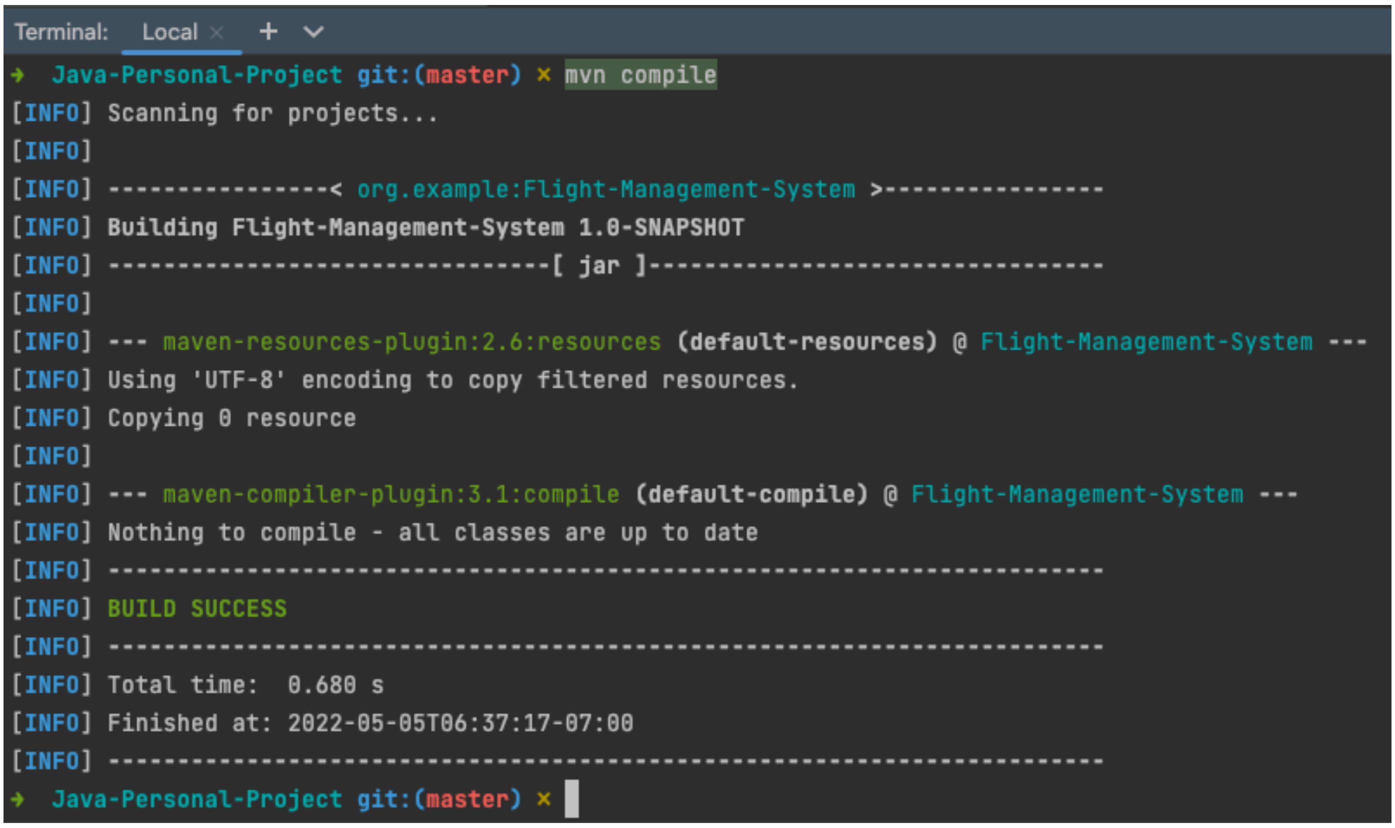
Flight Booking Application

Submitted By - Aryan Jadon (SJSU ID - 015260609)

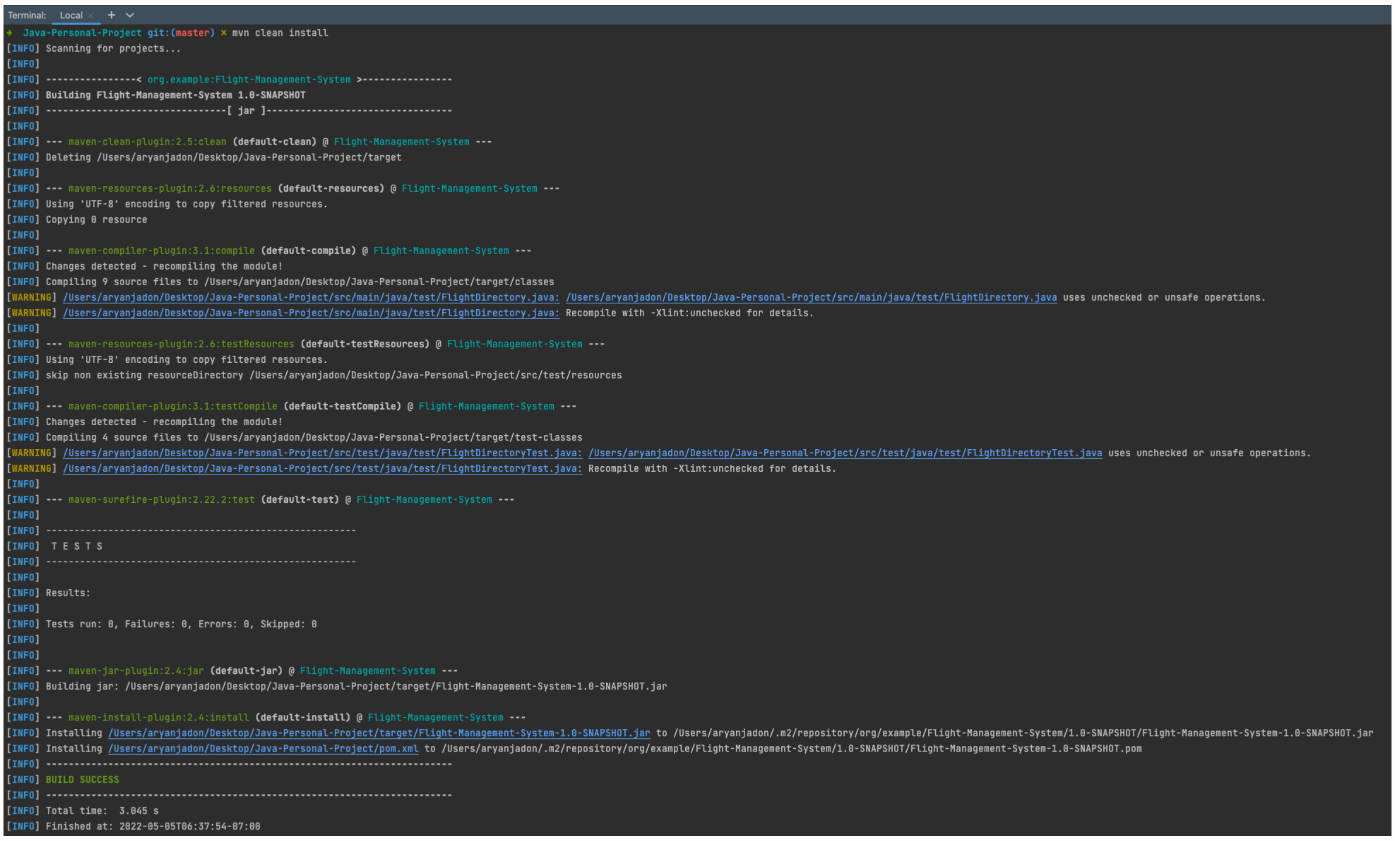
1. Instructions of building the project and steps to execute the same

