+1/1/60+

NOM et Prénom :

.....

Numéro d'étudiant :

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

Université Grenoble Alpes
UFR IM2AG
UE GBIN6U03
Première Interrogation
Année 2016-2017

Consignes pour répondre :

- Noircir entièrement les cases, pas de croix ou de simple trait qui barre la case
- l'usage du blanc pour effacer une réponse est possible, mais ne pas redessiner la case si vous débordez (laisser le contour blanc)
- ne pas barrer une réponse, utiliser du blanc ou demander une nouvelle copie et recommencer

Durée : 45mn
Tout document interdit. Calculatrices, ordinateurs, téléphones interdits.
La réponse aux questions n'est pas forcément unique. Pour certaines questions il ne faut remplir qu'une seule case, pour d'autres il faut en remplir plusieurs.

Question 1 Pour allouer un tableau de 42 entiers :

- ☐ `int []T; T.length = 42;`
- ☐ `int []T; T = new int[42];`
- ☐ `int T[]; T[42] = new int;`
- ☐ `int T[42];`
- ☐ *Aucune de ces réponses n'est correcte.*

Question 2 Lorsqu'il y a dépassement des bornes d'un tableau :

- ☐ c'est au programmeur de lever explicitement l'exception, sinon il y a une erreur à la compilation
- ☐ une exception est automatiquement levée
- ☐ ce n'est pas détecté, comme en C
- ☐ ce n'est pas possible, c'est détecté à la compilation
- ☐ *Aucune de ces réponses n'est correcte.*

Question 3 Par défaut, en l'absence de qualificateur de visibilité, un attribut est visible :

- ☐ uniquement dans le paquetage qui le contient
- ☐ uniquement dans la classe qui le contient
- ☐ uniquement dans la classe qui le contient et ses descendantes
- ☐ partout
- ☐ *Aucune de ces réponses n'est correcte.*

Question 4 Qu'est-ce qui ne va pas dans le code suivant :

```
class File {
    int taille;
    ...    }
...
File f;
System.out.println("Taille : " + f.taille);
```

- ☐ aucun objet n'est alloué
- ☐ on ne peut pas afficher taille
- ☐ taille n'est pas public
- ☐ taille n'est pas statique
- ☐ *Aucune de ces réponses n'est correcte.*

Question 5 Pour appeler une méthode m sur un objet o :

- ☐ `o(m)`
- ☐ `m(o)`
- ☐ `m.o()`
- ☐ `o.m()`
- ☐ *Aucune de ces réponses n'est correcte.*

Question 6 Sans contexte particulier, une méthode statique peut :

- ☐ appeler une méthode statique
- ☐ être appelée depuis une méthode statique
- ☐ appeler une méthode non statique
- ☐ être appelée depuis une méthode non statique
- ☐ *Aucune de ces réponses n'est correcte.*

Question 7 Quel code(s) est(sont) correct(s) ?

- ☐
```
int tab[10];
for (int i=0; i<tab.length; i++)
    tab[i] = 0;
```
- ☐
```
int [] tab = new int[10];
for (int i=0; i<tab.length; i++)
    tab[i] = 0;
```
- ☐
```
int [] tab = new int[10];
for (int i=0; i<=tab.length; i++)
    tab[i] = 0;
```
- ☐
```
int tab[10];
for (int i=0; i<=tab.length; i++)
    tab[i] = 0;
```
- ☐ *Aucune de ces réponses n'est correcte.*

Question 8 Sachant que `NoSuchElementException` hérite de `RunTimeException`, retrouvez la première ligne du code suivant :

```
{
if (taille > 0)
    return tab[--taille];
else
    throw new NoSuchElementException("Pile vide");
}
```

- ☐ `int depile()`
- ☐ `int depile(NoSuchElementException e)`
- ☐ `int depile() throws NoSuchElementException`
- ☐ `int depile() throws RunTimeException`
- ☐ *Aucune de ces réponses n'est correcte.*

Question 9 Examinez le code suivant :

```
try {
    in = out = null;
    in = new Scanner(new File("Fichier42.txt"));
    out = new PrintStream("Resultat42.txt");
} catch (FileNotFoundException e) {
    System.err.println("Erreur de fichier");
} catch (Exception e) {
    System.err.println("Erreur inconnue");
} finally {
    if ((in != null) && (out == null)) in.close();
}
```

Combien de lignes (sans compter celles du bloc try/catch/finally) sont-elles exécutées par ce code ?

