2011/2012 2nd Semester

Name: _____________

JEE Examination

Authorized documents: None
Duration : 1h30.

Each question can have multiple answers.
+1 for correct answers
-1 for incorrect answers

Total of 40 points.

Java Web

Question 1 : What is JEE ?

□ The paid version including support for Java from Sun (now Oracle)
□ An open source application server
□ A set of libraries and standards to facilitate the implementation of enterprise applications

Question 2 : How is the web content generated by the Servlet technology and those generated by the JSP technology different ?

□ JSPs offer types of Web content more advanced than servets, particularly in terms of presentation.
□ None, JSPs are models of writing Servlet
□ Servlets are compiled before being executed on the server, the JSP pages of scripts are interpreted by the web browser

Question 3 : Which of these classes are needed to make a query with a SELECT statement according to a standard JDBC 1 ?

□ `DataSource`	□ `ResultSet`
□ `DriverManager`	□ `org.hibernate.Session`
□ `Query`	□ `HttpSession`

Question 4 : What is the difference between the technique `forward` of `RequestDispatcher` and `sendRedirect` of `HttpServletResponse` ?

□ The `forward` technique is safer and more efficient
□ The `forward` technique allows the servlet to make objects available to JSP, the `sendRedirect` technique does not
□ the `forward` technique is performed on the server side , `sendRedirect` involves Web browser

Question 5 : What do we mean when we speak of scope for a bean ?

□ This is the lifetime of a bean in the server-side memory
□ Corresponds to the visibility of these properties (public, private, protected...)
□ Indicates if the bean has a reference to an object of type `HttpServletRequest` or `HttpSession` object

Question 6 : What is the advantage of introducing the MVC design pattern in a Web application ?

□ This essentially allows to manage object persistence in relational database
□ This allows among others to separate the problem of presentation when designing an application
□ This allows the flow of data between JSP and Servlet

Question 7 : What are the advantages in using an object type `Datasource`?

□ It allows you to write SQL queries regardless of the database type
□ Only an object of type `Datasource` allows to implement a connection pool
□ Access to the database is managed by the server which prevents changes in the application in case of migration

Question 8 : What is the major difference between `PreparedStatement` and a `CallableStatement` ?

□ A `PreparedStatement` allows dynamic values, unlike a `CallableStatement`. This enables the reuse and provides more security by avoiding SQL injections
□ The location the statement is written and the query compiled
□ A `PreparedStatement` is faster executing than `CallableStatement`

Question 9 :What is the file META-INF/context.xml that is put sometimes at the root of the Java Web application ?

□ This is the deployment descriptor of the application, it tells the application server how to supports this application
□ It represents the content to be added in the application scope (`ServletContext`) and therefore accessible to all users of the application
□ This is a configuration file used by Tomcat, it mainly describes the JNDI resources in charge of the application Server.

Question 10 : What difference is there between the libraries added to the build path in Eclipse and libraries placed in WEB-INF/lib ?

□ Libraries added in WEB-INF/lib will also be used when running the application classes, including Servlets
□ None, it's just 2 ways to reference libraries
□ Libraries present in WEB-INF/lib will be retrieved by the server and placed in its own library directory (/ lib or commons / lib on Tomcat)

Question 11 : Can you find an error in the following code ?

```
...
protected void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, java.io.IOException {
          Client c=clientService.getCurrentClient();
          Invoice f=(Invoice)request.getAttribute("invoice");
          boolean b=invoiceService.validateInvoice(f);
          if (b){
                    c.setInvoice(f);
                    RequestDispatcher disp=request.getRequestDispatcher("invoice-
                    ok.jsp");
                    disp.forward(request,response);
          }
          RequestDispatcher disp=request.getRequestDispatcher("invoice-ko.jsp");
          disp.forward(request,response);
}
...
```

□ The variable `invoice` must be recovered via a `request.getParameter` and not `request.getAttribute`
□ The variable `disp` is present twice, the code won't compile
□ No error, the code will compile and run correctly
□ The 1st forward does not stop the processing of doGet, an error occurs during the execution of the second forward

Question 12 : What difference is there between the directive `include` and the tag `include` of the namespace `jsp:` ?

