FA0: Zur Realisierung des Port-Konzeptes kann das Strategy-Pattern angewandt werden.
Bereits im Konstruktor der Klasse "HotelRetrieval" darf eine Referenz zu einer Caching-Implementation übegeben werden, welche
dann das eigentlich Caching übernimmt und in HotelRetrieval nur als Attribut verwendet wird.

FA1: Beim Aufruf des Interfaces "HotelSuche" gibt es folgende Reihenfolge, die stets einzuhalten ist,
da SQL- bzw. Datenbankoperationen sonst nicht korrekt ausgeführt werden. Zusätzlich wird die Methode closeSession()
zum Interface hinzugefügt.

1: openSession()
2 bis n-1: getHotelByXYZ(...)
n: closeSession()

FA2: Das Interface "Caching" sollte mindestens eine Methode zum auslesen des Caches anbieten:
Hotel getCachedResult()

FA3: Da das in HotelRetrieval verwendete Caching-Objekt nicht zwingend valide bzw. non-null sein muss,
können bei aufrufen dieses Objektes NullPointerExceptions auftreten.
Hier kann entweder vor jedem Aufruf ein null-check vorgenommen werden, oder bei der Initialisierung der Klasse direkt überprüft werden,
ob das Caching-Objekt == null ist. Das Ergebnis wird dann in einem weiteren Attribut abgespeichert.
Weiterhin kann z.B das Proxy-Pattern verwendet werden, welches eine Proxy-Klasse für HotelRetrieval implementiert, das Logging und exceptions übernimmt
und die eigentliche Logik unberührt in HotelRetrieval belässt.

FA4: Das Logging wird ebenfalls mittels des Strategy-Patterns implementiert und vom Proxy verwendet.

FA5: Die exportierte Jar-Datei des Buchungssystems wird dann lediglich in ein weiteres Modul "Buchungsclient" importiert
(analog zur Verwendung der JDBC-Bibliothek). Daraufhin können alle als öffentlich deklarierte Klassen im Buchungsclient aufgerufen und benutzt werden.