a) **mnv compile**



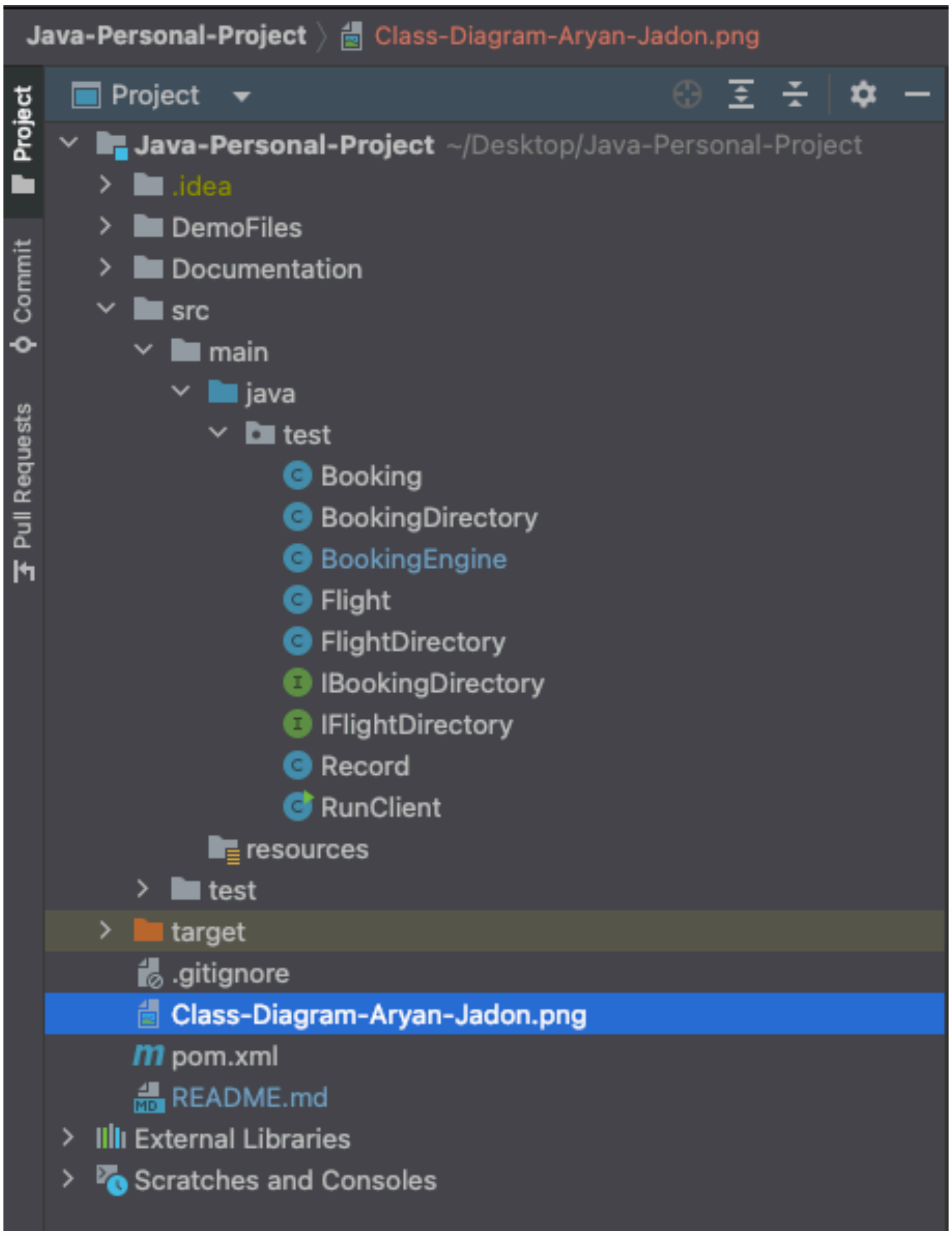
```
➜  Java-Personal-Project git:(master) ✗ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------< org.example:Flight-Management-System >----------------
[INFO] Building Flight-Management-System 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Flight-Management-System ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Flight-Management-System ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.680 s
[INFO] Finished at: 2022-05-05T06:37:17-07:00
[INFO] ------------------------------------------------------------------------
➜  Java-Personal-Project git:(master) ✗
```

