Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Number	COE692
Course Title	Software Design and Architecture
Semester/Year	Winter Semester 2024
Instructor	Dr. Faezeh Ensan
TA Name	Bita

Lab/Tutorial Report No.	Lab 5

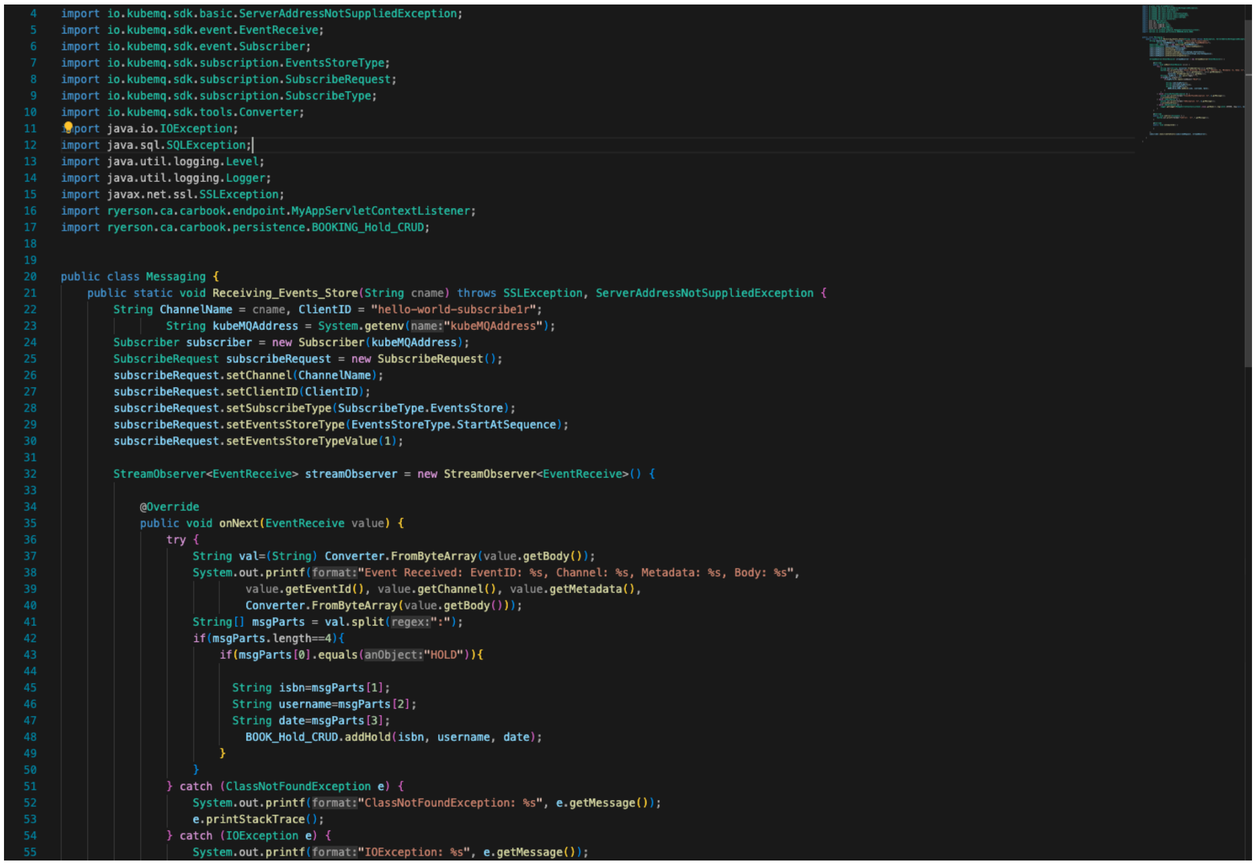
Report Title	Car Rental Management System

Section No.	07 & 02
Submission Date	Friday Apr 12, 2024
Due Date	Friday Apr 12, 2024

Student Name	Student ID	Signature*
Hamza Malik	501112545	HM
Kevin Bhatt	501093104	KM

