http://hibernate.org/orm/what-is-an-orm/CS 378 Mid Term preparation

HW 1

1. Consider the url http://canvas.utexas.edu Run following commands and observe the difference between their responses:
   - curl -v -X HEAD http://canvas.utexas.edu
   - curl -v -X HEAD http://canvas.utexas.edu -H "Pragma: no-cache" -H "Cache-Control: max-age=0, no-cache, must-revalidate"

   Experiment with different sites and try to find more sites for which you observe the caching behavior similar to the canvas site.

```
$ curl -v -X HEAD http://canvas.utexas.edu
* Adding handle: conn: 0x25d8618
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x25d8618) send_pipe: 1, recv_pipe: 0
* About to connect() to canvas.utexas.edu port 80 (#0)
*   Trying 128.83.21.174...
* Connected to canvas.utexas.edu (128.83.21.174) port 80 (#0)
> HEAD / HTTP/1.1
> User-Agent: curl/7.30.0
> Host: canvas.utexas.edu
> Accept: */*
>
< HTTP/1.1 200 OK
* Server nginx is not blacklisted
< Server: nginx
< Date: Fri, 21 Oct 2016 03:32:41 GMT
< Content-Type: text/html; charset=UTF-8
< Content-Length: 17460
< Vary: Accept-Encoding
< Last-Modified: Tue, 30 Aug 2016 22:17:48 GMT
< ETag: "9938bbca729d149e-4434-53b5159c75a18"
< X-CacheServer: 172.17.99.42
< X-Cache: HIT
< Accept-Ranges: bytes
< Set-Cookie: balancer_id=utweb-px-z1-p01_http|WAmMX|WAmMX; path=/
< Cache-control: private
<
* transfer closed with 17460 bytes remaining to read
* Closing connection 0
curl: (18) transfer closed with 17460 bytes remaining to read
```

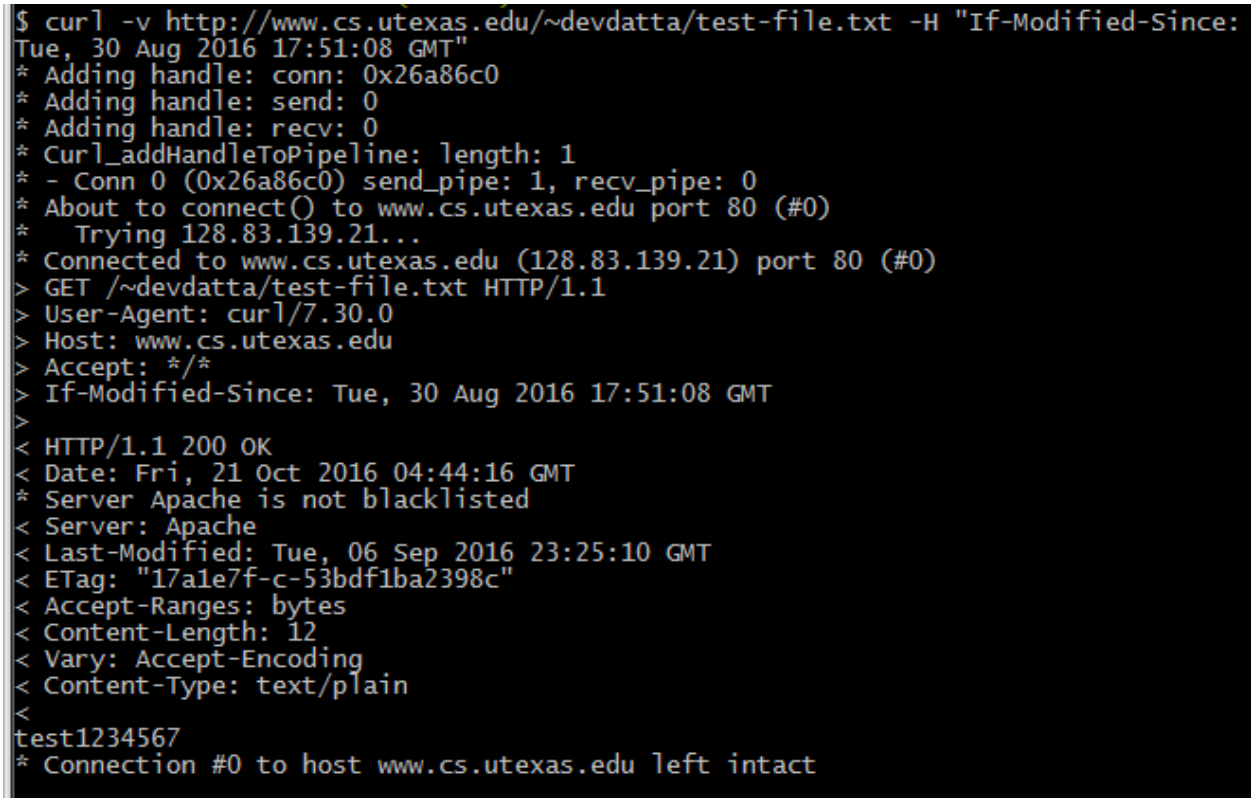
```
$ curl -v -X HEAD http://canvas.utexas.edu -H "Pragma: no-cache" -H "Cache-Control: max-age=0, no-cache, must-revalidate"
* Adding handle: conn: 0xc38728
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0xc38728) send_pipe: 1, recv_pipe: 0
* About to connect() to canvas.utexas.edu port 80 (#0)
*   Trying 128.83.21.174...
* Connected to canvas.utexas.edu (128.83.21.174) port 80 (#0)
> HEAD / HTTP/1.1
> User-Agent: curl/7.30.0
> Host: canvas.utexas.edu
> Accept: */*
> Pragma: no-cache
> Cache-Control: max-age=0, no-cache, must-revalidate
>
< HTTP/1.1 200 OK
* Server nginx is not blacklisted
< Server: nginx
< Date: Fri, 21 Oct 2016 03:36:00 GMT
< Content-Type: text/html; charset=UTF-8
< Content-Length: 17460
< Vary: Accept-Encoding
< Set-Cookie: cache_id=d5d9db6a6096d68494a5f589655f81ef; expires=Fri, 21-Oct-16 04:36:00 GMT; path=/
< Last-Modified: Tue, 30 Aug 2016 22:17:48 GMT
< ETag: "9938bbca729d149e-4434-53b5159c75a18"
< X-BackendServer: 172.17.99.34:80
< X-CacheServer: 172.17.99.42
< X-Cache: BYPASS
< Accept-Ranges: bytes
< Set-Cookie: balancer_id=utweb-px-z1-p01_http|WAmNI|WAmNI; path=/
< Cache-control: private
<
* transfer closed with 17460 bytes remaining to read
* Closing connection 0
curl: (18) transfer closed with 17460 bytes remaining to read
```

