Template Method PatternSED 2018

\*\* Comments and suggestions welcome (pls be kind) \*\*

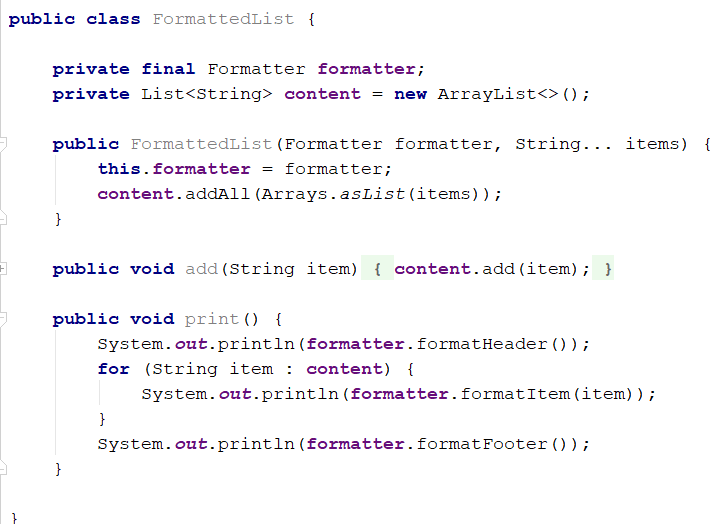
Question 1

Part A

i) Template Method Pattern

ii) Strategy Pattern

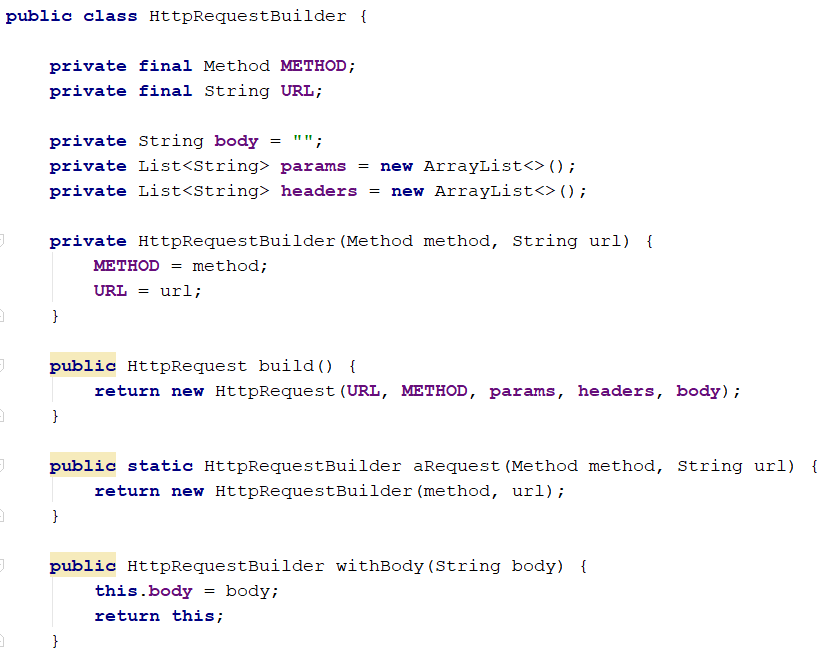
iii)



Part B

i) Builder Pattern

ii)

x

* URL and Method are necessary parameters so they should be required in the builder constructor
* Params and Header can be added one at a time

(Using the builder above)

iii) aRequest(Method.GET, "<http://www.imperial.ac.uk>").build();

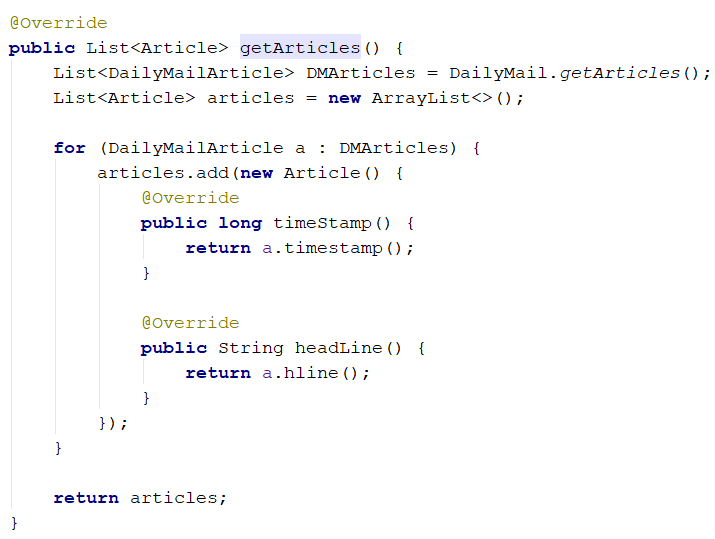
iv) aRequest(Method.POST, "<http://www.imperial.ac.uk>/220").withBody(“mark=100”).withHeaders(Arrays.asList(“Date=02-05-2018”)).build();

