RECOMMANDATIONS POUR PASSER ET REUSSIR UN EXAMN EN LIGNE UTILISANT LES QUESTIONS A REPONSES COURTES DANS DES MODULES BASES SUR UN LANGAGE DE PROGRAMMATION

Par : Djamal BENNOUAR, Enseignant au département d'Informatique de l'Université de Bouira

Un des grands objectifs des examens en ligne utilisant les réponses ouvertes (courtes ou longues), est de déterminer si l'étudiant donne une grande importance à ses réponses et prends le temps nécessaire pour les écrire de manière très rigoureuse, en contrôlant avec attention tout caractère écrit. Une réponse comportant un caractère en plus, même un espace, indique que l'étudiant ne s'est pas bien concentré pour élaborer sa réponse et devra donc assumer le résultat obtenu.

Pour élaborer ses réponses, il est fortement recommandé à l'étudiant d'écrire et de préparer ses réponses en utilisant **un éditeur de texte** ou un **traitement de texte**. Un traitement de texte permettrait en plus de déterminer les éventuelles erreurs d'orthographe. Dans le cas de réponse comportant un code dans un langage de programmation comme JAVA, **il est fortement recommandé d'écrire le code dans le contexte d'un environnement de développement tel qu'Eclipse ou NetBean**. Ces environnements permettent à l'étudiant de détecter les éventuelles erreurs de syntaxe et certaines erreurs sémantiques.

Dans un examen en ligne ou les réponses sont principalement des portions de code dans un langage de programmation, l'étudiant est tenu d'appliquer de manière très rigoureuse les recommandations suivantes

1. Utiliser < et > au lieu de { et }

Pour des raisons technique, dues à une faiblesse dans le logiciel d'évaluation automatique des réponses, si vous êtes amené à utiliser les accolades dans votre réponse, il faut les remplacer par les caractères < et >. Le caractère < remplacera { et le caractère > remplacera }

Exemple :

Au lieu d'ecrire

public boolean possedeUnesolution(){return true;}

il faut écrire

public boolean possedeUnesolution()<return true;>

Au lieu d'écrire:

public static boolean tabDesCodes[]={12,24,36,48,60} ;

il faut écrire

public static int tabDesCodes[]=<12,24,36,48,60>;

2. Nombre de déclarations et instructions par ligne

Sauf indication contraire, mettre

- Une et une seule instruction par ligne
- Une et une seule déclaration par ligne
- Les instructions de contrôle **if** , **while**, **for** représente chacune une instruction et doivent etre écrites seule sur une ligne. Si le bloc d'instruction contient plus d'une instruction, la ligne contenant le **if** , **while**, **for** doit se terminer par une accolade ouvrante (dans notre cas c'est < au lieu de {)

3. Type à utiliser obligatoirement

SAUF indication contraire, utiliser obligatoirement

- le type **double** pour les réels (**ne pas utiliser float**)
- le type **int** pour les entiers (**ne pas utiliser byte, short, long**)
- Pour les listes, utiliser obligatoirement **ArrayList<TypeDuContenu>**
  - Exemple
    - **ArrayList<Personne>** listPersonne ;
    - **ArrayList<Complex>** listComplex ;

4. Cas des réponses numériques

Sauf indication contraire, lorsque la réponse est numérique, il faut reporter juste la valeur

Exemple : si la réponse complète est **1024 Octets** écrire seulement **1024**

5. Noms des paramètres de constructeur ou de méthode et leur ordre d'apparition et traitement

Lorsque le constructeur doit comporter **des paramètres qui correspondent aux attributs** définis dans la classe,

- Les paramètres doivent avoir le même nom que les attributs correspondant
- Les paramètres doivent apparaître dans la zone des paramètres dans le même ordre d'apparition des attributs correspondant dans la définition de la classe
- Les instructions qui traitent les attributs doivent être citées selon l'ordre d'apparition des attributs dans la définition de la classe.

6. Désignation des attributs d'un objet d'une classe et accès aux attributs d'une classe

- Sauf indication contraire, pour désigner les attributs d'un objet, que ce soit au niveau d'un constructeur ou d'une méthode, vous devez obligatoirement utiliser le mot clé **this**.

Exemple :

```
class  Complex{
   double i, r ;
   public Complex(double i, double r){
      this.i  = i ;
      this.r  = r;
    }
    void  setI (double i){
      this.i = i ;
   }
}
```

7. Accès aux attributs d'un objet d'une autre classe (Spécifique à la POO)

Lorsque une méthode veut accéder en lecture ou écriture à un attribut d'**une autre classe**, même si les 2 classes sont dans le même package et même si l'attribut concerné est déclaré comme public, la méthode doit obligatoirement utiliser les accesseurs **get** et **set**.