2. Run following commands and observe their effect on the response:

- curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt

```
$ curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt
* Adding handle: conn: 0xe28698
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0xe28698) send_pipe: 1, recv_pipe: 0
* About to connect() to www.cs.utexas.edu port 80 (#0)
*   Trying 128.83.139.21...
* Connected to www.cs.utexas.edu (128.83.139.21) port 80 (#0)
> GET /~devdatta/test-file.txt HTTP/1.1
> User-Agent: curl/7.30.0
> Host: www.cs.utexas.edu
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Fri, 21 Oct 2016 04:24:26 GMT
* Server Apache is not blacklisted
< Server: Apache
< Last-Modified: Tue, 06 Sep 2016 23:25:10 GMT
< ETag: "17a1e7f-c-53bdf1ba2398c"
< Accept-Ranges: bytes
< Content-Length: 12
< Vary: Accept-Encoding
< Content-Type: text/plain
<
test1234567
* Connection #0 to host www.cs.utexas.edu left intact
```

- curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Modified-Since: Tue, 30 Aug 2016 17:51:08 GMT"



- curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Modified-Since: Tue, 29 Aug 2016 17:51:08 GMT"

```
bikrantа@BIKRANTA-VAIO ~ (master)
$  curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Modified-Since:
 Tue, 29 Aug 2016 17:51:08 GMT"
* Adding handle: conn: 0x26286c0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x26286c0) send_pipe: 1, recv_pipe: 0
* About to connect() to www.cs.utexas.edu port 80 (#0)
*   Trying 128.83.139.21...
* Connected to www.cs.utexas.edu (128.83.139.21) port 80 (#0)
> GET /~devdatta/test-file.txt HTTP/1.1
> User-Agent: curl/7.30.0
> Host: www.cs.utexas.edu
> Accept: */*
> If-Modified-Since: Tue, 29 Aug 2016 17:51:08 GMT
>
< HTTP/1.1 200 OK
< Date: Fri, 21 Oct 2016 04:46:22 GMT
* Server Apache is not blacklisted
< Server: Apache
< Last-Modified: Tue, 06 Sep 2016 23:25:10 GMT
< ETag: "17a1e7f-c-53bdf1ba2398c"
< Accept-Ranges: bytes
< Content-Length: 12
< Vary: Accept-Encoding
< Content-Type: text/plain
<
test1234567
* Connection #0 to host www.cs.utexas.edu left intact
```

- curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Modified-Since: Tue, 31 Aug 2016 17:51:08 GMT"

```
$ curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Modified-Since:
Tue, 31 Aug 2016 17:51:08 GMT"
* Adding handle: conn: 0x8086c0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x8086c0) send_pipe: 1, recv_pipe: 0
* About to connect() to www.cs.utexas.edu port 80 (#0)
*   Trying 128.83.139.21...
* Connected to www.cs.utexas.edu (128.83.139.21) port 80 (#0)
> GET /~devdatta/test-file.txt HTTP/1.1
> User-Agent: curl/7.30.0
> Host: www.cs.utexas.edu
> Accept: */*
> If-Modified-Since: Tue, 31 Aug 2016 17:51:08 GMT
>
< HTTP/1.1 200 OK
< Date: Fri, 21 Oct 2016 04:49:04 GMT
* Server Apache is not blacklisted
< Server: Apache
< Last-Modified: Tue, 06 Sep 2016 23:25:10 GMT
< ETag: "17a1e7f-c-53bdf1ba2398c"
< Accept-Ranges: bytes
< Content-Length: 12
< Vary: Accept-Encoding
< Content-Type: text/plain
<
test1234567
* Connection #0 to host www.cs.utexas.edu left intact
```

- curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-None-Match:\"17a1e7f-6-53b4da01d1629\""

```
$ curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-None-Match: \"17
a1e7f-6-53b4da01d1629\""
* Adding handle: conn: 0x25a86c0
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x25a86c0) send_pipe: 1, recv_pipe: 0
* About to connect() to www.cs.utexas.edu port 80 (#0)
*   Trying 128.83.139.21...
* Connected to www.cs.utexas.edu (128.83.139.21) port 80 (#0)
> GET /~devdatta/test-file.txt HTTP/1.1
> User-Agent: curl/7.30.0
> Host: www.cs.utexas.edu
> Accept: */*
> If-None-Match: "17a1e7f-6-53b4da01d1629"
>
< HTTP/1.1 200 OK
< Date: Fri, 21 Oct 2016 04:50:29 GMT
* Server Apache is not blacklisted
< Server: Apache
< Last-Modified: Tue, 06 Sep 2016 23:25:10 GMT
< ETag: "17a1e7f-c-53bdf1ba2398c"
< Accept-Ranges: bytes
< Content-Length: 12
< Vary: Accept-Encoding
< Content-Type: text/plain
<
test1234567
* Connection #0 to host www.cs.utexas.edu left intact
```

- curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Match: \"17a1e7f-6-53b4da01d1629\""

```
$ curl -v http://www.cs.utexas.edu/~devdatta/test-file.txt -H "If-Match: \"17a1e7f
-6-53b4da01d1629\""
* Adding handle: conn: 0xed86e8
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0xed86e8) send_pipe: 1, recv_pipe: 0
* About to connect() to www.cs.utexas.edu port 80 (#0)
*   Trying 128.83.139.21...
* Connected to www.cs.utexas.edu (128.83.139.21) port 80 (#0)
> GET /~devdatta/test-file.txt HTTP/1.1
> User-Agent: curl/7.30.0
> Host: www.cs.utexas.edu
> Accept: */*
> If-Match: "17a1e7f-6-53b4da01d1629"
>
< HTTP/1.1 412 Precondition Failed
< Date: Fri, 21 Oct 2016 05:48:58 GMT
* Server Apache is not blacklisted
< Server: Apache
< Last-Modified: Tue, 06 Sep 2016 23:25:10 GMT
< ETag: "17a1e7f-c-53bdf1ba2398c"
< Accept-Ranges: bytes
< Content-Length: 0
< Vary: Accept-Encoding
< Content-Type: text/plain
< X-Pad: avoid browser bug
<
* Connection #0 to host www.cs.utexas.edu left intact
```

HW 2

1. Run following examples from Servlets directory: hello-world, query-parameters, session-example, test-jsoup-1
   - https://github.com/devdattakulkarni/ModernWebApps/Servlets

2. Run examples from XMLParsing directory

3. Modify the HelloServlet code to generate a "Last-Modifed" response header.
   - `response.addDateHeader("Last-Modified", getLastModified(request));`

is an example on how to respond with the header enabled. The second parameter can be changed to give a better time result. In a real situation you might use the current time or for serving a file the time the file was last modified on the system. Key to the answer is really just response.addDateHeader.

4. Modify/update the code in question 3 to handle "If-Modified-Since" request header.
   You can store the "Last-Modifed" timestamp of the helloworld resource in a HashTable/HashMap.
   Make calls to your Servlet by passing in different values for the "If-Modified-Since" request header.

- If the header is pulled in from the request and compared to the last modified value the next action (return 304 or full response) can be determined.

5. Experiment with using combination of If-Modified-Since and If-None-Match request headers against a site of your choosing. Determine through experimentation which header takes precedence.
- ASSUMPTION: If-None-Match is a requirement to check (strong validator) so that one should take precedence. Spec has a MUST/SHOULD relationship.

6. Consider a University network where all the network traffic goes through a University-wide proxy server which implements caching according to HTTP 1.0 and 1.1.
Suppose that a user makes request to resource (r1) and the response sent by the origin server
for that resource has a Cache-control header with max-age: 10
Suppose another user makes request to the same resource after two minutes, what will happen?
Suppose a user wants to ensure that the response is only received from the origin server.
How can this be achieved?
- The second request will get the data from the origin server. Setting max-age to 10 caches the data for ten seconds. The next request would be at 120 seconds, so the cache would be invalidated. Can ensure that requests are given by the origin server by refreshing the page (modern browsers then bypass cache).

7. Explain what is a strong validator? Use examples to explain.
- Validators are used to verify that the data matches/is similar. A strong validator is tied to the data content to verify that that data is unchanged byte for byte, so any change in the data will invalidate the Etag. A validator can be seen as similar to a checksum. If a file changes then so with a checksum on that file. Similarly, if a file changes the etag/validator data will change.
d

8. Answer true or false:
(a) If-None-Match request header is a way for the client to control when new ETag value is calculated by the server:

False, client sends etag, if it has changed the server recalculated the etag, if it hasn't a 304 response is sent back. Client doesn't control when the new ETag value is calculated by the server. They merely check whether the Etag they have is the same as the etag on the server

(b) The reason for the origin server to include both Expires header and the max-age directive in its response is to allow proxy servers that understand HTTP 1.1 longer expiration

time for that resource by specifying the max-age value that is greater than the time included in the Expires header.

True, in HTTP 1.1 the max-age header overrides the Expires header. HTTP 1.0 will use the expires header.

9. Explain what is Transfer-encoding response header.

specifies the form of encoding used to safely transfer the entity to the user.

10. Suppose server wants to send a HTML page whose size is 48 bytes, and suppose each chunk can only accommodate 16 bytes.

Show the server's response.

HTTP: 1.1/ OK
Content-type: text/plain
Transfer-Encoding: chunked

16\r\n
data\r\n
16\r\n
data\r\n
16\r\n
data\r\n
0\r\n
\r\n

HW 3

1. Try test-load-on-startup Servlet project from class github repository
- Change the values of <load-on-startup> attribute in web.xml for both the Servlets to understand how Servlet startup behavior is affected by the specified values.
- Answer following questions:
- What is the output of deploying the project when "load-on-startup" for testingLoadOnStartup2 is set to 2 whereas the "load-on-startup" for testingLoadOnStartup1 is set to 1
- What happens when we omit load-on-startup from both Servlet definitions
- What is the init-param attribute in web.xml?
- What is the context-param attribute in web.xml?
- What is the difference between init-param and context-param attributes?

2. Try spring-email-service Servlet project
- Execute following queries:

- http://localhost:8080/spring-email-service/
- http://localhost:8080/spring-email-service/?action=compose
- http://localhost:8080/spring-email-service/calculator?values=1,4,5&operator=add
- http://localhost:8080/spring-email-service/calculator?values=1,4,5&operator=multiply
    - Study EmailController.java and read about following annotations:
- @RequestMapping, @RequestHeader, @RequestParam, @ResponseBody,@Controller

- What is dependency injection? What dependency injection method is used in spring-email-service?

