Université d'Alger1 – Faculté des Sciences - Département MI
Module Techniques avancées en Architecture Logicielle
Examen – durée 1h00
M1 (2019/2020)

Exercise 1 (10 points):

a. **QCM** : Choisir la ou les bonnes réponses : **8 pts**

1- Laquelle des méthodes suivantes est utilisée pour stocker un objet dans un objet request d'une servlet?	
☐ addAttribute(String name, String obj)	☐ setAttribute(String name, String obj)
☐ addObject(String name, Object obj)	☒ setAttribute(String name, Object obj)
2- Dans une Servlet, quelle est la façon de répondre au client avec html ?	
☐ PrintWriter out = response.getOutputStream();	☐ ServletOutputStream out = response.getWriter();
☒ PrintWriter out= response.getWriter();	☐ OutputStream out = response.getOutputStream();
3- Quelles sont les balises qui devraient remplacer les <XXXXXXXXXX> ? <servlet> <servlet-name>Controller</servlet-name> <servlet-class>somepackage.Controller</servlet-class> <init-param> <XXXXXXXXXX>user</XXXXXXXXXX> <XXXXXXXXXX>username</XXXXXXXXXX> </init-param> </servlet>	
☒ <param-name> & <param-value>	☐ <parameter-name> & <parameter-value>
☐ <init-param-name> & <init-param-value>	☐ <init-name> & <init-value>
4- La declaration de cette servlet peut etre s'ecrit : - <servlet-mapping> <servlet-name>controlleur</servlet-name> <url-pattern>*.php</url-pattern> <url-pattern>/services</url-pattern> </servlet-mapping>	
☒ @WebServlet(name=" controlleur ", urlPatterns={"/services ","*.php"})	☒ @WebServlet(urlPatterns={"/services ","*.php"})
☐ @Servlet(patterns={"/services ","*.php"})	☐ @ urlPatterns ({"/services ","*.php"})
5- Tomcat est ...	
☒Un serveur web	☒ Un conteneur de servlets
☐ Un serveur d'application	☐ Une extension de JEE
6- Une page JSP est compilé en:	
☒ Une Servlet	☐ Un EJB stateless
☐ Un code javascript	☐ Aucun
7- Quel (s) langage (s) de programmation ou langage de script Java Server Pages (JSP) prend-il en charge?	
☐ VBScript	☒ Java
☐ Jscript	☐ Tous les éléments mentionnés

0, 5 pts
0, 5 pts
0, 5 pts
0,25 pts
0,25 pts
0,25 pts
0, 5 pts
0, 5 pts

Université d'Alger1 – Faculté des Sciences - Département MI
Module Techniques avancées en Architecture Logicielle
Examen – durée 1h00
M1 (2019/2020)

	8- Dans une JSP, comment pouvez-vous savoir quelle méthode HTTP (GET ou POST) est utilisée par la requête du client?
0, 5 pts	☒ En utilisant request.getMethod () ☐ En utilisant req.getMethod () ☐ En utilisant this.getMethod ()) ☐ Impossible de savoir
	9- Pour appeler la jsp depuis une servlet, nous utilisons...
0, 5 pts	☐ La méthode doGet() ☒ Un dispatcher ☐ La méthode doPost() ☐ La méthode callServlet()
	10- Les directives JSP permettent de :
0, 5 pts	☒ Inclure une partie dans une page ☐ Déclarer des variables ☐ Afficher des variables ☐ Implémenter des méthodes
	11- Qu'est-ce qui est correct sur les Scriptlets JSP ?
0, 5 pts	☐ Une boucle peut commencer dans un Scriptlet et se terminer dans un autre ☐ Le point-virgule est nécessaire à la fin de chaque déclaration dans un Scriptlet ☐ Les instructions dans un Scriptlet doivent suivre la syntaxe Java ☒Toutes les réponses sont vrais
	12- <C:OUT/> <C:IF/>, etc... Font partie ...
0, 5 pts	☐ Des expressions langages ☒ Du JSTL ☐ Du JSP ☐ Du HTML
	13- Avec le code suivant dans une JSP, comment afficher "Hello World" ? <% request.setAttribute("hello", "Hello"); String world = "World"; %>
0, 5 pts	☐ ${ hello } ${ world } ☐ <%= hello %> ${ world } ☒ ${ hello } <%= World %> ☐ <%= hello %> <%= world %>
	14- Que sortira le code JSP suivant ? <% request.setAttribute("msg", "Hola"); %> ${message}
0, 5 pts	☐ Hola ☐ une NullPointerException ☐ msg ☒ une chaine vide
	15- Que sortira le code JSP suivant ? <% request.setAttribute("msg", null); %> ${msg}
0, 5 pts	☐ null ☐ une NullPointerException ☐ msg ☒ une chaine vide
	16- Que sortira le code JSP suivant ? <html> <body> <% for (int i=0; i<3 ; i++){ %>out.print(i);<% } %></body></html>
0, 5 pts	☐ 0 1 2 ☐ une RuntimeException ☐ 0 1 2 3 ☒ out.print(i); out.print(i); out.print(i);

