  
Make: Lambdadier Model: 293 tailNumber: PAP-JA1 weight: 324334.0 maximum takeoff weight: 336900.0  
flight #4 is too heavy to take off!

-----  
\* Flight #5 \*

-----  
From: Orly  
To: Malpensa  
Distance: 563.0  
First class passengers: 12  
Economy passengers: 72  
Unallocated seats: 0  
crew: Zach Quest, Queenie Xeni, Tilly Reagan, Winifred Nicoles, Yves Quest,  
Ursula Tolworth, Freddy Goodwin,  
Make: Lambdadier Model: 293-8 tailNumber: PAP-JA3 weight: 324380.0 maximum  
takeoff weight: 336900.0  
flight #5 weighs 336895.0KG

-----  
\* Flight #6 \*

-----  
From: Malpensa  
To: Edinburgh  
Distance: 2300.0  
First class passengers: 12  
Economy passengers: 72  
Unallocated seats: 0  
crew: Yvonne Brown, Zach English, Freddy Oscar, Delia Young, India Munster,  
Ursula Nicoles, Zara Xeni,  
Make: Lambdadier Model: 293 tailNumber: PAP-JA1 weight: 324334.0 maximum takeoff  
weight: 336900.0  
flight #6 weighs 334874.0KG

-----  
\* Flight #7 \*

-----  
From: Indira Gandhi  
To: Schiphol  
Distance: 6213.0  
First class passengers: 24  
Economy passengers: 112  
Unallocated seats: 0  
crew: Jamie Firth, William English, Simon Reagan, Kelly Young, Zara Venn, liam  
Tolworth, Asad English,  
Make: Skycoach Model: 387 tailNumber: PAP-JX2 weight: 800000.0 maximum takeoff  
weight: 817000.0  
flight #7 weighs 816310.0KG

-----  
\* Flight #8 \*

-----  
From: Edinburgh  
To: Madrid  
Distance: 1300.0  
First class passengers: 12  
Economy passengers: 72  
Unallocated seats: 0  
crew: Violet Kantt, Beau Lingard, Queenie Walsh, India Iqbal, William Iqbal,  
Winifred Xeni, Joanna Zerdin,  
Make: Lambdadier Model: 293 tailNumber: PAP-JA1 weight: 324334.0 maximum takeoff  
weight: 336900.0  
flight #8 weighs 334979.0KG





# Airline Booking: PAP Coursework 2