- ☐ 5
- ☐ 3
- ☐ 4 ou 5
- ☐ 3 ou 4
- ☐ 4

Question 10 Combien de fichiers sont-ils ouverts par ce code ?

- ☐ 1 ou 2
- ☐ 2
- ☐ 1
- ☐ 0 ou 2
- ☐ 0

Question 11 Une interface est :

- ☐ un moyen de communiquer une spécification
- ☐ un contrat engageant un objet qui l'implémente à définir tous ses méthodes
- ☐ une méthode servant à accéder à un attribut d'un objet
- ☐ un objet permettant d'appeler une méthode d'un autre objet depuis une de ses propres méthodes
- ☐ *Aucune de ces réponses n'est correcte.*

Question 12 Dans l'exemple suivant, que peut être le type `Ensemble` ?

```
Ensemble f = new EnsembleTableau();
```

- ☐ une classe abstraite
- ☐ une interface
- ☐ une classe concrète
- ☐ une classe anonyme
- ☐ *Aucune de ces réponses n'est correcte.*

Question 13 Quelle(s) propositions est/sont vraies ?

- ☐ tout héritage doit être eplicité
- ☐ toute classe hérite de la classe `Object`
- ☐ si une classe hérite d'une autre classe, on dit qu'elle spécialise la classe parent
- ☐ tout héritage est implicite
- ☐ *Aucune de ces réponses n'est correcte.*

Question 14 La surcharge permet de définir une méthode ayant le même nom qu'une autre méthode du moment que l'une des conditions suivantes est vérifiée :

- ☐ son type de retour change
- ☐ le type d'un ou plusieurs de ses paramètres change
- ☐ son nombre de paramètres change
- ☐ les exceptions qu'elle est susceptible de lever changent
- ☐ *Aucune de ces réponses n'est correcte.*

Question 15 Pour une classe A, hériter d'une autre classe B signifie

- ☐ avoir un type utilisable partout ou le type de B est utilisable
- ☐ avoir au moins les même attributs que B
- ☐ avoir au moins les même méthodes que B
- ☐ pouvoir redéfinir des méthodes ou attributs de B
- ☐ *Aucune de ces réponses n'est correcte.*

Qu'affiche le code suivant ?

```
class A {
    void affiche() { System.out.print("A"); }
}
class B extends A {
    void affiche() { System.out.print("B"); }
}
public static void main(String args[]) {
        A a = new A();
        A b = new B();
        B c = new B();
        a.affiche();
        b.affiche();
        c.affiche();
```

Question 16

- ☐ AAB
- ☐ BBB
- ☐ AAA
- ☐ ABB
- ☐ *Aucune de ces réponses n'est correcte.*

Question 17 Une classe abstraite :

- ☐ doit définir une ou plusieurs méthodes abstraites
- ☐ ne peut pas définir de méthode
- ☐ hérite des méthodes de sa classe parent
- ☐ peut définir des méthodes concrètes
- ☐ *Aucune de ces réponses n'est correcte.*

Question 18 Au sujet de la généricité :

- ☐ il n'est pas possible d'avoir des tableaux génériques
- ☐ Un type générique peut être instancié avec n'importe quel type
- ☐ une classe générique est nécessairement abstraite
- ☐ la généricité consiste à paramétrer du code par une classe
- ☐ *Aucune de ces réponses n'est correcte.*

Question 19 Quel code(s) est(sont) correct(s) ?