□ The directive allows inclusion in the translation, the tag inclusion in the execution
□ The directive allows inclusion at the execution, the tag inclusion in the translation
□ The directive allows inclusion in the compilation, the tag inclusion in the translation
□ The directive allows inclusion in the translation, the tag inclusion in the compilation
□ The directive allows inclusion at the execution, the tag inclusion in the compilation
□ The directive allows inclusion at the compilation, the tag inclusion in the execution

Question 13 : *What are the differences between the storage of data by Cookie and storage by user session ?*

□ There are many differences. The location of the data, its lifetime, its type (String / Object) among others...
□ No difference, session data are stored in cookies
□ The Cookies of a website may be "hijacked" by other websites, it is best to keep the data in session.

Question 14 : *Is it possible, like for servlets, to add methods to JSPs ?*

□ Yes using the directive `<%@taglib %>`
□ No it is not possible, it is also one of the difference between Servlet and JSP in terms of functionality (see question 2)
□ Yes just add the method between scriptlets `<% %>`
□ Yes it uses the same technique as to add properties to the JSP

Question 15 : What is the method `setContentType` of the `HttpServletResponse` class for?

□ To indicate whether the JSP page is written in java or in another language
□ To explicitly tell the browser how it should treat the content returned
□ To indicate the type of tag set (taglib) used
□ To indicate the rsponse's type of protocol (Http, Ftp...)

Question 16 : How will the following EL expression be translated :
`${requestScope.invoice.i.j}` ?

□ `request.getAttribute("invoice").i.j`
□ `request.getParameter("invoice.i.j")`
□ `request.getAttribute("invoice.i.j")`
□ `request.getAttribute("invoice").getI().getJ()`

Question 17 : What is displayed by the following code ?

```
...
<c:if test="${!empty boolean}">
        <c:forEach items="${req.content}" var="ct">
                ct<BR/>
        </c:forEach>
</c:if>
...
```

□ If there is a variable "boolean" neither void nor empty in a scope, the contents of the collection returned by `getContent` from the variable "req" located in a scope.
□ If there is a variable "boolean" neither void nor empty in a scope, the contents of the variable "content" are located in the request scope (`HttpServletRequest`)
□ If there is a variable of boolean type in a scope, the entire contents of the collection returned by `getContent` from variable "req" located in a scope
□ If there is a variable "boolean" neither void nor empty in a scope, as many lines containing the string "ct" as elements in the collection returned by `getContent` of the variable "req" located in a scope
□ Nothing at all, `empty` does not mean anything in the JSTL syntax

Question 18 : Why is the scope of type "page" not used in Web MVC architecture ?

□ It could be that developers are more used to using scopes request and session.
□ The scope page is too short in time, it does not allow sharing of objects between the controller and the view
□ This scope does not exist
□ the scope page requires the `<%@ page import %>` that will import specific classes to the server. This prevents code portability.

Question 19 : Which method of the `ResultSet` class allows to retrieve a column of type float ?

□ `getNumber()`

□ `getNumeric()`
□ `getFloat()`
□ Impossible to say, it depends on the driver and the type of database
□ `next()`

<u>Question 20</u> : What is the String array that can optionally be provided as a parameter when creating `PreparedStatement` or executing `Statement` ?

□ These are the values of dynamic parameters indicated with ? in the SQL query
□ These are the types of SQL null values that can be obtained in the result
□ These are the names of the columns of the SQL query in order to make getXXX (nameOfTheColumn) instead of getXXX (indexOfTheColumn)
□ This is the list of columns whose value is affected by the database itself

Hibernate

<u>Question 21</u> : Is it possible to add Hibernate to any existing Java project ? What are the obstacles ?

□ It must already be a JEE Application, it is the only constraint.
□ It may be necessary to modify tables especially when we investigate what identifies objects. If the script of the database can not be changed then this can be an obstacle.
□ No there is absolutely no constraint, Hibernate fits any application at all engine databases (Oracle, MySQL, SQL Server ...) and any existing database structure (DDL).