8. Choix du nom de variable et de références

Lorsqu'aucune indication n'est fournie sur le nom de variable ou de référence à utiliser, vous devez suivre le style suivant pour définir le nom des variables et des références

- si vous déclarer une seule référence, utiliser comme nom de référence le même nom que la classe, en mettant en minuscules le premier caractère Majuscules du nom de la classe. La seule différence entre le nom de la classe et le nom de la référence réside au niveau du premier caractère qui est en Majuscule pour le nom de la classe et est en minusculepour le nom de la référence.

  Exemple : **Personne personne;**
  Article article ;
  JButton jButton ;
  ASDTypeDeBase aSDTypeDeBase
  IHM iHM ;

- si vous déclarer un seul tableau d'objets, utilisez comme nom de référence du tableau le mot **tab** suivi du nom de la classe (le caractère qui suit **tab** doit etre en Majuscule):

  Exemple **Personne[ ] tabPersonne;**
  Article[ ] tabArticle;
  int[] tabInt
  double[] tabDouble

- si vous déclarer une seule liste (**utiliser obligatoirement ArrayList<….>, voir recommandation 7**) , utiliser comme nom de référence de la liste le mot **list** suivi du type de données des objets de la liste (attention le mot **list** est sans **e**)

  Exemple ArrayList<**Personne**> **listPersonne;**
  ArrayList<**Article**> **listPersonne = new** ArrayList<**Article**>**();**

- Si plusieurs références d'un même type de données doivent être déclarées :
  - Ajouter après le nom de la référence ou variable un numéro séquentiel, qui commence à 1 et qui est augmenté de 1 pour chaque nouvelle déclaration de référence ou variable
  - Il faut déclarer une et une seule référence ou variable par ligne ;

  Exemple **Personne personne1**:
  Personne personne2:
  Complex complex1;
  Complex complex2;
  Complex complex3;
  Personne[] tabPersonne1;
  Personne[][] tabPersonne2;
  Personne[] tabPersonne3;
  Article[] tabArticle1;
  Article[] tabArticle1;

```
int[] tabInt1;
int[]tabInt2;
int[] tabInt2;
double[] tabDouble1;
double[] tabDouble2;
double[] tabDouble2;
```

- Si vous devez déclarer plusieurs références de divers types structurés ou tableau, l'une à la suite de l'autre, procéder comme suit
  - Considérer les types structurés par ordre alphabétique de leurs noms,
  - Une et une seule déclaration par ligne
  - Commencer par les références de variable structurées puis les tableaux
  - Au niveau de chaque catégorie (variable structurée, tableaux), déclarer par ordre alphabétique

Exemple:
```
Article a ;
ArticleDeLuxe[] tabArticleDeLuxe ;
Complex c1;
Complex c2;
Complex[] tabComplex;
Personne personne;
Personne[] tabPersonne1;
Personne[] tabPersonne2;
```

9. Parcours de tableaux

SAUF indication contraire, dans les exercices qui nécessitent le parcours de tableaux: Utiliser obligatoirement l'identificateur i pour la variable de parcours des lignes et l'identificateur **j** pour la variable de parcours des colonnes. Pour l'incrémentation de 1 (ou décrémentation de 1) utiliser obligatoirement ++ (ou --).

10. Style d'Ecriture de fonctions, classes et position des accolades

- Une fonction vide s'écrit sur une et une seule ligne.

Exemple : **public void setCoordonnee(int x, int y)<> // Attention : utiliser < au lieu de { et > au lieu de }**
public double delta()<>

Les 2 accolades montrant un corps vide sont essentielles

- Une classe vide s'écrit sur une seule ligne
  - Exemple :
    - **class Article<>**
    - **class ArticleDeLuxe extends Article<>**
    - **class Dessin extends JPanel implements MouseLIstener<>**
- Lors de l'écriture de plusieurs lignes successives d'une fonction,
  - Ecrire une seule instruction par ligne.
  - **Mettre la première accolade ouvrante (ici c'est le caractère <) de la classe, ou d'une fonction sur la même ligne que le nom de la classe ou de la fonction.**
    Exemple :
    ```
    public class ArticleDeLuxe extends Article<
    public int somme(int nb, int debut)<
    ```
  - **Toute accolade fermante doit être mise seule dans une ligne.**
  - Mettre l'accolade ouvrante d'un bloc à la fin de l'instruction de contrôle (**if** , **while**, **for)** qui précède le bloc.
    Exemples
    - if (a==6&&b==12)<
    - while (x!=10)<
    - for(int i=100;i >=0;i++)<
  - Ne pas oublier d'utiliser les caractères représentant les accolades (< pour { et > pour })