b) **mvn clean install**



```
Terminal:   Local ×   +  ⌄
➜  Java-Personal-Project git:(master) ✗ mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------< org.example:Flight-Management-System >----------------
[INFO] Building Flight-Management-System 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ Flight-Management-System ---
[INFO] Deleting /Users/aryanjadon/Desktop/Java-Personal-Project/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Flight-Management-System ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Flight-Management-System ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 9 source files to /Users/aryanjadon/Desktop/Java-Personal-Project/target/classes
[WARNING] /Users/aryanjadon/Desktop/Java-Personal-Project/src/main/java/test/FlightDirectory.java: /Users/aryanjadon/Desktop/Java-Personal-Project/src/main/java/test/FlightDirectory.java uses unchecked or unsafe operations.
[WARNING] /Users/aryanjadon/Desktop/Java-Personal-Project/src/main/java/test/FlightDirectory.java: Recompile with -Xlint:unchecked for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Flight-Management-System ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/aryanjadon/Desktop/Java-Personal-Project/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ Flight-Management-System ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 4 source files to /Users/aryanjadon/Desktop/Java-Personal-Project/target/test-classes
[WARNING] /Users/aryanjadon/Desktop/Java-Personal-Project/src/test/java/test/FlightDirectoryTest.java: /Users/aryanjadon/Desktop/Java-Personal-Project/src/test/java/test/FlightDirectoryTest.java uses unchecked or unsafe operations.
[WARNING] /Users/aryanjadon/Desktop/Java-Personal-Project/src/test/java/test/FlightDirectoryTest.java: Recompile with -Xlint:unchecked for details.
[INFO]
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ Flight-Management-System ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Flight-Management-System ---
[INFO] Building jar: /Users/aryanjadon/Desktop/Java-Personal-Project/target/Flight-Management-System-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ Flight-Management-System ---
[INFO] Installing /Users/aryanjadon/Desktop/Java-Personal-Project/target/Flight-Management-System-1.0-SNAPSHOT.jar to /Users/aryanjadon/.m2/repository/org/example/Flight-Management-System/1.0-SNAPSHOT/Flight-Management-System-1.0-SNAPSHOT.jar
[INFO] Installing /Users/aryanjadon/Desktop/Java-Personal-Project/pom.xml to /Users/aryanjadon/.m2/repository/org/example/Flight-Management-System/1.0-SNAPSHOT/Flight-Management-System-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.045 s
[INFO] Finished at: 2022-05-05T06:37:54-07:00
```

c) Before Execution



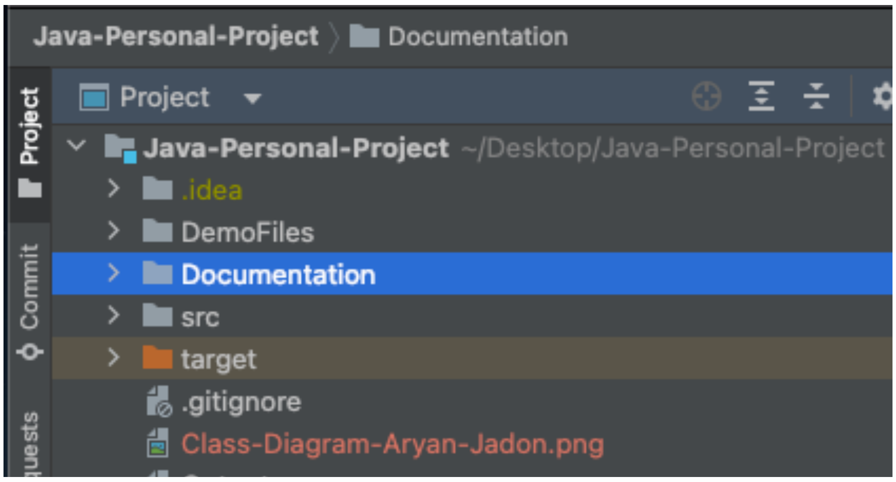
Java-Personal-Project 〉 Class-Diagram-Aryan-Jadon.png

Project

Java-Personal-Project ~/Desktop/Java-Personal-Project
.idea
DemoFiles
Documentation
src
main
java
test
Booking
BookingDirectory
BookingEngine
Flight
FlightDirectory
IBookingDirectory
IFlightDirectory
Record
RunClient
resources
test
target
.gitignore
Class-Diagram-Aryan-Jadon.png
pom.xml
README.md
External Libraries
Scratches and Consoles

d) After Execution - Output.csv and Output.txt are generated in Project Folder.

Command - **mvn exec:java -Dexec.mainClass=test.RunClient -Dexec.args="/Users/aryanjadon/Desktop/Java-Personal-Project/DemoFiles/flights.csv /Users/aryanjadon/Desktop/Java-Personal-Project/DemoFiles/Sample.csv /Users/aryanjadon/Desktop/Java-Personal-Project/Output.csv /Users/aryanjadon/Desktop/Java-Personal-Project/Output.txt"**