In software engineering, **dependency injection** is a software design pattern that implements inversion of control for resolving dependencies. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state.[1] Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern.

This fundamental requirement means that assigning values to attributes (class variables) using new or static methods is prohibited within the class. Constructors should not call constructors. They should accept values passed in from outside.

Dependency injection allows a program design to follow the dependency inversion principle. The client delegates the responsibility of providing its dependencies to external code (the injector) . The client is not allowed to call the injector code.[2] It is the injecting code that constructs the services and calls the client to inject them. This means the client code does not need to know about the injecting code. The client does not need to know how to construct the services. The client does not need to know which actual services it is using. The client only needs to know about the intrinsic interfaces of the services because these define how the client may use the services. This separates the responsibilities of use and construction

Setter Injection

- How are dependencies specified in Spring?

- What are advantages of using an interface instead of concrete implementation when building our programs?

- What is a Java Bean? How are Java beans used in dependency injection?

HW 4
1. servlet-unit-test
   - Run unit tests defined in TestHelloServlet.java andTestCalculatorServiceImpl.java through Eclipse/IntelliJ

- Test "testdoGet4" in TestHelloServlet is currently failing. Figure out why and then fix appropriate code in HelloServlet so that thetest passes.
- Several tests in TestCalculatorServiceImpl are currently failing. Figure out why and then fix CalculatorServiceImpl to make these tests pass.
- Add tests for following scenarios to TestCalculatorServiceImpl:
  - Values=[1,2,3,4], Operator=add
  - Values=[1,2,3,4], Operator=subtract
  - Values=[1,2,3,4], Operator=multiply
  - Values=[1,2,3,4], Operator=divide
  - Values=[1], Operator=subtract
  - Values=[1], Operator=multiply
  - Values=[1], Operator=divide
  - Values=[0,0], Operator=add
  - Values=[0,0], Operator=subtract
  - Values=[0,0], Operator=multiply
  - Values=[0,0], Operator=divide
  - Values=[0,1], Operator=add
  - Values=[1,0], Operator=add
  - Values=[1,0], Operator=subtract
  - Values=[0,1], Operator=subtract
  - Values=[1,0], Operator=multiply
  - Values=[0,1], Operator=multiply

2. mockito-example
- Run unit tests defined in TestEmailController.java andTestEnglishEditorServiceImpl.java through Eclipse/IntelliJ
- Test "testGetGreetingNullAction" in TestEmailController from mockito-example is currently failing. Figure out why and then modify appropriate code in EmailController so that the test passes.
- Test "testWithGhostObject" in TestEmailController from mockito-example is failing. Figure out what is causing the test to fail. What technique can you use to make this test pass?
- Write one or more unit tests similar to "testReadDataFromEavesdrop" test to test your assignment2 logic.

HW 5

1. What is the meaning of context-param element in web.xml?
   - If you want to store data which is common for **whole application** and if it doesn't change frequently you can use `<context-param>` instead of `setAttribute()` method of the application context. **context-param** are declared under the tag `web-app`.
     - E.g

```
<web-app>
        <context-param>
                <param-name>Country</param-name>
                <param-value>India</param-value>
```

```
        </context-param>
        <context-param>
        <param-name>Age</param-name>
        <param-value>24</param-value>
        </context-param>
</web-app>
```

Usage in the application either in a JSP or Servlet.

```
getServletContext().getInitParameter("Country");
getServletContext().getInitParameter("Age");
```

2. What is the meaning of init-param element in web.xml?
   - If you want to store particular data which is only confined to a **particular servlet scope**, then you can use `<init-param>` .Anything you declare inside `<init-param>` is only accessible only for that particular servlet.The **init-param** is declared inside the `<servlet>` tag.
     - E.g

```
<servlet>
        <display-name>HelloWorldServlet</display-name>
        <servlet-name>HelloWorldServlet</servlet-name>
        <init-param>
                <param-name>Greetings</param-name>
                <param-value>Hello</param-value>
        </init-param>
</servlet>
```

Access those parameters in the servlet:

```
System.out.println(getInitParameter("Greetings"));
```

3. What are similarities and differences between context-param and init-param
   - DIfference: Scope
   - Similarities: Access by servlet

4. Spring, RESTEasy, Servlets are examples of which of the following:
   (a) abstractions
   (b) frameworks
   (c) tools
   (d) all of the above

5. What is the purpose of getClasses() and getSingletons() methods in a RESTEasy application class?
   - getClass()

    - Get a set of root resource and provider classes. The default lifecycle for resource class instances is per-request. The default lifecycle for providers is singleton.
- getSingletons()
    - Get a set of root resource and provider instances. Fields and properties of returned instances are injected with their declared dependencies (see Context) by the runtime prior to use.

6. When a POST request is received by a REST application, the RESTEasy framework uses a specific technique to convert the XML body of POST request into a Java object. What is this technique called?
    - Unmarshalling? - <we used JAXB for doing marshalling/unmarshalling>

7. When designing a REST API, what are some of questions/concerns that should be kept in mind?
    - The API semantics must be intuitive. URI, payload, request or response: a developer should be able to use them without referring to the API documentation.
        - The terms must be common and concrete, rather than emanate from a functional or technical jargon. Customers, orders, addresses, products are all good examples.
        - There should not be different ways to achieve the same action.

8. One of the reasons Internet is so popular is that we can navigate from one web page to another by following the links available on a page.
In designing REST APIs, is it a good idea to provide such links?
What REST principle can we use to inform our decision?

