Full Name: **Student ID:** **CS544 Midterm - April 2020**

Note Write your name on all pages of exam (including extra sheets of paper given to you). *Please make sure you read all questions. There are 4 pages of questions. You can answer on exam paper.*

1) [10 points] – List 5 different tiers/layers of **Service Oriented Architecture**. No explanation necessary.

2) [6 points] – Explain the difference between "**field**" vs. "**property**" accesses in Hibernate?

3) [4 points] – What are two main advantages of using an ORM (versus using plain database calls)?

4) [10 points] – Explain the difference between implicit vs. explicit updates in Hibernate?

5) [10 points] – Explain the **Single-Table** inheritance strategy in JPA?

6) [10 points] – Name two advantages and two disadvantages of using the **Joined-Table** inheritance strategy. Provide a little explanation so that I know that you understand it.

7) [10 points] – Under what situations Hibernate flushes the cache and writes all updates to DB? Name 3.

8) [10 points] – Is this the proper way to map a bi-directional mapping in JPA? Explain.

```
@Entity
public class Person {
  @Id
  @GeneratedValue
  private int id;
  private String firstname;
  private String lastname;
  @OneToMany
  @JoinColumn(name="person_id")
  private List<Car> cars =
            new ArrayList();
```

```
@Entity
public class Car {
  @Id
  @GeneratedValue
  private int id;
  private short year;
  private String model;
  private String maker;
  @ManyToOne
  @JoinColumn(name="owner_id")
  private Person owner;
```

9) [10 points] – Given the requirements of next questions (DB and Java design), write an HQL query to find out:

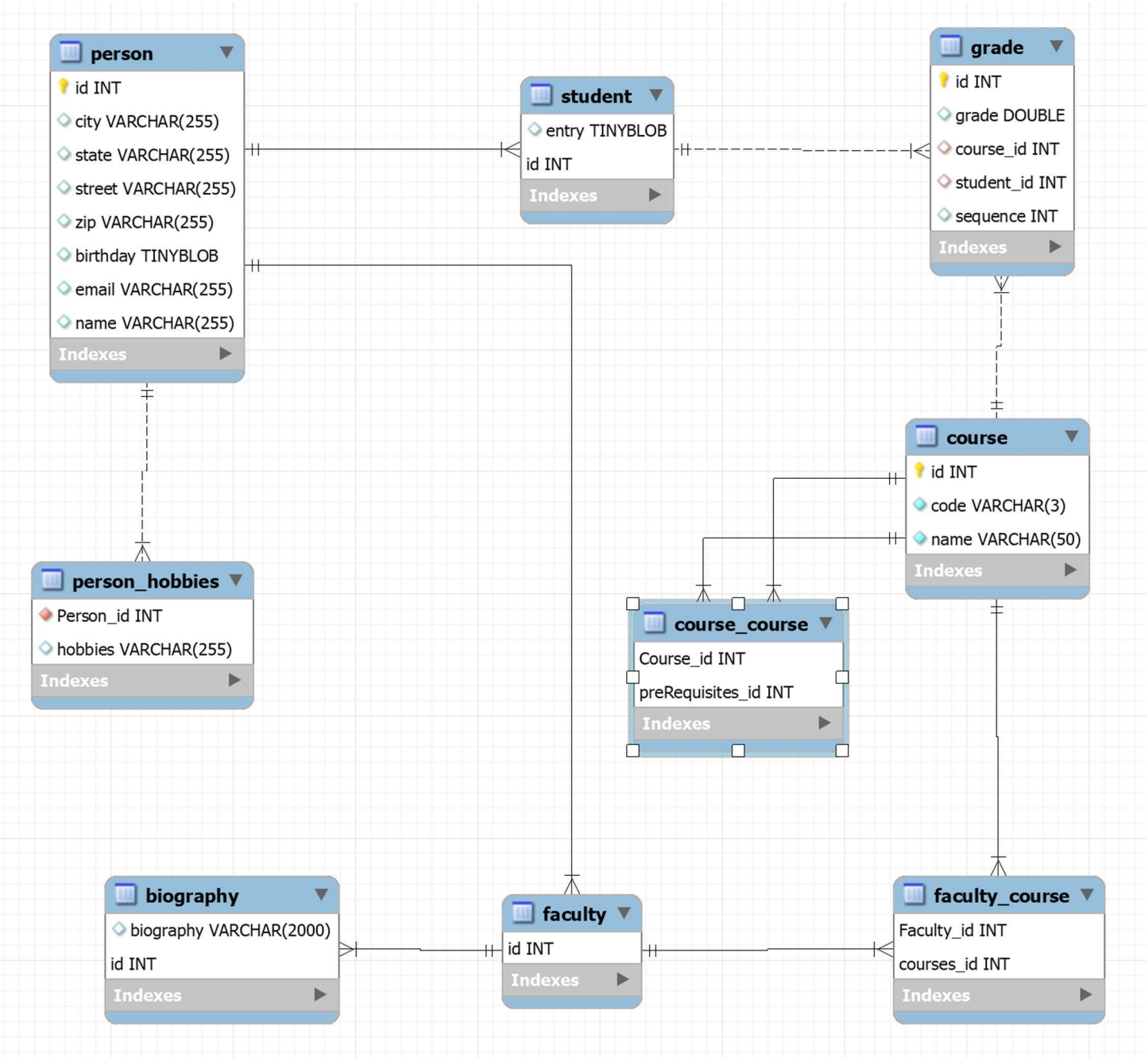
   a) List of students who have taken course with code 544 and have a grade of 3.8 or better