☐
```
class Toto<E> {
    Object [] t;
    Toto() { t = new E[10]; }
    E get(int i) { return t[i]; }
}
```

☐
```
class Toto<E> {
    E [] t;
    Toto() { t = new E[10]; }
    E get(int i) { return t[i]; }
}
```

☐
```
class Toto<E> {
    E [] t;
    Toto() { t =
        (E []) new Object[10]; }
    E get(int i) { return t[i]; }
}
```

☐
```
class Toto<E> {
    Object [] t;
    Toto() {
        t = new Object[10]; }
    E get(int i) {
        return (E) t[i]; }
}
```

☐ *Aucune de ces réponses n'est correcte.*

Question 20 Qu'est-ce qui constitue un paquetage ?

- ☐ le regroupement d'interfaces et de classes dans un même répertoire
- ☐ un fichier .java regroupant une classe, ses méthodes, ses attributs
- ☐ n'importe quel fichier qui ne comporte pas de main
- ☐ l'instanciation d'une classe générique
- ☐ *Aucune de ces réponses n'est correcte.*

Question 21 Une fabrique abstraite sert à :

- ☐ fabriquer des objets
- ☐ fabriquer des instances de classes abstraites
- ☐ fabriquer des objets de classes anonymes
- ☐ fabriquer des fabriques
- ☐ *Aucune de ces réponses n'est correcte.*

Question 22 Qu'est-ce qui constitue une utilisation incorrecte d'un itérateur dans le code suivant ?

```
Iterateur it = l.iterateur();
while (it.aProchain()) {
    Iterateur it2 = l.iterateur();
    int element = it.prochain();
    while (it2.aProchain()) {
        int element2 = it2.prochain();
        if (element == element2) it.supprime();
    }
}
```

- ☐ on ne peut pas appeler une méthode du premier itérateur dans une boucle utilisant le second
- ☐ la liste est changée durant son parcours par un itérateur
- ☐ deux itérateurs sur la même liste sont définis
- ☐ le second itérateur démarre au début de la liste
- ☐ *Aucune de ces réponses n'est correcte.*

Question 23 Que fait le code suivant ?

```
Iterateur it = l.iterateur();
while (it.aProchain()) {
    int element = it.prochain();
    num[element] = 0;
    Iterateur it2 = l.iterateur();
    while (it2.aProchain()) {
        int element2 = it2.prochain();
        if (element == element2) num[element]++;
    }
    if (num[element] > 1) it.supprime();
}
```

- ☐ il supprime tous les éléments d'une liste
- ☐ il supprime les éléments égaux contigus d'une liste
- ☐ il supprime les doublons d'une liste
- ☐ il compte les éléments distincts d'une liste
- ☐ *Aucune de ces réponses n'est correcte.*

Question 24 Pour remplir correctement son rôle de visiteur, étant donné une hiérarchie de classes dont les objets à visiter sont les instances, la classe abstraite `Visiteur` doit forcément disposer d'une méthode `visite` :

- ☐ pour chacune des classes de la hiérarchie
- ☐ seulement pour l'ancêtre commun à toute la hiérarchie
- ☐ seulement pour les classes de la hiérarchie n'ayant aucun descendant
- ☐ seulement pour les classes de la hiérarchie ayant un descendant
- ☐ *Aucune de ces réponses n'est correcte.*

Question 25 En reprenant la même spécification que dans les TPs, que fait le code suivant ?

```
Visiteur v = new Visiteur() {
    boolean visite(Balle b) {
        return true;
    }
}
```

- ☐ il crée un visiteur qui supprime tous les objets qu'il visite
- ☐ il crée un objet de la classe `Visiteur`
- ☐ il crée une classe anonyme
- ☐ il crée un visiteur qui ne fait rien
- ☐ *Aucune de ces réponses n'est correcte.*