9. True or False:
- **When a dependency is changed, functional tests need to be updated.**
    False, changing a dependency does not change the end result of a function call or program (unit tests would need to be updated however)
- **Functional testing is focused on testing the functional behavior of methods within the code.**
    True, functional tests are for expected outcomes
- **Unit testing and functional testing are both required for ensuring software robustness**
    True
- **An example of idempotent sequence of REST methods is <PUT, PUT>**
    True, PUT lines up with the update, assuming that the update is the same information, the update would be idempotent in subsequent calls
- **PATCH is a REST method that allows conditional updates to REST resources**
    False? Patch allows us to update only certain fields in a RESTful API without having to send the entire object's data but I wouldn't consider that conditional

HW 6

1) What is the difference between primary key and foreign key?

Primary Key	Foreign Key
● Primary key allows each **row in a table to be uniquely identified** and ensures that no duplicate rows exist and no null values are entered. ● Primary Key can't accept null values. ● By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index. ● We can have only one Primary key in a table.	● Foreign key is a field in the table that is **primary key in another table.** ● Foreign key can accept multiple null value. ● Foreign key do not automatically create an index, clustered or nonclustered. You can manually create an index on foreign key. ● We can have more than one foreign key in a table.

2) What is the difference between natural key and surrogate key?

Natural key	**Surrogate key.**
A key that is formed of attributes that already exist in the real world. For example, U.S. citizens are issued a Social Security Number (SSN)	A key with no business meaning.

3) What are some of the advantages of using a natural key as the primary key in a table?

It's simpler to use vs a surrogate key as its a key that's already prebuilt into the data that your using. Consider the SSN example. Everyone already has a SSN to be uniquely identified, so when you store your data you can use it.

4) What are some of the advantages of using a surrogate key as the primary key in a table?

It's a dependable way of storing unique values. If you can't find a natural key or your natural key isn't a stable (i.e it changes a lot per person) or small (just a single value, rather than a connection of different values - think a combination key) then adding the surrogate is better since it lets you handle the data with ease as you can just enforce your own criteria for uniqueness into the data

5) What are bind parameters in context of JDBC?

Parameter binding provides a means of separating executable code, such as SQL,from content, transparently handling content encoding and escaping."

```
String query = "update projects set name=? ,description = ? where
project_id = ?";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, name);
stmt.setString(2, description);
stmt.setInt(3, project_id);
```

6) Why is it important to use PreparedStatement with bind parameters when using JDBC? Give precise answer.

Prevent sql injection.

“This way, the developer doesn't need to understand the complexities that arise from mixing user input with executable code. For this to be effective all untrusted inputs need to be bound. If SQL is built through concatenation, interpolation, or formatting methods, none of the resulting string should be created from user input.”

7) Consider that we are building an email system in which we want to support a feature where we can attach arbitrary tags to emails. An email can have multiple tags and multiple emails can have the same tag.

For example, consider following three emails with subject and tags.

Subject	Tags
Assignment 6	UT, Modern Web apps
Study group	UT
Black Friday	Shopping

Define tables with appropriate columns to enable following queries:

- Find all emails corresponding to a specific tag
- Find all tags for a particular email

Since we have a cross join kind situation here we’ll need to use similar set of tables that we used in assignment 5. However a tag used in one email isn’t unique to the email, but unique over all (that makes the most sense anyway)

```
create table emails(subject varchar(255), email_id int NOT NULL
AUTO_INCREMENT, PRIMARY KEY(email_id));

create table tags(name varchar(255) NOT NULL, tag_id int, PRIMARY
KEY(tag_id), FOREIGN KEY(email_id) REFERENCES emails(email_id));

select * from (tags join emails on tags.email_id = emails.email_id) where
tags.name t="UT";

select * from (tags join emails on tags.email_id = emails.email_id) where
emails.subject="Assignment6";
```

8) Using JDBC write following queries:

- Insert an email with following subject and tag ("Finals", "UT")

    Basically need to break query into two parts

    -First to add the email with the subject finals

    -2nd to add the Tag “UT” and add the reference to the email “Finals”

Adding the email finals

```
Connection conn = ds.getConnection();

String insert = "INSERT INTO projects(name, description) VALUES(?, ?)";
PreparedStatement stmt = conn.prepareStatement(insert, Statement.RETURN_GENERATED_KEYS);

               stmt.setString(1, c.getSubject());
                stmt.setString(1, c.get());

               int affectedRows = stmt.executeUpdate();

               if (affectedRows == 0) {
                      throw new SQLException("Creating project failed, no rows affected.");
               }

               ResultSet generatedKeys = stmt.getGeneratedKeys();
               if (generatedKeys.next()) {
                      c.setProjectId(generatedKeys.getInt(1));
               } else {
                      throw new SQLException("Creating project failed, no ID obtained.");
               }
```

```
                // Close the connection
                conn.close();

                return c;
```

Adding the tag “UT”, with the reference to UT

insert into tags(name, email_id) values ("UT", (select email_id from emails where email_subjuct=""));

```
Public static tag addTag(Tag t){
     Connection conn = ds.getConnection();
     String insert = "INSERT INTO tags(name, email) VALUES(?, ?)";
     PreparedStatement stmt = conn.prepareStatement(insert,
Statement.RETURN_GENERATED_KEYS);
                stmt.setString(1, t.getName());
                stmt.setString(2, t.getEmail());


                int affectedRows = stmt.executeUpdate();

                if (affectedRows == 0) {
                        throw new SQLException("Creating tag failed, no rows affected.");
                }

                ResultSet generatedKeys = stmt.getGeneratedKeys();
                if (generatedKeys.next()) {
                        c.setProjectId(generatedKeys.getInt(1));
                } else {
                        throw new SQLException("Creating project failed, no ID obtained.");
                }

                // Close the connection
                conn.close();

                return t;
       }
```

One thing to remember is that you’ll need to define the objects that JDBC is going to be working with in order for an insert of any kind is gonna be working.