   b) List of faculty whose biography is more than 1000 characters

List<Student> students = session.CreateQuery("Select Student From Gradee g Where g.Course.code = 544 and g.grade >= 3.8").list()

List<Faculity> faculities = ("Select Faculity From bioagraphy b Where b. biography.length > 1000").list()

10) [40 points] – Given the following database ER (Entity Relationship) diagram:



Annotate the following classes with JPA annotations to map them to the above DB.

Note: Pay attention to relationships in Java and where each attribute is potentially going to be mapped to

Use field access.

```
@Embedable
public class Address {

	private String street;


	private String city;


	private String state;


	private String zip;

… // getters and setters
}
```

```
@Entity
public class Course {

	@Id
	private Integer id;

	@Column(name ="code" , length = 3, nullable = false )
	private String code; // Maximum of 3 characters (e.g. “544”), required field

	@Column(name ="name" , length = 50, nullable = false )
	private String name; // Maximum of 50 characters, required field

	@OneToMany
	// Set of pre-requisite courses for this course. Can be empty.
	private Set<Course> preRequisites = new HashSet<>();

… // Getters and setters
}
```

```
@Entity
public class Grade {

	@Id
	private Integer id;



	private Double grade;


	OneToOne(casecade = CasecadeType.persist)
	private Course course;

	ManyToOne(mappedBy = "grades")
	private Student student;
… // Getters and setters
}
```

```
@Entity
@inheritance (strategy = InheritanceType.JOINED)
@NoArgConstructor
public abstract class Person {
        @Id
       private Integer id;


       private String name;


       private String email;

        @Temporal(TemporalType.DATE)
       private LocalDate birthday;
        @ElementCollection
       private List<String> hobbies = new ArrayList<>();
        @embedded
       private Address address;

… // Getters and setters
}



public class Student extends Person {

       @Temporal(TemporalType .Date)
       private LocalDate entry;
       OneToMany(casecadeType.Persist)
       OrderColumn(name = "sequence")
       private List<Grade> grades = new ArrayList<>();

… // Getters and setters
}



public class Faculty extends Person {


        @Column(name =" bio" , length = 50, nullable = true)
       private String biography; //optional field with a maximum length of 2000 characters


        OneToMany
       private Set<Course> courses = new HashSet<>();

… // Getters and setters
}
```