QCM UML corrigé

Q1. Que veut dire UML?

	Union mondiale de la lecture.
x	Unified modeling language.
	Unité mesure libre

Q2. L'agrégration est-elle un type d'association? :

x	Oui
	Non

Q3. Une composition est-elle un type d'agrégration?

x	Oui
	Non

Q4. Que signifie la multiplicité 1..*? :

	Plusieurs incluant la possibilité d'aucun
	Exactement 1
	Au plus un
x	Au moins un

Q5. Une action qu'un objet peut réaliser s'appelle :

x	Une opération
	Une classe
	Un attribut
	Une formule

Q6. Qu'est ce qu'une relation dite include?

	Le cas d'utilisation A est réalisé avant le cas d'utilisation B.
x	Le cas d'utilisation A toujours besoin du cas d'utilisation B.
→	La relation include indique qu'un cas a toujours besoin d'un autre cas d'utilisation lié
	Le cas d'utilisation A peut éventuellement avoir besoin du cas d'utilisation B.
	Le cas d'utilisation B peut éventuellement avoir besoin du cas d'utilisation A

Q7. Qu'est ce qu'une relation dite extend?

	Le cas d'utilisation A est réalisé avant le cas d'utilisation B.
	Le cas d'utilisation A toujours besoin du cas d'utilisation B.
x	Le cas d'utilisation A peut éventuellement avoir besoin du cas d'utilisation B.
→	La relation extend est une relation qui est soumise à une condition

	Le cas d'utilisation B peut éventuellement avoir besoin du cas d'utilisation A

Q8. Que doit-on faire lors de l'analyse de besoins principaux d'un projet logiciel ?

	Définir toutes les informations nécessaires du futur logiciel.
X	Découvrir les acteurs et les fonctionnalités du futur logiciel.
	Décrire une fonctionnalité du futur logiciel.
	Définir les packages ou les grandes parties du logiciel à créer.

Q9. Dans un diagramme de classes en langage UML, la généralisation :

X	Est une relation transitive : si C dérive d'une classe B qui dérive elle-même d'une classe A, alors C dérive également de A
	Est une relation réflexive : une classe peut dériver d'elle-même
	Est une relation symétrique : si une classe B dérive d'une classe A, alors la classe A peut dériver de la classe B
	Représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité

Q10. En Java, une association (dans un diagramme de classes UML) est-elle implémentée par ?

X	une variable d'instance
	une opération
	une variable de classe
	un constructeur

Q11. Un rôle (dans un diagramme de classes UML) se traduit-il en Java par ?

X	un nom de variable
	une association
	un constructeur
	une opération

Q12. Des véhicules sont dotés de châssis et de propulsions. Différents types de châssis permettent aux véhicules de rouler, voler ou bien flotter. Différents types de propulsions leur permettent d'avancer grâce au vent ou bien grâce à un moteur. Toutes les combinaisons de véhicules sont possibles : une voiture roule avec un moteur ; un planeur vole avec du vent etc. Au minimium, combien de classes sont nécessaires pour représenter les véhicules avec toutes leurs déclinaisons, si on n'utilise **que l'héritage** ? (en incluant la classe Véhicule)

	7
x	8
	9
	10

Q13. Même problème que la question précédente, mais **en utilisant des interfaces** combien de classes/interfaces sont nécessaires pour représenter les véhicules avec toutes leurs déclinaisons (en incluant la classe Véhicule) ?

	7
X	8
	9
	10

La question n'est pas tant le nombre de classes, les modélisations peuvent différer mais sur l'impact du choix des interfaces pour les combinaisons (combinaisons statiques avec l'heritage, dynamiques avec les interfaces)

Q14. Quel(s) type(s) de relations sont permises entre des cas d'utilisation ? (plusieurs choix possibles)

	Association
X	Généréralisation / héritage
X	Dépendance de type "includes"
X	Dépendance de type "extends"
	Aucune de ces réponses n'est correcte.

Q15. Queel est le modificateur d'accès le plus contraignant ?

X	private
	public
	protected

Q16. Entre une classe Vehicule et une classe Roue, quel type de relation est adéquate ?

X	Composition
	Association
	Héritage
	Agrégation

Q17. Entre une classe Vehicule et une classe Conducteur, quel type de relation est adéquat ?

	Composition
X	Association
	Héritage
	Agrégation

Q18. Entre une Classe Véhicule et une classe Bâteau, quel type de relation est adéquat ?

	Composition
	Association
X	Héritage
	Agrégation

Q19. Considérons une association entre une classe Client et une classe Commande. Quelle multiplicité mettriez-vous du côté de Commande ?

	0..1
X	0..*
	1..*

	1..1

Q20. Considérons une association entre une classe Client et une classe Commande. Quelle multiplicité mettriez-vous du côté du Client ?

	0..1
	0..*
	1..*
X	1..1

Q21. Grâce à une relation d'héritage, de quoi hérite la classe enfant ? (plusieurs choix possibles)

X	des opérations
X	des propriétés
X	des associations
X	des relations d'héritage
X	des attributs

QCM JAVA

Q1. Combien d'instances de la classe A crée le code suivant ?

```
A x,u,v;
x=new A();
A y=x;
A z=new A();
```

	Aucune
	Cinq
	Trois
X	Deux

Il y a deux instances de A créées par les deux new, la première est référencée par x et y, la deuxième par z.