- Query email subjects that have the tag "UT"

//The question here wants us to add an email with subject “Finals” and Tag is “UT”.
//This answer should be the following:

```
insert into emails(subject("Finals");
insert into tags(name) values("UT");
insert into tags(name, email_id) values("UT", (select email_id from emails where
subject="Finals"));
```

9) Explain with an example what is a transaction?
Transactions are mechanism used to handle multiple changes to a database from multiple sources as well handling if a change doesn’t work correctly and rolling back.
For example, with the Emails and tags tables, a transaction would be similar to the answer of 8: with the inclusion of over head to establish the transaction and commiting the changes, probably using sessions.

```
public Long addAssignment(String title) throws Exception {
             Session session = sessionFactory.openSession();
             Transaction tx = null;
             Long assignmentId = null;
             try {
                    tx = session.beginTransaction();
                    Assignment newAssignment = new Assignment( title, new Date(),
new Long(1));
                    session.save(newAssignment);
                assignmentId = newAssignment.getId();
                tx.commit();
             } catch (Exception e) {
                    if (tx != null) {
                           tx.rollback();
                           throw e;
                    }
             }
             finally {
                    session.close();
             }
             return assignmentId;
      }
```

10) What mechanism is available in JDBC to obtain the primary key id of an inserted row?

```
      pInsertOid =
connection.prepareStatement(INSERT_OID_SQL,Statement.RETURN_GENERATED_KEYS);
```

11) What is the difference between executeUpdate and executeQuery JDBC methods?

executeQuery is used for SQL statements which retrieve some data from the database. For example is SELECT statement. This method is meant to be used for select queries which fetch some data from the database

executeUpdate is used for SQL statements which update the database in some way. For example INSERT, UPDATE and DELETE statements. All these statements are DML(Data Manipulation Language) statements. This method can also be used for DDL(Data Definition Language) statements

executeQuery()	executeUpdate()	execute()
This method is used to execute the SQL statements which retrieve some data from the database.	This method is used to execute the SQL statements which update or modify the database.	This method can be used for any kind of SQL statements.
This method returns a ResultSet object which contains the results returned by the query.	This method returns an int value which represents the number of rows affected by the query. This value will be the 0 for the statements which return nothing.	This method returns a boolean value. TRUE indicates that query returned a ResultSet object and FALSE indicates that query returned an int value or returned nothing.
This method is used to execute only select queries.	This method is used to execute only non-select queries.	This method can be used for both select and non-select queries.
Ex : SELECT	Ex : DML → INSERT, UPDATE and DELETE DDL → CREATE, ALTER	This method can be used for any type of SQL statements.

12) Explain with examples the following - inner join, left outer join, right outer join

Simple Example: Lets say you have a `Students` table, and a `Lockers` table.

Each student can be assigned to a locker, so there is a `LockerNumber` column in the `Student`table. More than one student could potentially be in a single locker, but especially at the beginning of the school year, you may have some incoming students without lockers and some lockers that have no students assigned.

For the sake of this example, lets say you have **100 students**, 70 of which have lockers. You have a total of **50 lockers**, 40 of which have at least 1 student and 10 lockers have no student.

INNER JOIN is equivalent to "*show me all students with lockers*". Any students without lockers, or any lockers without students are missing. **Returns 70 rows**

LEFT OUTER JOIN would be "*show me all students, with their corresponding locker if they have one*". This might be a general student list, or could be used to identify students with no locker. **Returns 100 rows**

RIGHT OUTER JOIN would be "*show me all lockers, and the students assigned to them if there are any*". This could be used to identify lockers that have no students assigned, or lockers that have too many students.

Returns 80 rows (list of 70 students in the 40 lockers, plus the 10 lockers with no student)

FULL OUTER JOIN would be silly and probably not much use. Something like "*show me all students and all lockers, and match them up where you can*"

Returns 110 rows (all 100 students, including those without lockers. Plus the 10 lockers with no student)

CROSS JOIN is also fairly silly in this scenario. It doesn't use the linked `lockernumber` field in the students table, so you basically end up with a big giant list of every possible student-to-locker pairing, whether or not it actually exists.

Returns 5000 rows (100 students x 50 lockers). Could be useful (with filtering) as a starting point to match up the new students with the empty lockers.

13) True or False

- JDBC code should be added in resource layer when designing a REST API
  False - service layer
- A datasource in JDBC is the place where data is stored
  True
- We used web.xml to pass JDBC connection parameters to our program so that it made the program independent of the specific database used.
  True

HW 7

1) What is the meaning of @Entity annotation in Hibernate?
   It marks the class as an entity bean.

@Entity Annotation:

The EJB 3 standard annotations are contained in the javax.persistence package, so we import this package as the first step. Second we used the @Entity annotation to the Employee class which marks this class as an entity bean, so it must have a no-argument constructor that is visible with at least protected scope.

```
import javax.persistence.*;

@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;
```

2) What is the purpose of transaction when using Hibernate to interact with the database?

A transaction is associated with a Session and is usually instantiated by a call to Session.beginTransaction(). A single session might span multiple transactions since the notion of a session (a conversation between the application and the datastore) is of coarser granularity than the notion of a transaction. However, it is intended that there be at most one uncommitted Transaction associated with a particular Session at any time.

```
public Long addAssignment(String title) throws Exception {
              Session session = sessionFactory.openSession();
              Transaction tx = null;
              Long assignmentId = null;
              try {
                     tx = session.beginTransaction();
                     Assignment newAssignment = new Assignment( title, new Date(),
new Long(1));
                     session.save(newAssignment);
                  assignmentId = newAssignment.getId();
                  tx.commit();
              } catch (Exception e) {
                     if (tx != null) {
                            tx.rollback();
                            throw e;
                     }
              }
              finally {
                     session.close();
              }
              return assignmentId;
       }
```

3) In hw6, you defined an email system that allowed tagging individual emails. Now consider that we want to model the feature that an email can contain zero or more attachments. Using Hibernate, write Entity classes to model this scenario.