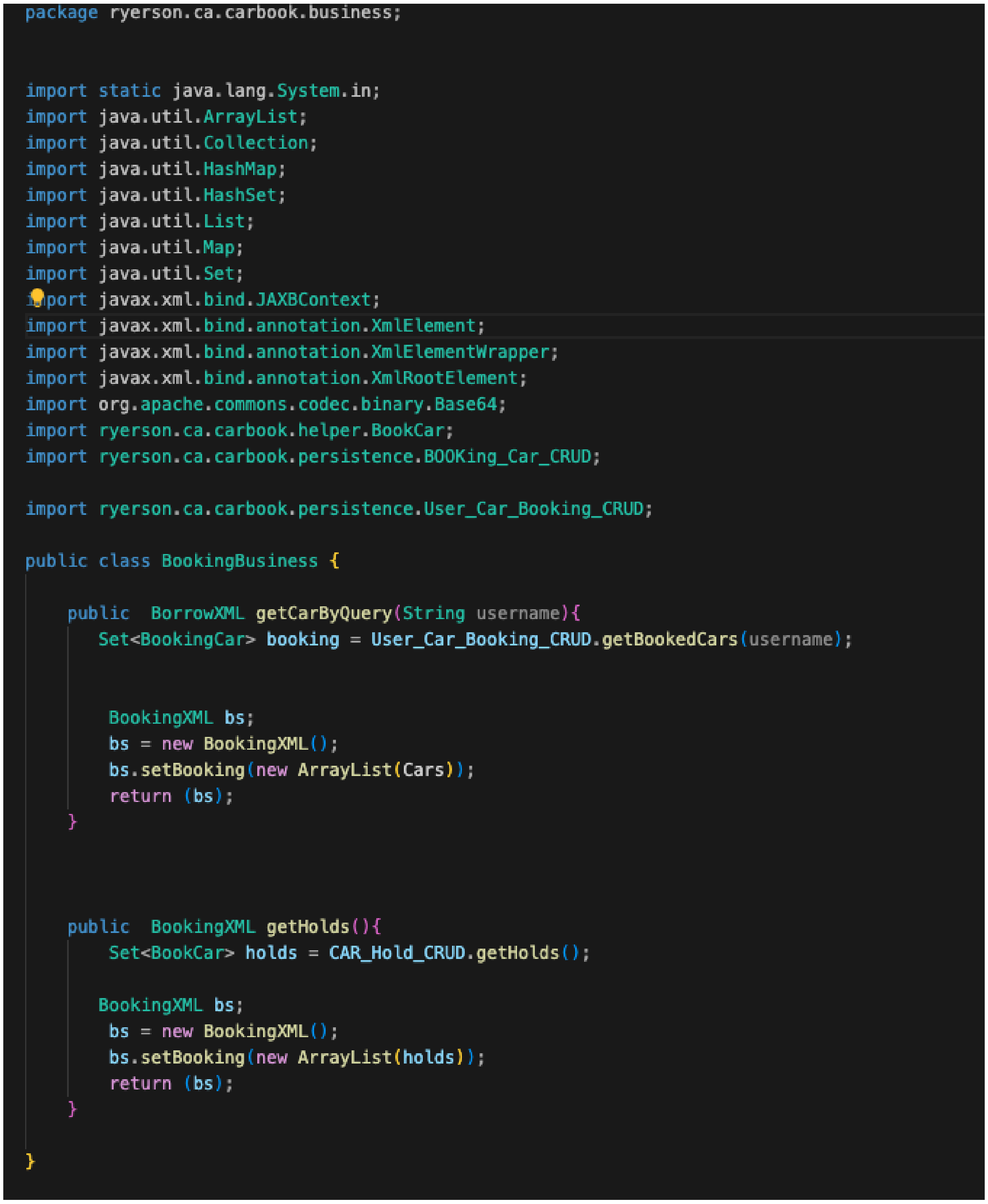
In this lab, we worked with two microservices, the SearchCar MicroService and the BookCar MicroService. The SearchCar MicroService is responsible for handling user requests to search for available vehicles based on various criteria such as date and time of rental, location, car model, size, and specific vehicle identification number (VIN) if known. It processes the input and retrieves matching vehicles from the inventory, ensuring a seamless user experience. The SearchCar MicroService is dependent on the BookCar MicroService to provide the necessary booking functionality once a user has selected a vehicle. The BookCar MicroService, on the other hand, is responsible for handling all aspects of the booking process, from availability checks to reservation confirmation and modification. It integrates with other services, such as inventory management, to access real-time availability data and payment processing for financial transactions.