<u>Question 22</u> : With Hibernate and generally all ORM framework, why should we describe mapping files (or annotations) ?

□ The `type` of columns in database (VARCHAR2, NUMBER ...) and the Java `type` (String, long …) are different, it is necessary to indicate to Hibernate to make both match, particularly through the `type` attribute of the `<property>` tag
□ The mapping files allow Hibernate to generate format-independent SQL queries
□ Because it is impossible to automate the mapping between object model and relational model because there are incompatible.

<u>Question 23</u> : In the "many to many" relation, what should we be careful about ?

□ In the case of a bidirectional association, one side "master" and one side "inverse" to avoid hibernate from performing a double update.
□ To give the join table a name that follows the Hibernate standard “TableName1”_”TableName2”
□ To have a Java class in the image of the join table, otherwise it will be impossible to Hibernate to create objects in the image of the records of this table.

<u>Question 24</u> : What is a Hibernate `Session` ?

□ It is a Http session which can automatically benefit of the Hibernate mechanism.
□ This is the Hibernate connection pool. This pool allows hibernate to get a connection to the database and this is transparent to the user. It is provided by an instance of `SessionFactory`.

□ This is a memory cache that references objects synchronized with the database or to be synchronized with the database.

Question 25 : In which case should the technique of “lazy-loading” of hibernate be called?

□ When it is not necessary to recover (SELECT), recording from the joins explained in the mapping files.
□ When there are too many columns in a table, we will prefer to return only the columns used in the application.
□ When it is desired to generate queries in SELECT one by one on the basis of the identifier of the record.

Question 26 : With Hibernate, how do we get the auto generated ID after the insertion of a record ?

□ We use the method `getGeneratedKeys` of `Statement`.
□ After insertion, performing a `get` or a `load` on the session Hibernate.
□ After insertion, performing a HQL query by adding criteria corresponding to the values of properties of the newly inserted record.
□ There is nothing special, the property representing the identifier will be valued by Hibernate.

Question 27 : What is the best solution to update a record with Hibernate ?

□ The Session object of Hibernate has a method `update` waiting in the parameter object to update.
□ The Session object of Hibernate has a method `modify` waiting in the parameter object to update.
□ Applying the changes on the object and wait for the flush usually generated on `commit()`.
□ Remove the current record with the `delete` method of the Hibernate Session and add the new record with the `save` method of the Hibernate Session.

Question 28 : In which case(s) is it not recommended to add the following tag in the hibernate.cfg.xml configuration file:

```
<property name="hbm2ddl.auto">update</property>
```

□ When there are records in the database incompatible with hibernate mappings.
□ When there are records in the database.
□ When it changes the dialect and the location of the data source.
□ When the database management is not the responsibility of developer.

Question 29 : What is the static method `initialize` of the `Hibernate` class for?

□ To disable the lazy-loading on the mapping of a column.
□ To initialize proxies returned by the lazy-loading.
□ Value objects and collections of objects held null because of the lazy-loading.
□ To initialize Hibernate by exploiting the configuration file hibernate.cfg.xml.
□ To request Hibernate to support Hibernate mappings and according to the value of hbm2ddl.auton generate or update the DDL of the database.

Spring

Question 30 : What is(are) the **objective(s)** of Spring light container?

□ Automate the instantiation of all types of classes without using the Java development.
□ Standardize the lifecycle of components from JEE framework.
□ Structure the application to allow a better maintainability, scalability and testability.
□ Provide a solution for the implementation of managed components that is independent of JEE including the EJB standard.
□ Easier integration of the controller layer by integrating the Java Servlets API.

Question 31 : What is the `<import>` tag of Spring for ?

□ To import the Spring library.
□ To import the Java classes instantiated by Spring.
□ To import files properties and allow to set dynamic values in the XML file.
□ Make use of the container additional files.

Question 32 : What is the declaration of a Conversion Service for ?

□ To convert the `<bean>` tag in Java object.
□ To convert the values declared as strings, regardless of the type of the target property.
□ To convert the values declared as strings when the target type is a class not supported by a PropertyEditor.
□ To convert the date and numeric formats.