b. **Questions libres** : **(2 pts)**

1. Expliquer brièvement le cycle de vie d'une servlet dans son conteneur.
2. En JSP, quelle différence y a-t-il entre déclarer une variable dans un scriptlet et la déclarer dans une déclaration JSP ?

1. Cycle de vie d'une Servlet **1,5 pts**

a- chargement de la Servlet,

b-Création d'une instance de la Servlet (pour chaque Servlet, au cours de son cycle de vie, on a une et une seule instance créée),

c- Initialisation de la Servlet par appel de la méthode init() par le Conteneur web.

d- Une fois que la Servlet est initialisée, elle est prête pour exécuter les requêtes.

e-Pour chaque requête le conteneur génère une thread qui exécute la méthode service()

f-Quand le conteneur reçoit l'ordre de détruire la Servlet, il fait appel à la méthode destroy() de la Servlet.

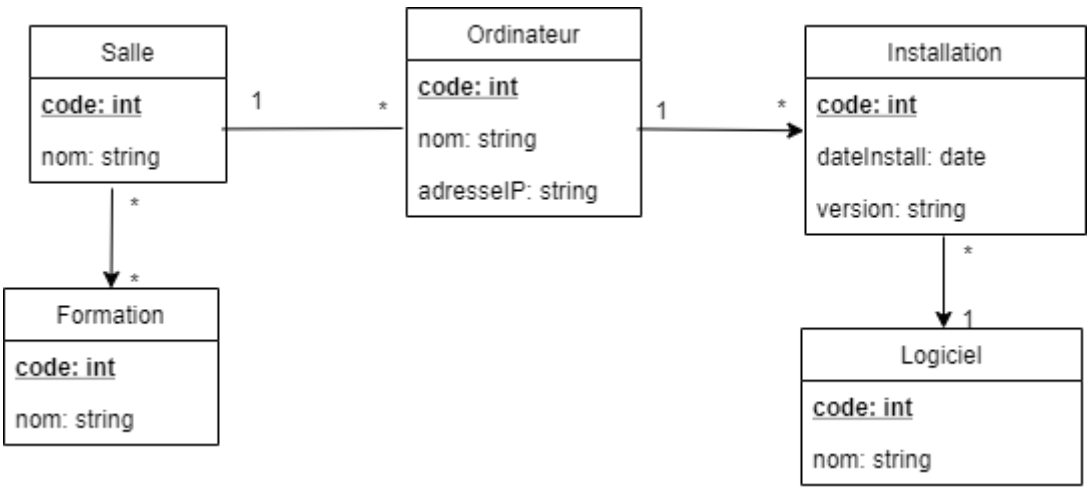
Les trois étapes (a, b, c) sont exécutées une et une seule fois; la quatrième étape (l'étape d) quant à elle, est exécutée à chaque fois qu'il y a une nouvelle requête. La dernière étape, quand elle est exécutée, la Servlet est définitivement détruite.

2. Dans un scriptlet, c'est une variable locale : elle sera dans le corps de la méthode de service de la servlet générée. Dans une déclaration, c'est une variable globale : ce sera une variable d'instance de la servlet. **0,5 pts**

Exercice 2 (10 points) :

On souhaite développer une application web JEE qui permet de gérer un parc informatique. Dans les salles du parc informatique se trouvent des ordinateurs. Un ordinateur est identifié par un code et se caractérise par son nom de machine ainsi que son adresse IP associé.