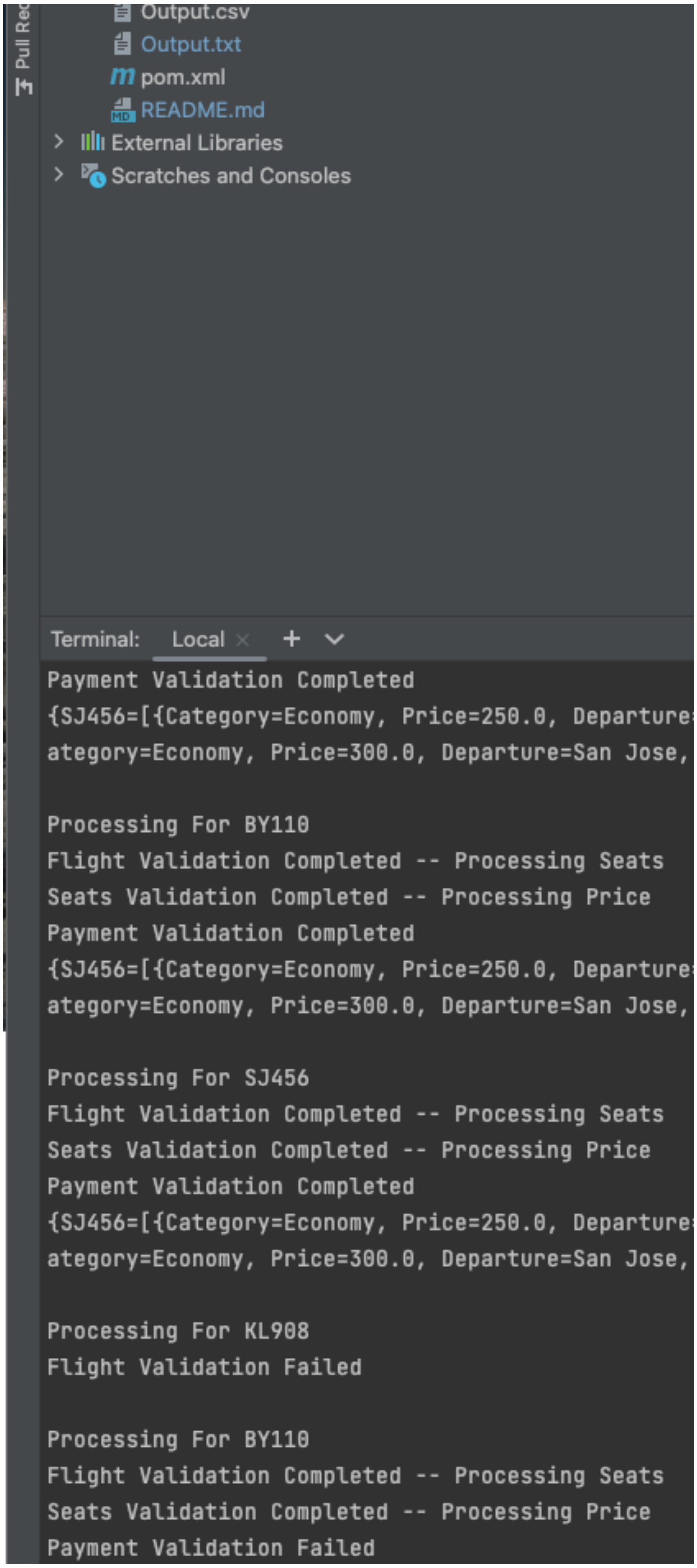


Java-Personal-Project 〉 Documentation

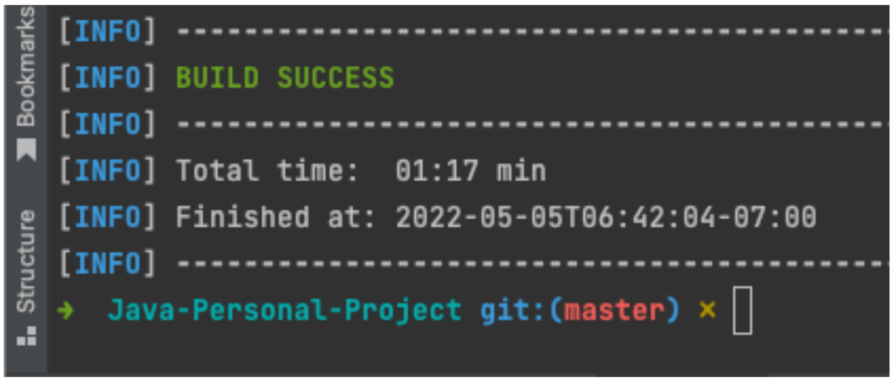
Project

Java-Personal-Project ~/Desktop/Java-Personal-Project
.idea
DemoFiles
Documentation
src
target
.gitignore
Class-Diagram-Aryan-Jadon.png

d)

