**Exercise 1: Code Review**

Imagine the following class in a Java code base. It contains no syntactic errors and is compilable. Your task is to perform a fictitious code review. What comments do you have?

@Singleton  
*public class* AlarmScheduler {  
  
 *private static final* Logger LOG = LoggerFactory.getLogger(BookingRepository.*class*);  
  
 *private* BookingRepository repository;  
  
 *public* AlarmScheduler() {  
 *// ejb* }  
  
 @Inject  
 *public* AlarmScheduler(BookingRepository repository) {  
 *this*.repository = repository;  
 }  
  
 */\*\*  
 \* runs every 1 hour  
 \*/* @Schedule(minute = "\*", hour = "\*")  
 *public void* run() {  
 List<Booking> bookings = repository.selectByState("DELETED");  
 *for* (*int* i = 0; i < bookings.size() - 1; i++) {  
 repository.delete(bookings.get(i).getId());  
 LOG.info("Booking with ID {} is deleted.", bookings.get(i));  
 }  
 }  
  
}

* Singleton cannot have public constructor, probably in this case the class should not be singleton
* Wrong class in the logger BookingRepository.*class instead of* AlarmScheduler
* According the comment it should be triggered every hour but actually it is called every minute
* There is bad indexing in the for loop – in this case the last booking will not be deleted
* It would be better to use foreach instead of for loop (or even better would be delete bookings in bulk operation – deleteAll)

**Exercise 2: Multiple Choice**

Below you will see several suggestions for an answer. Please tick the correct answer.

2.1 Does the Java annotation "@ApplicationScoped" require a default constructor?

* Yes
* No

2.2 What does the output of the following Java program look like?

*class* output {  
 *public static void* main(String args[])  
 {  
 *double* a, b,c;  
 a = 3.0/0; //Infinity  
 b = 0/4.0; //0.0  
 c=0/0.0; //NaN  
  
 System.out.println(a);  
 System.out.println(b);  
 System.out.println(c);  
 }  
}

* NaN
* Infinity
* 0.0
* all of the mentioned

2.3 Which of the following is not an OOP concept in Java?

* Polymorphism
* Inheritance
* Compilation
* Encapsulation

2.4 What does the output of the following Java program look like?

*class* output  
{  
 *public static void* main(String args[])  
 {  
 *int* x;  
 x = 5;  
 {  
 *int* y = 6;  
 System.out.print(x + " " + y);  
 }  
 System.out.println(x + " " + y);  
 }  
}

* Compile error as y is not defined on the last line when it tries to output that
* Runtime error
* 5 6 5 6
* 5 6 5

2.5 Which error occurs in the following Java code?

*byte* b = 50;  
b = b \* 50;

* b cannot contain value 50
* b cannot contain value 100, limited by its range
* No error in this code
* operator has converted b \* 50 into int, which can not be converted to byte without casting

2.6 What is the purpose of the @javax.persistence.Version column?

* JPA will throw a PessimisticLockException when the database entry has changed since last read and before update is performed
* JPA will throw an OptimisticLockException when the database entry has changed since last read and before update is performed
* Serves as documentation only
* The EntityManager will refresh the fetched entity automatically to ensure that the newest version of the entity is always available

2.7 Which CDI scope should be preferred for a Repository/Data Access Object (DAO) class?

* @ApplicationScoped
* @SessionScoped
* @ConversationScoped
* @RequestScoped
* @Dependent

**Exercise 3: Theoretical implementation of a Booking Update method**

Imagine a multi-tier Java architecture, where you have to add a new method to an existing BookingEndpoint. Write in **pseudo code** **below** an updateBookingState(Id bookingId, String state, Integer version) method that returns a Booking object. Make assumptions as you see fit. Use the ER model below, which might help you.

Graphical user interface, application

Description automatically generated

Public class BookingEndpoint {

@Autowired BookingRepository bookingRepository;

updateBookingState(Id bookingId, String state, Integer version) {

Optional<Booking> optBooking = bookingRepository.findById(bookingId);

if (optBooking.isPresent()) {

Booking booking = optBooking.get();

if (booking.getVersion > version) {

throw new Exception(“You are using the older version.”);

}

booking.setState(state);

booking.setVersion(version)

Booking created = bookingRepository.save(booking);

return created;

} else {

throw new NotFoundException(“The booking with id {} not found.”, bookingId);

}

}

}

**Exercise 4: Practical implementation of a Booking Update method**

Please take a look at the **project shared with you** **(contained in the folder “code-review-exercise-for-interview”)** and, using the information you have been given, try to implement the complete functionality stub for a REST API endpoint that updates a booking based on its booking ID, the state and the provided version, setting the BookingState to "ACCEPTED". Make assumptions as you see fit.

**IMPORTANT**

Please share your notes and your coding exercise solution to: [kevin.murray@hlag.com](mailto:kevin.murray@hlag.com) or upload it to the git repository shared with you. Please make sure that your amended files are included as an attachment!

Many thanks