

Questions de cours.

1-Donner l'utilité des mots réservés suivants?

<u>Mots</u>	<u>Utilité</u>
throws	
interface	
static	

2-Expliquer en quelques lignes les rôles des blocs d'initialisation statiques en Java ?

3-Compléter le tableau suivant par la visibilité de chaque modificateur d'accès ?

<u>Modificateur</u>	<u>Visibilité</u>
default	
public	

4-Détailler en quelques lignes les notions suivantes :

Notion	Définition	Exemple en java
Redéfinition		
Co-variance		
Cast		

5-Détailler les rôles des méthodes suivantes en donnant un exemple ?

<u>Méthode</u>	<u>Rôles</u>
public boolean equals(Object o)	
public int compareTo(Object o)	

6-Compléter le tableau suivant par la description des collections suivantes ?

<u>Collection</u>	<u>Description</u>
HashMap	
TreeMap	
Set	

Exercice 1 :

En utilisant le langage Java :

Créer une classe Citoyen définie par l'ID (entier), Nom (Chaine de caractère), Cin (Chaine de caractère) et Age (Réal) ?

Créer les accesseurs pour chaque attribut ?

Créer deux constructeurs pour la classe Citoyen ? Vous avez le choix pour les paramètres.

Créer la méthode main dans la classe Citoyen ?

Instancier deux objets de la classe Citoyen ? Vous avez le choix pour les arguments.

Exercice 2:

Soit la classe Etudiant et les classes Test1, Test2 et Test3 suivantes. Donner le contenu de chaque collection issue de l'exécution des classes Test1, Test2 et Test3.

```
class Etudiant implements
    Comparable<Etudiant>{
    private String nom;
    private int age;
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public int hashCode() {
        return age;
    }
    @Override
    public boolean equals(Object obj) {
        Etudiant e = (Etudiant) obj;
        return age == e.getAge();
    }
    public Etudiant(String nom, int age) {
        this.nom = nom;
        this.age = age;
    }
    @Override
    public String toString() {
        return "[nom=" + nom + ", age=" + age + "]";
    }
    @Override
    public int compareTo(Etudiant o) {
        return 0;
    }
}
```

```
public class Test1 {
    public static void main(String[] args) {
        Set<Etudiant> s = new HashSet<Etudiant>();
        s.add(new Etudiant("AHMED", 20));
        s.add(new Etudiant("Omar", 20));
        s.add(new Etudiant("Said", 10));
        s.add(new Etudiant("Badr", 20));
        System.out.println(s);
    }
}
```

Le contenu de la collection « s » de Test 1 est :

.....
.....
.....
.....

```
public class Test2 {
    public static void main(String[] args) {
        Set<Etudiant> s = new TreeSet<Etudiant>();
        s.add(new Etudiant("AHMED", 20));
        s.add(new Etudiant("Omar", 20));
        s.add(new Etudiant("Said", 10));
        s.add(new Etudiant("Badr", 20));
        System.out.println(s);
    }
}
```

Le contenu de la collection « s » de Test 2 est :

.....
.....
.....
.....

```
public class Test3 {
    public static void main(String[] args) {
        List<Etudiant> l = new List<Etudiant>();
        l.add(new Etudiant("AHMED", 20));
        l.add(new Etudiant("Omar", 20));
        l.add(1,new Etudiant("Said", 10));
        l.add(new Etudiant("Badr", 20));
        System.out.println(l);
    }
}
```

Le contenu de la collection « L » de Test 3 est :

.....
.....
.....
.....
.....

Exercice 3:

Donner l'affichage issu de l'exécution des codes java suivants :

```
public class Local {
    private int nbLocaux;
    private String description;
    private int surface;
    public int getNbLocaux() {
        return nbLocaux;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public Local(int nb, String description) {
        this(description);
        nbLocaux += nb;
        this.description = description;
    }
    public Local(String description) {
        this(description,100);
        this.description += description;
    }
    public Local(String description, int surface) {
        this.description += description;
        this.surface += surface;
    }
    public int getSurface() {
        return surface;
    }
    public void setSurface(int surface) {
        this.surface = surface;
    }
}
```

```
public class Test1 {

    public static void main(String[] args) {
        Local loc1 =new Local(10, "A");
        Local loc2 =new Local(20, "B");
        System.out.println(loc1.getDescription());
        System.out.println(loc1.getNbLocaux());
        System.out.println(loc1.getSurface());
    }
}
```

L'affichage issu de l'exécution de Test1 :




.....
.....
.....
.....

```
public class Test2 {

    public static void main(String[] args) {
        Local loc1 =new Local("A",10 );
        Local loc2 =new Local("B",20);
        System.out.println(loc1.getDescription());
        System.out.println(loc1.getNbLocaux());
        System.out.println(loc1.getSurface());
    }
}
```

L'affichage issu de l'exécution de Test2 :

.....
.....
.....
.....

 	<p>Examen « Java de base ». Année scolaire : 2015/2016. Documents <u>non autorisés</u> Filière ABDTW – V1 La durée : <u>1H30min</u></p>	 <p>UNIVERSITÉ HASSAN 1^{er} FACULTÉ DES SCIENCES ET TECHNIQUES DE SETTAT</p>
---	--	---

	<pre>public class Test3 { public static void main(String[] args) { Local loc1 =new Local(1000,"S"); System.out.println(loc1.getDescription()); System.out.println(loc1.getNbLocaux()); System.out.println(loc1.getSurface()); } }</pre>
	<p><u>L'affichage issu de l'exécution de Test3 :</u></p> <p>.....</p> <p>.....</p> <p>.....</p> <p>.....</p> <p>.....</p>

Exercice 4:

En utilisant le langage java :

- 1-Créer une classe Connexion en respectant les règles du patron de conception Singleton.
- 2-Donner un exemple d'utilisation du patron de conception Factory.