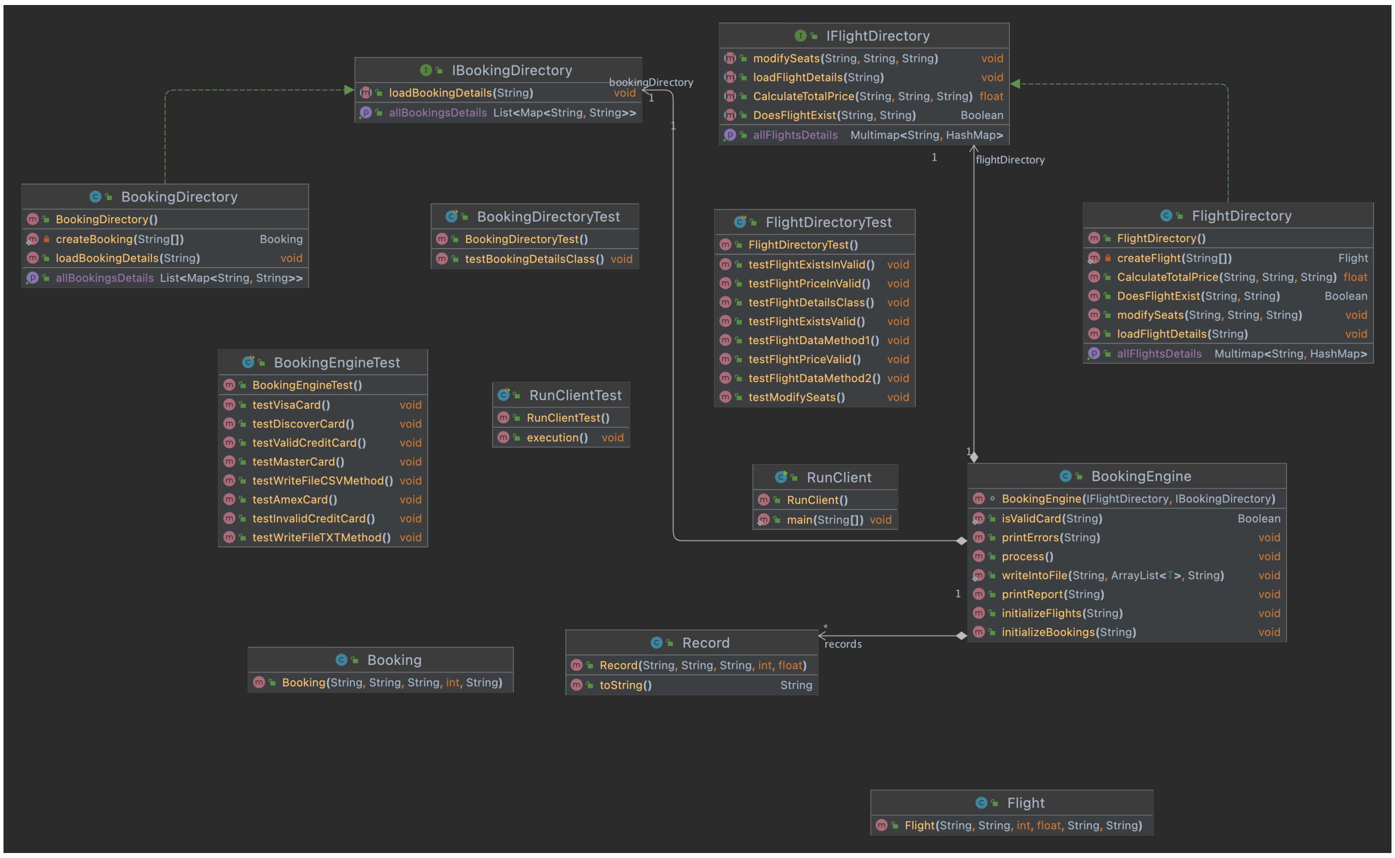


```
Payment Validation Completed
{SJ456=[{Category=Economy, Price=250.0, Departure
ategory=Economy, Price=300.0, Departure=San Jose,

Processing For BY110
Flight Validation Completed -- Processing Seats
Seats Validation Completed -- Processing Price
Payment Validation Completed
{SJ456=[{Category=Economy, Price=250.0, Departure
ategory=Economy, Price=300.0, Departure=San Jose,

Processing For SJ456
Flight Validation Completed -- Processing Seats
Seats Validation Completed -- Processing Price
Payment Validation Completed
{SJ456=[{Category=Economy, Price=250.0, Departure
ategory=Economy, Price=300.0, Departure=San Jose,

Processing For KL908
Flight Validation Failed

Processing For BY110
Flight Validation Completed -- Processing Seats
Seats Validation Completed -- Processing Price
Payment Validation Failed
```



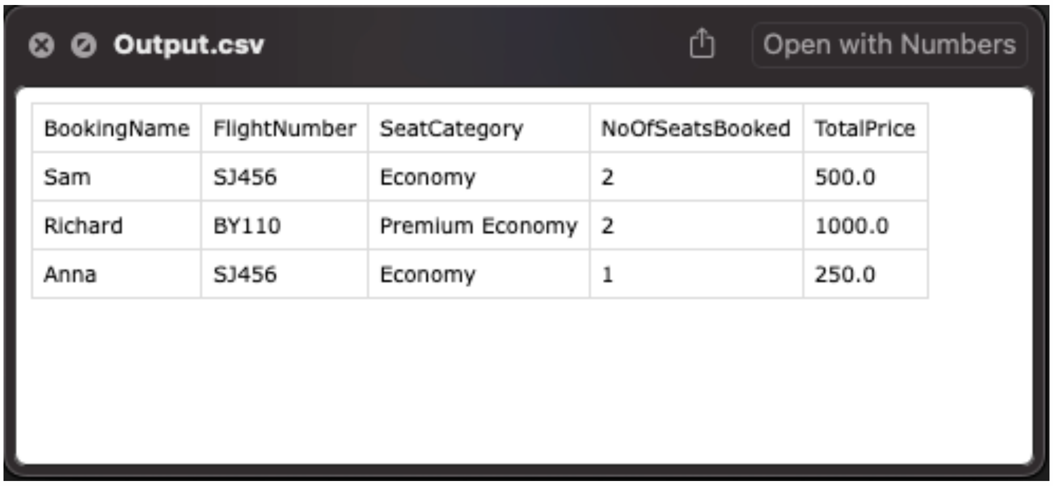
```
[INFO] ------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------
[INFO] Total time:  01:17 min
[INFO] Finished at: 2022-05-05T06:42:04-07:00
[INFO] ------------------------------------------------
➜  Java-Personal-Project git:(master) ✗
```