Différents logiciels sont installés sur les différents ordinateurs. Plusieurs versions d'un même logiciel peuvent être installées sur un même ordinateur. Enfin, les salles du parc informatique peuvent être utilisées pour plusieurs formations. La figure suivante décrit le digramme de classes proposé.



1. Compléter l'implémentation des classes suivantes de telle sorte qu'elles puissent être sauvegardées dans une base de données relationnelle en utilisant JPA. **4 pts**
2. Implémenter l'interface IMetier en utilisant les requêtes JPQL. **6 pts**

```
public interface IMetier {
        public void ajouterFormation(Formation f);
        //permet d'insérer une nouvelle formation dans le système
        public List<Ordinateur> consulterTousLesOrdinateurs();
        //permet d'afficher la liste de tous les ordinateurs gérés par le système
        public Long afficherNbreOrdinateursParSalle(String salle);
        //permet d'afficher le nombre d'ordinateurs dans une salle donnée
         public void modifierAdressIPOrdinateurs ()
        //permet de mettre l'adresse IP de tous les ordinateurs à localhost
        public List<Ordinateur> chercherDesOrdinateursParLogiciel(String log);
        //permet d'afficher la liste des ordinateurs où le logiciel log est installé
}
```

Les annotations	Les classes
@Entity	public class **Formation** implements Serializable **0,25 pts** {
@Id	private int code;
	private String nom;
	//Constructeur avec paramètres
	//constructeur sans paramètres **0,25 pts**
	// setters & getters **0,25 pts** }

0,25 pts
0,25 pts

@Entity	public class **Salle** implements Serializable {
@Id	private int code;
	private String nom;
@OneToMany(mappedBy="salle")	private List<Ordinateur> ordinateurs **0,25 pts**
@ManyToMany	private List<Formation> formations **0,25 pts**
	//Constructeur avec paramètres
	//constructeur sans paramètres
	// setters & getters }

0,25 pts
0,25 pts

@Entity	public class **Ordinateur** implements Serializable {
@Id	private int code;
	private String nom;
	private String adressIP;
@ManyToOne	private Salle salle; **0,25 pts**
@OneToMany	private List<Installation> logicielsInstalles **0,25 pts**
	//Constructeur avec paramètres
	//constructeur sans paramètres
	// setters & getters }
@Entity	public class **Installation** implements Serializable {
@Id	private int code;
@Temporal(TemporalType.TIMESTAMP)	private Date dateInstall;

0,25 pts
0,25 pts
0,25 pts
0,25 pts

	private String version;
@ManyToOne	private Logiciel logiciel;
	//Constructeur avec paramètres
	//constructeur sans paramètres
	// setters & getters }

@Entity	public class **Logiciel** implements Serializable {
@Id	private int code;
	private String nom;
	//Constructeur avec paramètres
	//constructeur sans paramètres
	// setters & getters }

2.

```
public class ImpMetier implements IMetier{
        @PersistenceContext("unitePersistence")    0,5 pts
        private EntityManager em;    0,5 pts
@Override
public void ajouterFormation(Formation f) {
                                                    0,5 pts
        em.persist(f);

}

@Override
public List<Ordinateur> consulterTousLesOrdinateurs() {
                Query req=em.createQuery("select o from Ordinateur o");    0,5 pts
                return req.getResultList();    0,5 pts
}
@Override
public Long afficherNbreOrdinateursParSalle(String salle);
                TypedQuery<Long> req=em.createQuery("select count(o) from Ordinateur o join
                o.salle s where s.nom= :nom", Long.class);    0,5 pts
                req.setParameter("nom", salle);    0,25 pts
                return req.getSingleResult();    0,25 pts
}
@Override
Public void modifierAdressIPOrdinateurs () {
0,5 pts em.createQuery("update Ordinateur set adressIP= :adr").setParameter("adr",
"localhost").executeUpdate();    0,5 pts
}
@Override
public List<Ordinateur> chercherDesOrdinateursParLogiciel(String log) {
                Query req=em.createQuery("select o from Ordinateur o join o.logicielsInstalles ins
                join ins.logiciel log where log.nom= :l");    1 pts
                req.setParameter("l",log);    0,25 pts
                return req.getResultList();    0,25 pts
}
```