Q2. Pour la classe B définie comme suit:

```
class B {
        public B() {
                System.out.print("Ciao");
        }

        public B(int i) {
                this(); System.out.println("Bonjour "+i);
        }

}
```

qu'affichera l'instruction suivante?

```
B monB=new B(2003);
```

	erreur de compilation
	erreur d'exécution
X	CiaoBonjour 2003
	Bonjour 2003

L'instruction invoque le constructeur avec un argument entier (2003). Ce dernier appelle explicitement le constructeur sans arguments (this()) qui imprime "Ciao", et ensuite le message "Bonjour 2003" est imprimé

Q3.

X	Une classe peut implémenter plusieurs interfaces mais doit étendre une seule classe
	Une classe peut étendre plusieurs classes mais ne peut étendre qu'une seule interface
	Une classe peut implémenter plusieurs classes et peut étendre plusieurs interfaces
	Une classe peut implémenter une seule interface et ne peut étendre qu'une seule classe

Q4. Etant donnée que la classe Grande étend la classe Petite, trouvez une ligne correcte parmi les suivantes :

	Petite y =new Petite(); Grande x= (Grande)y; Petite z=x;
	La deuxième affectation Grande x= (Grande)y;

	essaye de transformer un objet (r´ef´erenc´e par y) de la classe Petite vers un objet de sa sous-classe Grande. Un tel downcasting est impossible.
X	Grande x= new Grande(); Petite y = x; Grande z=(Grande)y;
	Tout va bien. On crée un objet de classe Grande référencé par x. Ensuite on fait une variable y (de type Petite) référencer le même objet – c'est un upcasting explicite qui est toujours possible. A la fin on fait encore une référence z (cette fois Grande) sur ce même objet. Ce dernier downcasting est possible parce que l'objet est en fait une instance de la classe Grande
	Grande x= new Grande(); Petite y = x; Grande z=y;
	C'est presque comme dans le cas pr´ec´edent, mais la dernière affectation Grandez=(Grande)y; est un downcasting implicite, ce qui est interdit.
	Petite y =new Petite(); Grande x= (Grande)y; Petite z=(Petite)x;
	Grande x=(Grande)y; est un downcasting impossible, comme dans le (a)

Q5. Pour la classe C définie comme suit:

```
class C{
        public static int i=0;
        private int j;
        public C(){
                i++; j=i;
        }
}
```

qu'affichera le code suivant ?

```
public static void main(String[] args){
        C x=new C(); C y=new C(); C z= x;
        System.out.println(z.i + " et " + z.j);
}
```

	2 et 2
	1 et 1
X	2 et 1
	1 et 3

On remarque d'abord, que i est une variable (statique) de classe commune à toutes les instances, tandis que chaque objet de la classe a son propre j. Donc, après la première affectation on a i=1, x.j=1; après la deuxième : i=2, y.j=2 (x.j est resté inchangé et égal à 1); la troisième n'appelle pas le constructeur mais fait référencer z au mˆeme objet que x. D'o`u z.i est la valeur globale de i, c-`a-d 2, et z.j=x.j=1.

Q6. On se donne les classes et interfaces suivantes :

```
interface Propulsion{
        public void bouger() ;
}
```

Q6.1 Quel est ou quels sont les codes qui provoquent une ou plusieurs erreurs :

	public class Moteur implements Propulsion {
	public void bouger(){
	// faire quelque chose...
	}
	}

X	public class Moteur implements Propulsion{
	public void ronronner(){
	// faire quelque chose...
	}
	}

La 2e doit implémenter la méthode bouger()

Q6.2 on suppose la classe Moteur correcte et on ajoute la classe suivante (supposée correcte) :

```
class Vent implements Propulsion{
        //....
}
```

Quel est ou quels sont les codes qui provoquent une ou plusieurs erreurs (0, 1 ou plusieurs réponses possibles) :

a)
 	
X
	public class Vehicule{
	private Propulsion p ;
	public void setPropulsion(){p = new Propulsion() ;}
	}

b)	public class Vehicule{
	private Propulsion p ;
	public void setPropulsion(){p = new Moteur() ;}
	}

c)	public class Avion extends Vehicule{
	public Avion(){setPropulsion(new Moteur()) ;}
	}

d)
 	
X
	public class Avion extends Vehicule{
	public Avion(){p=new Vent() ;}
	}

e)
 	
X
	public class Avion extends Vehicule{
	public Avion(){setPropulsion(new Propulsion()) ;}
	}

g)
 	
X
	public class Avion extends Vehicule{
	public Avion(){p=new Moteur() ;}
	}

h)	public class Avion extends Vehicule{
	public Avion(){setPropulsion(new Vent()) ;}
	}

a) : on ne peut instancier une interface
d) : p est un attribut hérité mais privé
e) idem a)
g) idem d)