2. Class Diagram



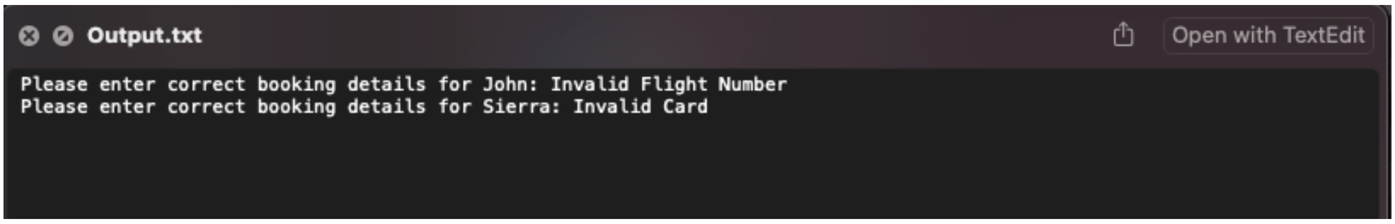
IBookingDirectory
loadBookingDetails(String) void
allBookingsDetails List<Map<String, String>>
bookingDirectory
IFlightDirectory
modifySeats(String, String, String) void
loadFlightDetails(String) void
CalculateTotalPrice(String, String, String) float
DoesFlightExist(String, String) Boolean
allFlightsDetails Multimap<String, HashMap>
flightDirectory
BookingDirectory
BookingDirectory()
createBooking(String[]) Booking
loadBookingDetails(String) void
allBookingsDetails List<Map<String, String>>
BookingDirectoryTest
BookingDirectoryTest()
testBookingDetailsClass() void
FlightDirectoryTest
FlightDirectoryTest()
testFlightExistsInValid() void
testFlightPriceInValid() void
testFlightDetailsClass() void
testFlightExistsValid() void
testFlightDataMethod1() void
testFlightPriceValid() void
testFlightDataMethod2() void
testModifySeats() void
FlightDirectory
FlightDirectory()
createFlight(String[]) Flight
CalculateTotalPrice(String, String, String) float
DoesFlightExist(String, String) Boolean
modifySeats(String, String, String) void
loadFlightDetails(String) void
allFlightsDetails Multimap<String, HashMap>
BookingEngineTest
BookingEngineTest()
testVisaCard() void
testDiscoverCard() void
testValidCreditCard() void
testMasterCard() void
testWriteFileCSVMethod() void
testAmexCard() void
testInvalidCreditCard() void
testWriteFileTXTMethod() void
RunClientTest
RunClientTest()
execution() void
RunClient
RunClient()
main(String[]) void
BookingEngine
BookingEngine(IFlightDirectory, IBookingDirectory)
isValidCard(String) Boolean
printErrors(String) void
process() void
writeIntoFile(String, ArrayList<T>, String) void
printReport(String) void
initializeFlights(String) void
initializeBookings(String) void
Booking
Booking(String, String, String, int, String)
Record
Record(String, String, String, int, float)
toString() String
records
Flight
Flight(String, String, int, float, String, String)
1
1
1
1
1
*

3. Output Files



BookingName	FlightNumber	SeatCategory	NoOfSeatsBooked	TotalPrice
Sam	SJ456	Economy	2	500.0
Richard	BY110	Premium Economy	2	1000.0
Anna	SJ456	Economy	1	250.0

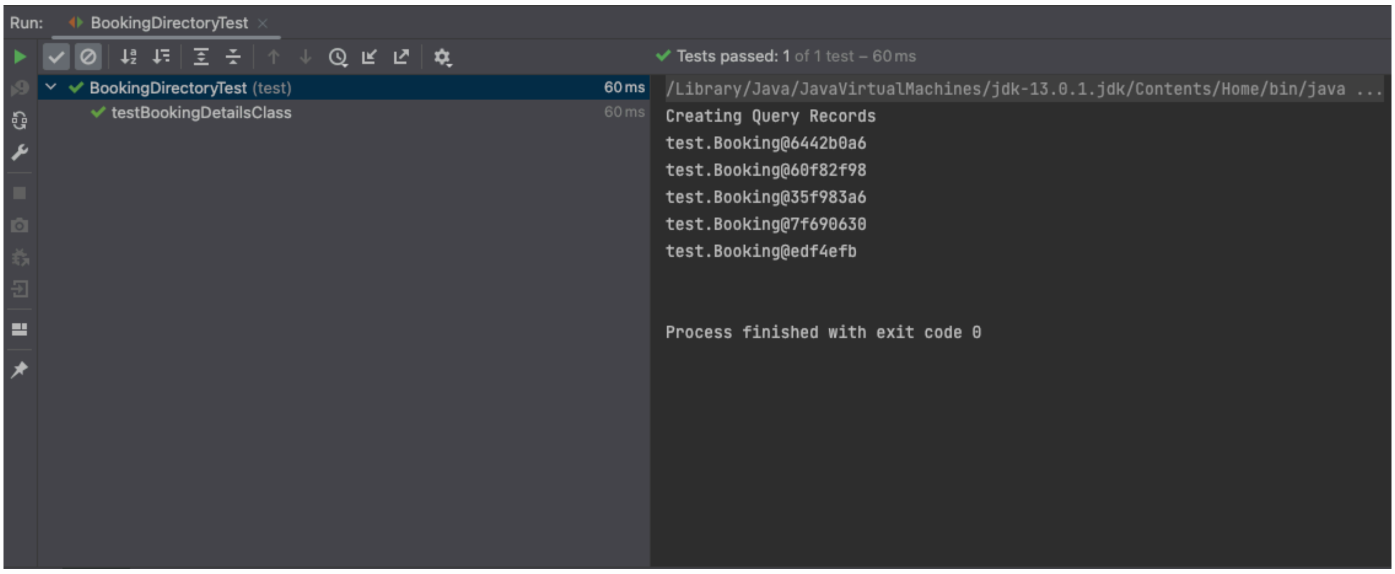
Output CSV File



```
Please enter correct booking details for John: Invalid Flight Number
Please enter correct booking details for Sierra: Invalid Card
```