Email Object

```
package assign.domain;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;
import org.hibernate.annotations.CascadeType;
import org.hibernate.annotations.GenericGenerator;


@XmlRootElement(name = "email")
@XmlAccessorType(XmlAccessType.FIELD)
@Entity
@Table( name = "emails" )
public class Email {

   @XmlAttribute
   private Long id;

    private String subject;
    private Set<Tag> tags;
    private Set<Attachments> attachments;


    public Email() {
      // this form used by Hibernate
    }

    public Email(String subject, String description, Long id) {
      this.subject = subject;
      this.description = description;
      this.id = id;
    }

    @Id
   @GeneratedValue(generator="increment")
   @GenericGenerator(name="increment", strategy = "increment")
    public Long getId() {
      return id;
    }

    private void setId(Long id) {
```

```
      this.id = id;
    }

    @Column(name="subject", nullable = false)
    public String getSubject() {
      return name;
    }

    public void setSubject(String subject) {
      this.name = name;
    }


    @OneToMany(mappedBy="email")
    @Cascade({CascadeType.DELETE})
    public Set<Tag> getTags() {
      return this.meetings;
    }

    public void setTags(Set<Tag> meetings) {
      this.meetings = meetings;
    }


    @OneToMany(mappedBy="email")
    @Cascade({CascadeType.DELETE})
    public Set<Attachments> getAttachments() {
      return this.meetings;
    }

    public void setAttachments (Set<Attachments> meetings) {
      this.meetings = meetings;
    }
}
```

Attachments object

```
package assign.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
```

```java
@XmlRootElement(name = "attachment")
@XmlAccessorType(XmlAccessType.FIELD)
@Entity
@Table( name = "attachments" )
public class Attachment {

   @XmlAttribute
   private Long id;

    private String name;
    private int year;

    @XmlTransient
    private Project utcourse; // course or something else

    public Attachment() {
      // this form used by Hibernate
    }

    public Attachment(String attachmentName, int year, long id) {
      // for application use, to create new assignment
      this.name = attachmentName;
      this.year = year;
      this.id = id;
    }


    @Id
   @GeneratedValue(generator="increment")
   @GenericGenerator(name="increment", strategy = "increment")
    @GeneratedValue(strategy=GenerationType.AUTO)
    public Long getId() {
      return id;
    }

    private void setId(Long id) {
      this.id = id;
    }

    @ManyToOne
    @JoinColumn(name="email_id")
```

```
    public Project getProject() { // property named course available on
this object
      return this.utcourse;
    }


    public void setEmail(Project c) {
      this.utcourse = c;
    }

    public String getName() {
      return name;
    }

    public void setName(String attachmentName) {
      this.name = attachmentName;
    }


}
```

4) Using Hibernate, write following queries:

- Insert an email with one subject and two attachments
  (subject: "Finals", attachment names: "Study material", "class notes")

Same as JDBC, need to add the Email first, and then the attachments, with reference to the emails

```
create table emails(subject varchar(255), email_id int NOT NULL AUTO_INCREMENT, PRIMARY KEY(email_id));

create table attachments(name varchar(255) NOT NULL, attachment_id int, PRIMARY KEY(attachment_id),
FOREIGN KEY(email_id) REFERENCES emails(email_id));
```

```
public Long addAttachmentsToEmail(List<String> attachment, String emailName) throws Exception {
                Session session = sessionFactory.openSession();
                Transaction tx = null;
                Long emailId = null;
```

```
            try {
                    tx = session.beginTransaction();
                    Email email = new Email(emailName);
                    session.save(email);
                    emailId = email.getId();
                    for(String a : assignments) {
                            Attachment newAttachment = new Attachment( a, new Long(1));
                            newAttachment.setEmail(email);
                            session.save(newAttachment);
                    }
                tx.commit();
            } catch (Exception e) {
                    if (tx != null) {
                            tx.rollback();
                            throw e;
                    }
            }
            finally {
                    session.close();
            }
            return emailId;
      }
```

- Query all the attachment names for a given email subject.

```
public List<Object[]> getAttachmentsForAEmail(String emailSubject) throws
Exception {
            Session session = sessionFactory.openSession();
            session.beginTransaction();
            String query = "from Assignment a join a.subject c where
c.subjct = :cname";

      List<Object[]> attachments =
session.createQuery(query).setParameter("cname",
            emailSubject).list();
            session.close();

            return attachments;
      }
```

5) Explain with examples meaning of the following annotations:

- @ManyToOne:

Example: Customer orders would be many to one. A single customer has multiple orders tied to it.

- @JoinColumn: used to specify a mapped column for joining an entity association.
  Example:
  ```
  @ManyToOne
  @JoinColumn(name="ADDR_ID")
  public Address getAddress() { return address; }
  ```

- @OneToMany(mappedBy="..."):
  Example - Customer orders. The customer class would be one to many, with many orders attached to a single customer.

- @Column: The @Column annotation is used to specify the details of the column to which a field or property will be mapped

- @Id: Annotate the id column using @Id. Each entity bean will have a primary key, which you annotate on the class with the **@Id** annotation

6) True or False
- A detached object is an object that was associated with a session some time in its lifetime. True
- In a REST API, the object into which the input XML is unmarshalled is a transient object. True
- Bind parameters are important when using Hibernate. True
- Bind parameters are only relevant and useful in JDBC. False, Hibernate
- Many-to-one annotation should be included on the parent entity in the relationship. True
- The owning side of the relationship is the entity that has reference to the other entity. True

  Unidirectional Relationship: A unidirectional relationship is a relationship in which only one of the two entities have the owning side. In unidirectional relationship only one entity can have the relationship property or field that may refer to the other.