We used a MySQL database to store information about the vehicles available for rent. The database schema consisted of several tables, including one for vehicle information, one for vehicle availability, and one for vehicle reservations. The vehicle information table contained details about each vehicle, such as the make, model, and year. The availability table contained information about when each vehicle was available for rent, and the reservations table stored information about current and past reservations. The SearchCar MicroService retrieved data from the vehicle information and availability tables to provide search results to the user, while the BookCar MicroService updated the reservations table to confirm and manage bookings.



```java
import io.kubemq.sdk.basic.ServerAddressNotSuppliedException;
import io.kubemq.sdk.event.EventReceive;
import io.kubemq.sdk.event.Subscriber;
import io.kubemq.sdk.subscription.EventsStoreType;
import io.kubemq.sdk.subscription.SubscribeRequest;
import io.kubemq.sdk.subscription.SubscribeType;
import io.kubemq.sdk.tools.Converter;
import java.io.IOException;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.ssl.SSLException;
import ryerson.ca.carbook.endpoint.MyAppServletContextListener;
import ryerson.ca.carbook.persistence.BOOKING_Hold_CRUD;


public class Messaging {
    public static void Receiving_Events_Store(String cname) throws SSLException, ServerAddressNotSuppliedException {
        String ChannelName = cname, ClientID = "hello-world-subscribe1r";
                String kubeMQAddress = System.getenv(name:"kubeMQAddress");
        Subscriber subscriber = new Subscriber(kubeMQAddress);
        SubscribeRequest subscribeRequest = new SubscribeRequest();
        subscribeRequest.setChannel(ChannelName);
        subscribeRequest.setClientID(ClientID);
        subscribeRequest.setSubscribeType(SubscribeType.EventsStore);
        subscribeRequest.setEventsStoreType(EventsStoreType.StartAtSequence);
        subscribeRequest.setEventsStoreTypeValue(1);

        StreamObserver<EventReceive> streamObserver = new StreamObserver<EventReceive>() {

            @Override
            public void onNext(EventReceive value) {
                try {
                    String val=(String) Converter.FromByteArray(value.getBody());
                    System.out.printf(format:"Event Received: EventID: %s, Channel: %s, Metadata: %s, Body: %s",
                            value.getEventId(), value.getChannel(), value.getMetadata(),
                            Converter.FromByteArray(value.getBody()));
                    String[] msgParts = val.split(regex:":");
                    if(msgParts.length==4){
                        if(msgParts[0].equals(anObject:"HOLD")){

                          String isbn=msgParts[1];
                          String username=msgParts[2];
                          String date=msgParts[3];
                            BOOK_Hold_CRUD.addHold(isbn, username, date);
                        }
                    }
                } catch (ClassNotFoundException e) {
                    System.out.printf(format:"ClassNotFoundException: %s", e.getMessage());
                    e.printStackTrace();
                } catch (IOException e) {
                    System.out.printf(format:"IOException: %s", e.getMessage());
```

The code is the Messaging class. This method is used to receive events from a Kubernetes Message Queue (Kubemq) with a specified channel name and client ID. The subscribe type is set to EventsStore and the events store type is set to StartAtSequence. When an event is received, the method converts the event body from bytes to a string and checks if it contains four parts separated by colons. If it does, the method further checks if the first part is "HOLD". If it is, the method extracts the ISBN, username, and date from the second, third, and fourth parts, respectively, and adds a hold to the database using the CAR_BOOKING_CRUD.addHold method. If any exceptions occur during the processing, they are caught and printed to the console. This method is used to handle holds in the system and ensure that they are properly recorded in the database.



```
package ryerson.ca.carbook.business;


import static java.lang.System.in;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;
import org.apache.commons.codec.binary.Base64;
import ryerson.ca.carbook.helper.BookCar;
import ryerson.ca.carbook.persistence.BOOKing_Car_CRUD;

import ryerson.ca.carbook.persistence.User_Car_Booking_CRUD;

public class BookingBusiness {

    public  BorrowXML getCarByQuery(String username){
       Set<BookingCar> booking = User_Car_Booking_CRUD.getBookedCars(username);


        BookingXML bs;
        bs = new BookingXML();
        bs.setBooking(new ArrayList(Cars));
        return (bs);
    }



    public  BookingXML getHolds(){
        Set<BookCar> holds = CAR_Hold_CRUD.getHolds();

       BookingXML bs;
        bs = new BookingXML();
        bs.setBooking(new ArrayList(holds));
        return (bs);
    }

}
```

This implementation uses the User_Car_Booking_CRUD class and has a method called getBookedCars that takes a username as a parameter and returns a set of borrowed cars for that user. The getCarByQuery method will call this method and then create a BookingXML object with the list of booked cars.