CS544
Enterprise Architecture
Midterm December 2016

Name____

Student ID ____

NOTE: This material is private and confidential. It is the property of MUM and is not to be disseminated.

1. [10 points] **Circle w**hich of the following is TRUE/FALSE concerning Spring Inversion of Control/Dependency Injection:

   T F Only Managed Beans can be injected in Spring, a POJO or JavaBean cannot.

   EXPLAIN:____

   T F **@Autowired works only on interfaces. It cannot work directly on classes.**

   EXPLAIN:____

   T F In practice, IoC container is not exactly the same as Dependency Injection as it involves a discovery step concerning the dependency.

   EXPLAIN:____

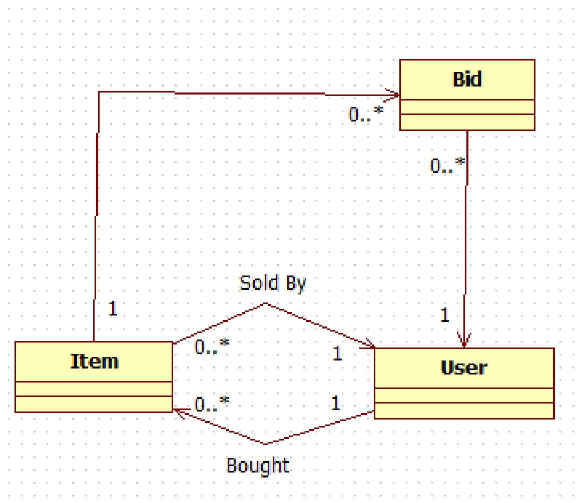
   T F A domain object needed in a @Service class is usually a good candidate for Dependency Injection

   EXPLAIN:____

   T F In Spring, DI can be done through either XML or through Annotations. They are mutually exclusive. That means, if you use XML for DI for one bean, they you should use it, exclusively for all beans.

   EXPLAIN:____

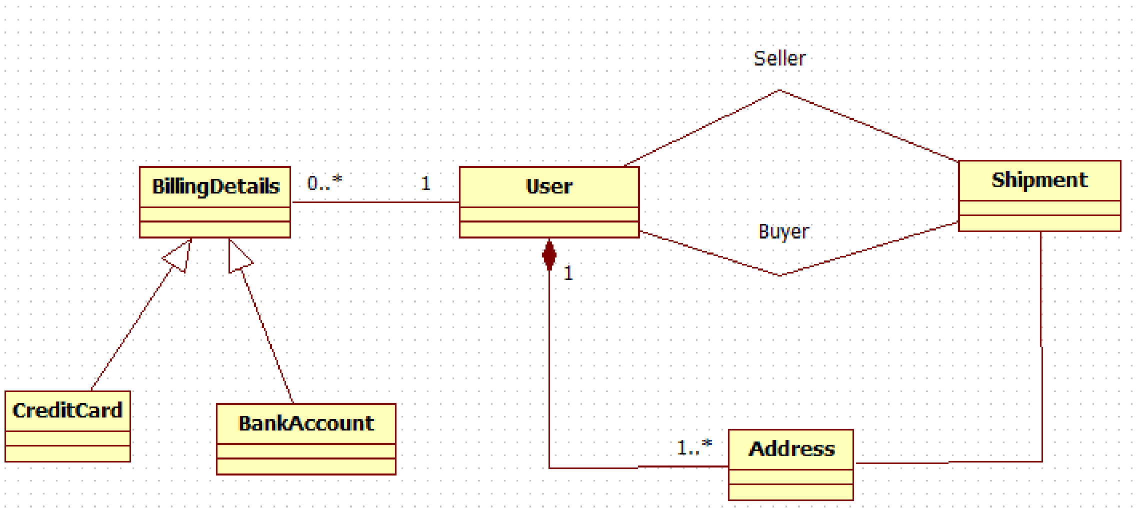
2. [15 points] For the following relationships implement a SubSelect that fetches all items with their corresponding collection of bids.