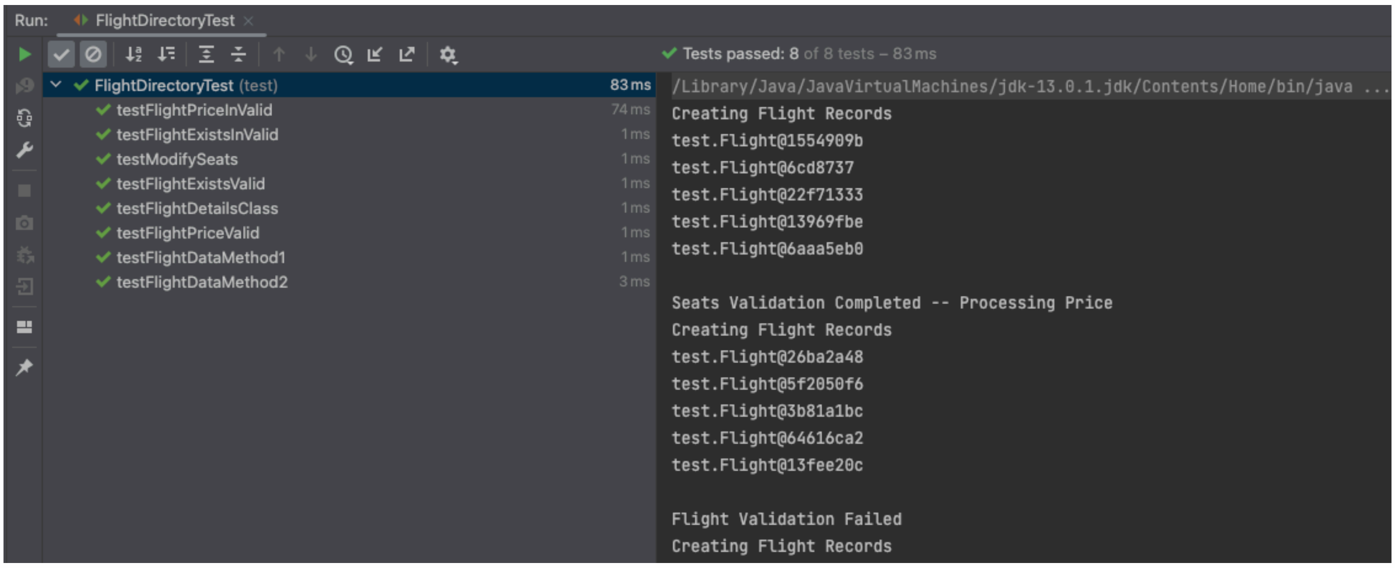
Output Txt File

4. Unit Test Cases



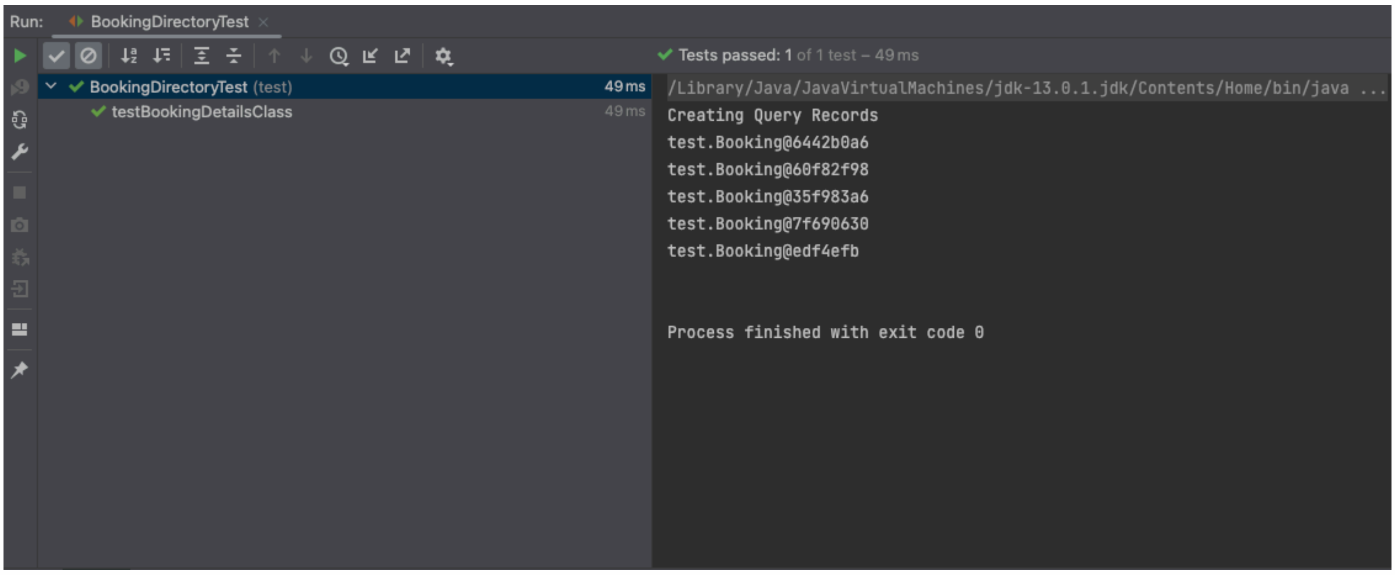
```
BookingDirectoryTest (test)    60 ms
  testBookingDetailsClass      60 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java ...
Creating Query Records
test.Booking@6442b0a6
test.Booking@60f82f98
test.Booking@35f983a6
test.Booking@7f690630
test.Booking@edf4efb


Process finished with exit code 0
```



```
FlightDirectoryTest (test)     83 ms
  testFlightPriceInValid       74 ms
  testFlightExistsInValid       1 ms
  testModifySeats               1 ms
  testFlightExistsValid         1 ms
  testFlightDetailsClass        1 ms
  testFlightPriceValid          1 ms
  testFlightDataMethod1         1 ms
  testFlightDataMethod2         3 ms
```