Question 33 : Interpret the following code ?
```
<bean class=”tp.Example” p:name=”example”>
        <property name=”id” value=”12” type=”java.lang.String”>
</bean>
```

□ Instantiating an object of class `Example` through a constructor taking a parameter variable having the value "name" and assigning the value 12 of property “id” through the setter `setId (int)`.
□ Instantiating an object of class `Example` through a constructor without parameter, the name of this instance will be “`example`”, and assigning to the property “id” the value “12” through the setter `setId(String)`.
□ Instantiating an object of class `Example` through a constructor without parameter, the instance has a property “name” with the value “`example`”, and assigning to the property “id” the value “12” through the setter `setId(String)`.
□ Instantiating an object of class `Example` through a constructor without parameter, the name of this instance will be “`example`”, and assigning to the property “id” the value “12” through the setter `setId(int)`.
□ Instantiating an object of class `Example` through a constructor accepting two parameters, the first is a String value “example”, and the second is a String value “12”.

Question 34 : What does the `@Repository` annotation mean?

□ This is a stereotype that will qualify the component of the architecture as a DAO application.
□ This is a component of the application representing a instance of `DataSource`
□ This is an annotation to put over a property declaration or the setter of this property to inject a value of type `DataSource`.
□ This is a software layer enabling to put data from a database in cache for further usage

<u>Question 35</u> : What must we be careful when using Spring inheritance ?

□ The class of the inheriting bean must extend the class of the parent bean
□ The properties coming from the parent and injected by autowiring must be explicitly valued in the inheriting bean, unless the autowiring has been configured in the inheriting bean declaration as well
□ If the classes are inheriting in a Java sens, the constructor used to instanciate the parent bean must be added in the Java extending class
□ Give values to the parent bean properties aiming to have a value (even not the same) in the inheriting bean

<u>Question 36</u> : What is autowiring ?

□ To auto-detect the components of the application
□ To save lines of XML configuration
□ A enable support for the `@Resource` annotation
□ To make the configuration file more readable

<u>Question 37</u> : What is the `<prop>` tag for?

□ To externalize properties in a .properties file
□ The define the value or the reference to be injected in a bean property
□ The give a value to a bean property of type java.util.Properties.
□ This tag does not exist

<u>Question 38</u> : How does the annotation `@Resource` work?

□ If it is used with a “value” attribute, it is the bean having as id the provided value which will be injected. Else, Spring will use the autowiring byName based on the name of the property and finally byType based on the class of the property.
□ If it is used with a “value” attribute, it is the bean having as id or name the provided value which will be injected. Else, Spring will use the autowiring byName based on the name of the property and finally byType based on the class of the property.
□ If it is used with a “value” attribute, it is the bean having as id the provided value which will be injected. Else, Spring will use the autowiring byName based on the class of the property with a small letter for the first letter.
□ In any case, Spring will proceed with a autowiring byName based on the name of the property and finally byType based on the class of the property.

<u>Question 39</u> : What is the difference between `<context:component-scan>` and `<context:annotation-config>` ?

□ `<context:annotation-config`> indicates the Spring must interpret the `@Resource`

annotations, `<context:component-scan>` indicates the Spring must detect `@Component` annotations and the inheriting annotations
□ `<context:annotation-config` and `<context:component-scan>` come always together. We cannot use both separately. `<context:component-scan>` allows indicating where are located the @`Component` and inheriting annotated classes, `<context:annotation-config` allows injecting the dependecies in the classes.
□ `<context:component-scan>` allows detecting the `@Component` annotations. `<context:annotation-config` allows detecting all annotations specific to a software layer (`@Service`, `@Repository`, `@Controller`)

Misc.

<u>Question 40</u> : What does the acronym JPO : mean?

□ Java Persistant Object: It is the way to qualify a persistent object with Hibernate
□ Java Presentation Object: It is the way to qualify a JavaBean used by a JSP through the JSTL
□ Java Persistence Oriented: It is a Java architecture using an ORM framework
□ Maybe a finnish rock'n roll band ? Anyway not a concept studied in the course