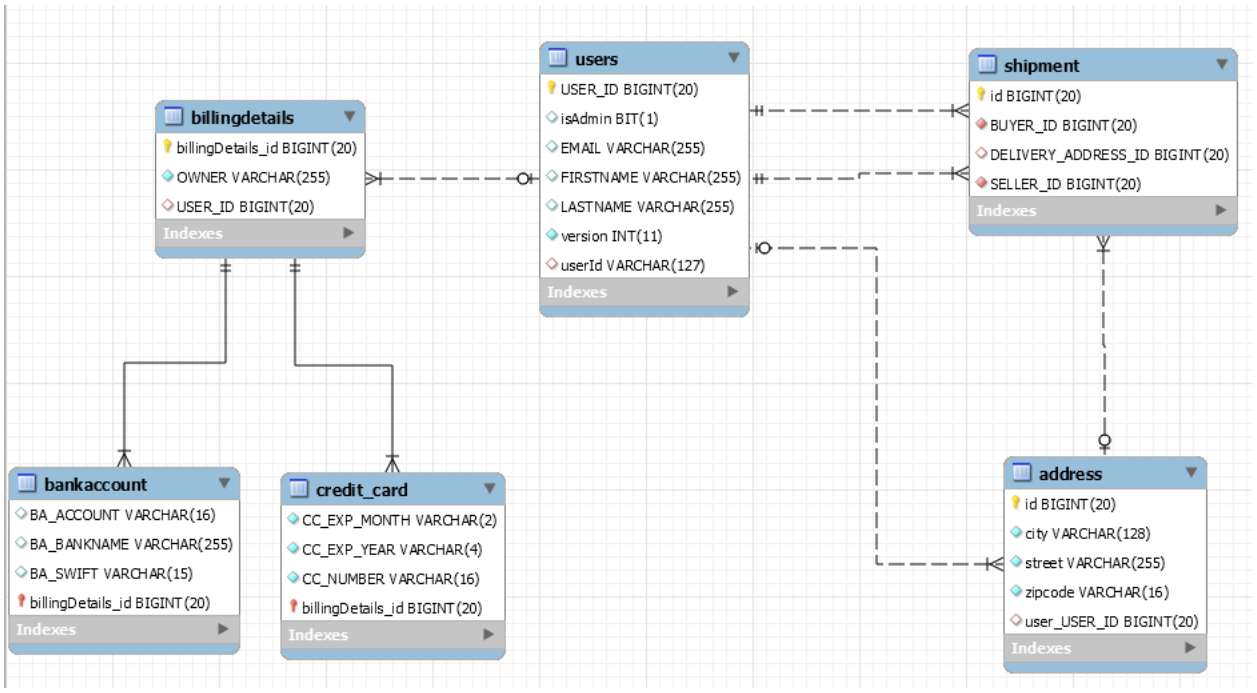


What performance problem[s] does the SubSelect fetch address?
How does it work? – Explain the "algorithm" based on a universe of 10 Items each with a collection of 5-10 Bids.
Compare it to Join Fetch.

3. [15 points] The Core J2EE DAO pattern is fundamental to a well-organized ORM application.
   Explain the pattern, what is for, how it works. Include in the explanation, the sequence of interactions between the user, a DAO and the database. Include an explanation [& UML diagram] of the Generic DAO design.
   Be specific, give examples.

4. [20 points] Annotate the Domain Objects based on the Domain Model and Entity Relationship Diagram provided. NOTE: All the Domain Objects are not listed. All the fields are not listed. Only annotate the objects and fields that are listed.





User.java

```
public class User {

private Long id = null;

private String firstName;

private String lastName;

private String email;

private boolean admin = false;

private Set<BillingDetails> billingDetails;

private Set<Address> addresses;

private Set<Shipment> buyShipments;

private Set<Shipment> sellShipments;
```

Shipment.java

```
public class Shipment {

   private Long id = null;

   private Address deliveryAddress;

   private User buyer;

   private User seller;
```

Address.java

```
public class Address implements Serializable {

public class Address implements Serializable {

private Long id = null;

private String street;

private String zipcode;

private String city;

private User  user;
```

BillingDetails.java

```java
public abstract class BillingDetails  {


    private Long id = null;

    private String owner;


    private User user;
```

BankAccount.java

```java
public class BankAccount extends BillingDetails {


        private String account;

        private String bankname;


        private String swift;
```

5. [10 points] Explain the concept of locking. Include the definition of the two strategies covered in class. Give the details of Version-Based Optimistic Concurrency [Locking], how it relates to isolation levels, how it is implemented in JPA.

6. [15 points] Implement a JQPL query that looks up a User by email who bought an Item with a shipping address that has a specific zip code. [Reference UML in problem #4]
For instance:

Find User who has an email address of JohnDoe@mail.com who bought an Item that was shipped to an address with a zip code equal to 52556
OR
Find User who has an email address of JBean@post.com who bought an Item that was shipped to an address with a zip code equal to 12345.

The Query should be a parameterized query. Also identify all the classes in the specific packages that need to be modified to adhere to the N-Tier architecture convention.