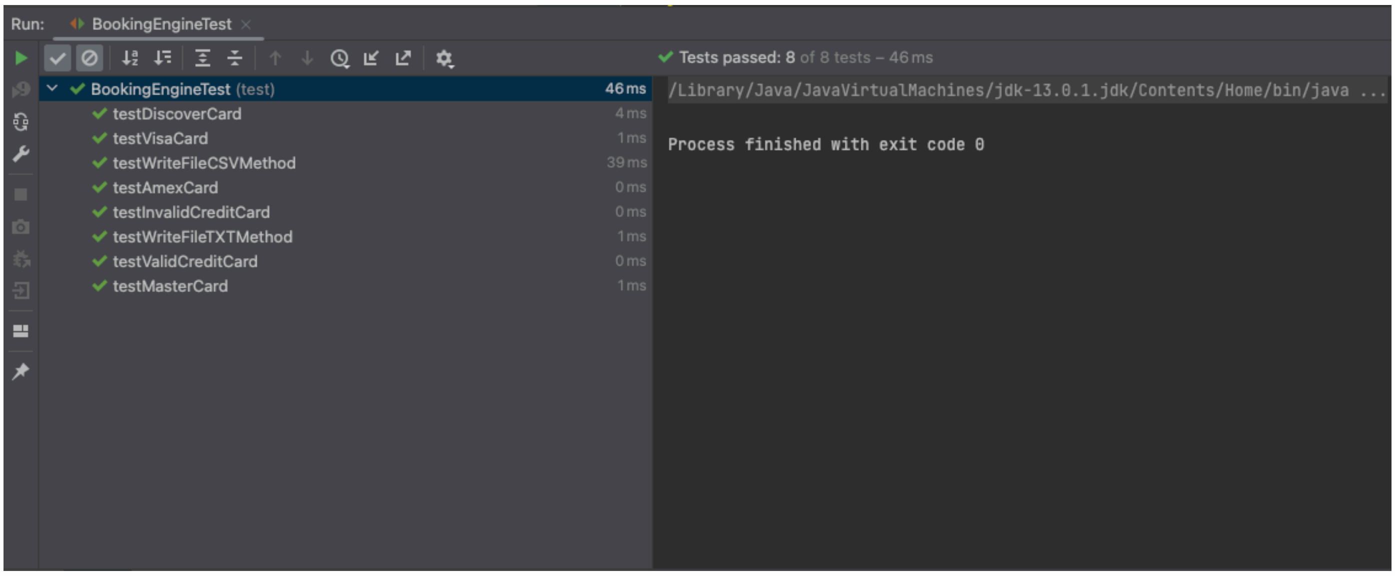
```
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java ...
Creating Flight Records
test.Flight@1554909b
test.Flight@6cd8737
test.Flight@22f71333
test.Flight@13969fbe
test.Flight@6aaa5eb0

Seats Validation Completed -- Processing Price
Creating Flight Records
test.Flight@26ba2a48
test.Flight@5f2050f6
test.Flight@3b81a1bc
test.Flight@64616ca2
test.Flight@13fee20c

Flight Validation Failed
Creating Flight Records
```



Tests passed: 1 of 1 test – 49 ms

BookingDirectoryTest (test) 49 ms
testBookingDetailsClass 49 ms

```
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java ...
Creating Query Records
test.Booking@6442b0a6
test.Booking@60f82f98
test.Booking@35f983a6
test.Booking@7f690630
test.Booking@edf4efb


Process finished with exit code 0
```



Tests passed: 8 of 8 tests – 46 ms

BookingEngineTest (test) 46 ms
testDiscoverCard 4 ms
testVisaCard 1 ms
testWriteFileCSVMethod 39 ms
testAmexCard 0 ms
testInvalidCreditCard 0 ms
testWriteFileTXTMethod 1 ms
testValidCreditCard 0 ms
testMasterCard 1 ms

```
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

5. Problem Solved

1. The primary problem is to get the records from files for both Flight Information and Booking , and store them into a dataset, which it needs to ensure that there's only one instance for entire data access.
2. The secondary problem is to follow-best principles and design patterns, validate if the request is valid(requested flight exists, the number of seats is available for the category, user's payment card is valid based on known rules for different credit card types)

6. Design Patterns

Iterator Pattern

Iterator is a behavioral design pattern that lets you traverse elements of a collection without exposing its underlying representation, Iterator Pattern is used to get a way to access the elements of a collection of objects in a sequential manner. Booking and Flight Details are loaded in Hash Map to perform operations on the application.

Interfaces and methods are created so that the data are accessed from Hashmaps of flight and booking input. files and once the transactions are completed, the output files are updated.

Pros of Using Iterator Pattern

- *Single Responsibility Principle*. You can clean up the client code and the collections by extracting bulky traversal algorithms into separate classes.
- *Open/Closed Principle*. You can implement new types of collections and iterators and pass them to existing code without breaking anything.
- You can iterate over the same collection in parallel because each iterator object contains its own iteration state.
- For the same reason, you can delay an iteration and continue it when needed.

Cons of Using Iterator Pattern

- Applying the pattern can be an overkill if your app only works with simple collections.
- Using an iterator may be less efficient than going through elements of some specialized collections directly.

Repository Design Pattern

The Repository pattern is a planned and documented way of working with a data source or multiple data sources. IFlightDirectory and IBookingDirectory are created to fetch the data from input files that contains flight and booking details so that if we want to make changes in the code then it can be modified in one place and it doesn't effect the other code. Communication between data access and functionality of application si being done through interfaces.

Pros of Repository pattern

- Database access logic and domain logic can be tested separately with this pattern.
- Domain-driven development is easier.
- Clean, maintainable, and reusable code
- It reduces redundancy of code; generic repository is better solution than normal repository pattern to reduce code duplication.
- With this pattern it is easy to maintain the centralized data access logic.
- DRY (Don’t Repeat Yourself) design, the code to query and fetch data from data source, commands for updates (update, deletes) are not repeated.
- With using the Repository design pattern, you can hide the details of how the data is eventually stored or retrieved to and from the data store (data store can be a database, an xml file, etc)

Cons of Repository pattern

- An extra layer of abstraction - Due another layer of abstraction a certain level of complexity making it an overkill for small applications.
- With repository pattern require to create a new repository for each entity.
- The repository pattern does not decouple the data access from the data store, yes that is correct and it breaks here clean codding approach.

Visitor Design Pattern

Visitor is a behavioral design pattern that lets you separate algorithms from the objects on which they operate. While booking a ticket on the flight application system, we check if flight exists and then look for the category of flights and number of seats available. Once all these criteria are met then payment validation is perfomed based on different bank providers card and booking gets confirmed. Flight Ticket Price calculation is performed and the input data is updated based on the booking transaction.

Pros of Using Vistor Design Pattern

- *Open/Closed Principle*. You can introduce a new behavior that can work with objects of different classes without changing these classes.
- *Single Responsibility Principle*. You can move multiple versions of the same behavior into the same class.
- A visitor object can accumulate some useful information while working with various objects. This might be handy when you want to traverse some complex object structure, such as an object tree, and apply the visitor to each object of this structure.

Cons of Using Visitor Design Pattern

- You need to update all visitors each time a class gets added to or removed from the element hierarchy.
- Visitors might lack the necessary access to the private fields and methods of the elements that they're supposed to work with.