|  |  |
| --- | --- |
|  |  |
|  | C:\Users\hamadi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\logo_inti_vector-04.jpg  Institut des Nouvelles Technologies  de l’Information |

QCM

Qualimétrie

Session 2016

## Guide d'utilisation

* Notez qu'en fonction de la question, il est possible de cocher une ou plusieurs réponses. Ce sera à vous de déterminer le nombre de bonnes réponses possibles.

## Quelques conseils de mise en œuvre

* Ne restez pas bloqué sur une question pour laquelle vous n'avez pas de réponse. Passez à la question suivante. Vous reviendrez par la suite sur les questions sans réponse.
* Identifiez les questions auxquelles vous avez répondu sans certitude. Que ces réponses soient justes ou non, cela vous permettra de mémoriser les questions sur lesquelles vous devrez revenir dans la correction du TEST.
* Ne cherchez pas à aller le plus vite possible. Le temps imparti doit être utilisé au maximum pour vous permettre d'effectuer une synthèse des points abordés dans le module.

**QCM**

**1. Dans la gestion des dépendances avec Maven, quelles sont les trois balises qu’il faut ?**

* artifactId, groupId, version
* id, name, version
* repository, className, version
* namespace, library, version

**2. Quel *scope* nécessaire pour la compilation non déployée faut-il choisir pour une librairie ?**

* compile
* provided
* application
* session

**3. Avec l’arborescence des répertoires Maven, où doivent se trouver nos classes java ?**

* main/classes
* src/java
* src/main/java
* src/resources/java

**4. Dans Maven, où se trouvent toutes les dépendances et configurations de *build* d’un projet ?**

* Dans le Project Organization Model
* Dans le maven­config.xml
* Dans le Programe Organigram Metadata
* Dans le Project Object Model

**5. Que faut-il y inscrire pour spécifier le type d’archive générée pour le projet ?**

* <archive>web</archive>
* <packaging>war</packaging>
* <output>jar</output>
* <compile>web</compile>

**6. Que doit-on indiquer dans la *javadoc* d’une méthode ?**

* Une description de la méthode
* Une description des paramètres de celle-ci
* Une description de sa valeur de retour
* Les trois

**7. Comment décrire un argument d’une méthode dans la *javadoc* ?**

* @nomParam Type description
* @param nomParam description
* @argument nomParam description
* Type nomParam description

**8. Selon les conventions de codage *Oracle*, quelle est le souci avec la méthode suivante ?**

public void setTruc(String truc) {

this.truc = truc;

}

* Il manque des espaces entre les parenthèses et le paramètre
* Le nom du paramètre ne devrait pas être le même que l’attribut
* Le setter ne fait rien d’autre qu’une assignation sans contrôles
* Les trois

**9. Quel est le principe d’un test unitaire ?**

* Tester une méthode dans son environnement
* Tester une méthode dans un environnement contrôlé et reproductible
* Tester une méthode de manière isolée et contrôlée
* Tester une méthode et ses dépendances de manière isolée

**10. Faut-il tester plusieurs fois la même méthode ?**

* Non, un seul test suffit
* Oui, une fois pour un succès et une fois pour un échec
* Oui, il faut tester les différentes utilisations de la méthode, “bonnes” et “mauvaises”
* Non, on fait tout dans le même test

**11. Avec *jUnit*, comment tester que la méthode *calculate()* renvoie 3 si on lui donne 2 en argument ?**

* calculate(2).assertEquals(3);
* Assert.assertEquals(3, calculate(2) );
* Assert.calculate(2, 3);
* assert(calculate(2) == 3);

**12. Comment définir les méthodes à exécuter dans les tests avec *jUnit* ?**

* Avec l’annotation @ToTest sur les méthodes
* Avec l’annotation @Test sur la classe contenant les méthodes
* Avec l’annotation @Test sur les méthodes
* Toutes les méthodes dont le nom commence par Test seront exécutées

**13. Quelle annotation ajouter à une méthode qui devra s’exécuter avant chaque test ?**

* @SetUp
* @Before
* @BeforeClass
* @BeforeTest

**14. Qu’est-ce qu’un *mock* ?**

* Un type de test unitaire qui prend en compte les dépendances
* Une classe qui émule la classe à tester en supprimant ses dépendances
* Une classe qui ira se substituer à une dépendance de la classe à tester
* Une classe contenant des tests unitaires

**15. Quel principe de programmation permet l’utilisation d’un *mock* ?**

* L’Architecture Distribuée
* Le Test Driven Development
* La Service Oriented Architecture
* L’injection de dépendances

**16. Y a-t-il généralement un grand intérêt à faire un test unitaire sur un DAO ?**

* Oui, dans tous les cas
* Oui, s’il comporte beaucoup d’algorithmie
* Non, ça reviendrait à tester un mock
* Non, car c’est impossible

**17. Comment s’appelle le moteur de test unitaire en *javascript* ?**

* Mantra
* Chakra
* Karma
* Lama

**18. Avec *Jasmine*, comment déclare-t-on les méthodes de test ?**

* function test() { }
* Avec une annotation @Test sur les fonctions
* it(‘description du test’, function(){ });
* test(‘description du test’, function(){ });

**19. Avec *Jasmine*, comment créer l’équivalent d’un *mock* ?**

* var mock = mock(objet);
* var spy = spy(objet);
* spyOn(objet, ‘methode’);
* mock(objet, methode);

**20. Qu’est-ce que le TDD ?**

* Une méthode de programmation préconisant d’écrire des tests pour 100% des méthodes d’une application
* Un framework de test java
* Une méthode de programmation où les tests unitaires sont écrits avant le code source
* Aucun des trois

FIN