NPart C

i) Adapter Pattern

ii) Hexagonal Architecture / Ports and Adapters

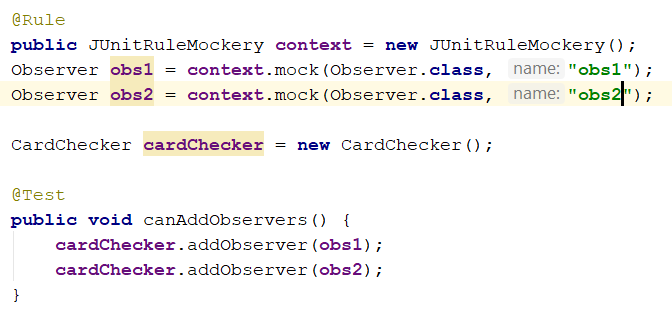
iii)

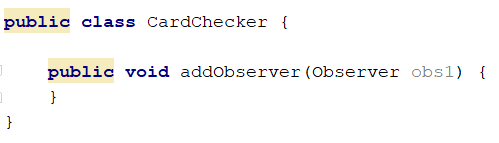


Question 2

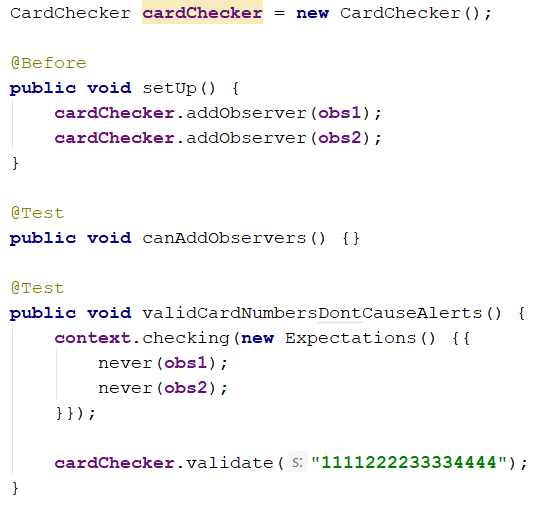
Part A

i)

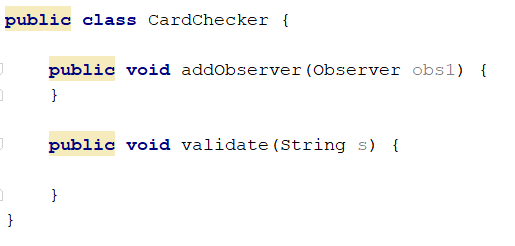




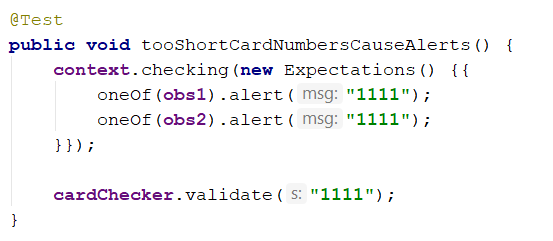
ii)

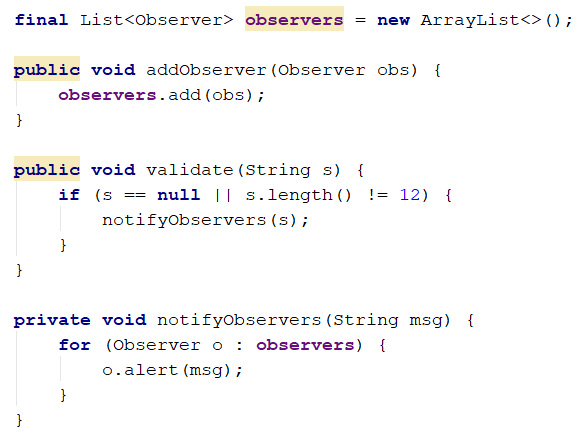


(16 digits - fixed later)



iii)





Part B

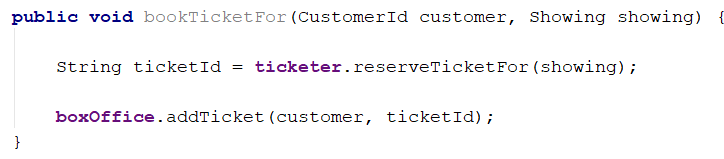
i) boxOffice.getCustomerDatabase().getCustomer(customer).getTickets().add(ticketId);

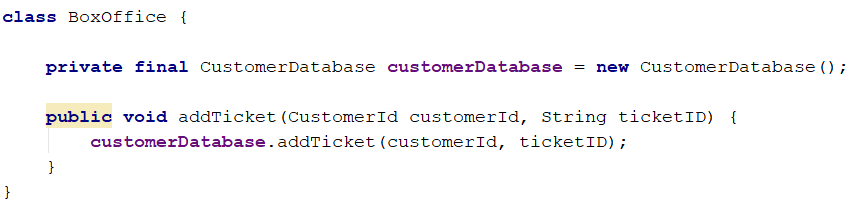
This is a ‘Train Wreck’ and shouldn’t occur if the Law of Demeter is followed

ii)

Not following the Law of Demeter risk making code more fragile. By allowing classes to interact with those ‘far’ away we introduce long reaching dependencies and tighter coupling. Changing a class can cause another unit to stop working.

iii)





And so on ….

iv)

“Tell, Don’t Ask”

Classes should tell other classes to do something rather than asking for the information to do it themselves