  Bi-directional Relationships: A bi-directional relationship is the relationship in which both the entities have the owning side. In bi-directional relationship each entity in the relation have the relationship fields or property.
- Hibernate entities can be used to represent input/output objects as well by using JAXB annotations alongside Hibernate annotations on entity classes. True

HW 8

1) What are the key differences between IaaS and PaaS models of cloud computing?

Infrastructure as a service ● This model is about getting a VM, sshing onto it, and installing an app contain on it like Tomcat and deploy your code Advantages ● Get complete control over how the environment is set up. More or less like a virtualevn for you to work with Disadvantages ● Have to deal with setting up the environment – the stuff outsides of actually coding: ○ setting up the ports for the server for example. ○ Firewall Examples: ● Commercial ○ Amazon AWS, Google Computer Engine. Microsoft Azure, IBM, Softlayer ● Open source ○ Open Stack, CloudStack	Platform as a service ● Develop your code and push it to a vm that's all set up for you: It's all abstracted away as the "platform" which handles the particulars of deployment Pros: ● No need to deal with environment bs ● Only focus on coding the app ○ Appeal from a dev's point of view Cons: ● Hard to trouble shoot since no access to the VM or the application, only the code you push ○ Logs are available for you to look at to see. Examples: ● ● Heroku, Amazon Elastic Beanstalk, Google App Engine, RedHat Open Shift IBM BlueMix ● ● Openstack Solum, RedHat Open Shift

2) Consider a developer's workflow where a developer is developing their web application code.
During the development phase, one needs to deploy the work-in-progress code often.
What cloud computing model is suitable for this workflow. Why?

- The PaaS Model would be best as it allows minimal interruption of the service in the dev's workflow when deploying often.
- For example, deploying to Heroku is much easier to set up and maintain vs a AWS EC2 instance. Besides this the Heroku instance is also a lighter instance which means that it will be a much faster deployment rate vs the EC2 which is important when needing frequent iteration cycles.

3) Consider an operator's workflow where the operator needs to deploy an already built web application to production. What cloud computing model is suitable for this workflow. Why?

- The IaaS model as it would give the dev complete control over how the application is being deployed. At this part of the dev cycle, the dev isn't going to be needing to push that frequently, but they need to be able to make the deployment up to their specification

4) Consider the IaaS model of cloud computing such as Amazon EC2 which allows spinning up Virtual Machines in the cloud. Define a REST API to support following operations:

- Not sure how to approach this, but I guess focus on what resources a VM needs and that build the api based on that. These specifications are probably up for debate but i'll give my best answer
- Using the Azure's specification for an example
- https://docs.microsoft.com/en-us/rest/api/compute/virtualmachines
    - ID
    - SuperUser Access controls.
    - Size?
    - OS
    - Destination
    - Startup on Creation?
    - Uptime,
    - List of processes on the machine

- Spin up a virtual machine

Post

- Example: example.com/machines/new/

- Get information about a spawned virtual machine

Get

- Example: example.com/machines/456

- Stop a spawned virtual machine

Update (GET would work too)

- Update example:
    - example.com/machines/456
        - Data would include something like "state: stop"
- GET example
    - example.com/machines/456/stop

- Delete a spawned virtual machine

- GET would work best here (rather than DELETE as this is a live object not just a db entry)

- example.com/machines/456/delete

5) Consider your assignment 6. Suppose you have set the level of Logger and Handler to WARNING. Show the output that will be generated for each of the following calls: https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6/html/Development_Guide/Add_Logging_to_an_Application_with_JBoss_Logging.html

- Used this example for logging. Added it to my assignment 6. The import is already there if you had the resteasy dependency already added
- WARN – Use the WARN level priority for events that may indicate a non-critical service error. Resumable errors, or minor breaches in request expectations fall into this category. The distinction between WARN and ERROR may be hard to discern and so its up to the developer to judge. The simplest criterion is would this failure result in a user support call. If it would use ERROR. If it would not use WARN.

- Logger.info("Web App is starting up")

The Logger will let the call happen, but will give all the handlers this message
- Logger.config("Web App configuration is a.b.c")
- Logger.severe("Something severe has happened")

6) Considering the Java logging architecture, what message level would you assign to following situations that may happen in your program:
- Trace message of the program execution that should be displayed only when debugging an issue
    - FINE/FINER. Will show needed debug data.
- Exception stack trace
    - SEVERE. Exceptions/stack traces are needed at all times to diagnose crashes and other faults.
- Displaying the database credentials that would be used to connect to a database
    - FINEST. Never want to log sensitive information.

7) Consider an HTML page as shown below:

```
<html>
<body>
<div id="myDiv">
<p id="myP">Click this paragraph</p>
</div><br>

<script>
document.getElementById("myP").addEventListener("click", function() {
alert("You clicked the P element!");
}, false);

document.getElementById("myDiv").addEventListener("click", function() {
alert("You clicked the DIV element!");
}, false);
</script>
</body>
</html>
```

What messages will be displayed and in what order when you click inside the box on the "Click this paragraph" text.

You clicked the P element!
Why is it the P element and not DIV?
Tested it, the DIV alert will also appear after the P alert

The 3rd parameter is false so it uses bubbling which goes from the inner child to the parent. If it was true, then it would use capturing which would go from parent to the inner child.

8) Answer whether following URL pairs satisfy same origin policy or not:
a) http://www.abc.com/
http://www.abc-1.com/
No?

- I think it's no too, there's a chart https://en.wikipedia.org/wiki/Same-origin_policy

b) https://www.s3.com/ (private connection)
https://www.s3.com:445/

Different port so no. Careful… If it was port 443 is would be implementation dependent. 445 is SMB/netbios so it would be no. If It was port 80 then they would be the same since 80 is the default port.

What about c? This one is yes

http://www.cs.utexas.edu/
http://www.cs.utexas.edu/~devdatta