11. Pas de délimitation de bloc if, while, for, do ayant seulement une instruction

Utiliser les accolades (donc > et <) lorsque le bloc **if, while, for, do** contient plus d'une instruction. **Cependant Si le bloc contient une seule instruction, n'utiliser pas les accolades et une seule instruction par ligne**

Exemple 1 :
```
while (i < 100)
    i = i+1 ;
```

Exemple 2 :

```
while (a+b < 120)
    if (a > 70)
        return a%2 +5;
```

Exemple 3 :

```
for  (int a=100 ; a <  200; a=a+5)
        if (a%3 == 0 )
            return a;
```

12. Le else et le bloc du else

- N'utiliser le **else** que lorsque il devient obligatoire.

  Exemple :

```
If (a==100)
    return true ;
else    // ici il n'est pas obligatoire.
     return false;

Il faut écrire
If (a==100)
   return true ;
return false;
```

- Lorsque le bloc du **else** contient une seule instruction, celle-ci doit être écrite sur la ligne qui suit e **else**.

Exemple :

```
If (a > b)
  c = a ;
else
  c = b;
```

- Si le bloc **else** contient plus d'une instruction, la ligne du **else** se termine par une accolade ouvrante et les instructions seront mise dans les lignes qui suivent

Exemple

```
If (a > b)
   c = a ;
else {
   c = b;
   b = a ;
}
```

13. Style d'écriture de la déclaration avec initialisation de tableau 1D et 2D

- Pour tableau 1 dimension : sur 1 seule ligne :

  Exemple : **int[]a=<1, 67,32,6>;**

- Pour tableau 2 Dimension : plusieurs lignes
  - Première ligne contient la déclaration de la référence. Cette ligne doit se terminer par **<** (c'est {)
  - Chaque ligne qui suit contiendra un tableau 1 Dimension composant le tableau 2D
  - La dernière ligne contiendra le caractère > (représente }) suivi du point virgule

    Exemple

```
int[]x=<
<1,2,4,6>,
<10,20,40,60>,
<12,22,42,62>
>;
```

14. Transformation d'un diagramme de classe (POO):

- Lorsque le bout de la relation comporte une constante entière plus grande que 1 ou un intervalle dont la limite supérieure est une constante entière plus grande que 1 (exemple 2..7, 0..18), **il faut utiliser un tableau** dont la taille est égale à la constante ou à la borne supérieur de l'intervalle.

- Lorsque le bout de la relation comporte une étoile (nombre inconnu) (exemple *) ou un intervalle dont la borne supérieure n'est pas connue (exemple 4..*), il faut obligatoirement utiliser une ArrayList. (Attention : il ne faut pas utiliser LinkedList dans les tests en ligne, sauf indication explicite)
- Pour écrire le contenu de la classe, suivre l'ordre suivant :
    - Commencer les attributs représentants les relations par ordre alphabétique croissant du nom de la classe de l'attribut.
    - Déclarer ensuite les attributs cités dans la notation graphique de la classe selon l'ordre de leur apparition dans la notation graphique
    - Définir les constructeurs et méthodes par ordre de leur apparition dans la notation graphique
- Ecrire sur une seule ligne une classe vide, un constructeur vide, une méthode ordinaires vide.

15. Les opérateurs à utiliser en priorité

- Pour une incrémentation ou décrémentation de 1, utiliser ++ ou - - et ne pas utiliser + et -. Par exemple au lieu d'écrire **a=a+1 ;** écrire **a++ ;**
- Ne pas utiliser les opérateurs combiné : += -= *= /= %=
- Lorsque une variable est associée à une constante dans une opération, commencer par la variable. Par exemple écrire a*2 au lieu de 2*a et écrire (a+b)*c + 15 au lieu de 15 + (a+b)*c
- Dans une expression commencer par écrire la sous expression qui se trouve en parenthese. Ecrire (x+y)*a au lieu de a*(x+y)
-

16. Les messages en rouges

Faire très attention aux messages écrits en rouge. Il faut bien les lire. Ils peuvent contenir d'autres règles à respecter ou modifier localement une règle définie dans ce document

17. En cas de situations non prévues

Si vous faites face à une situation non couverte par les recommandations de ce document, appeler immédiatement l'enseignant responsable du cours

18. Orthographe

Ne pas utiliser les accents

Si vous avez un problème d'orthographe lors de l'écriture de votre réponse consulter l'enseignant chargé de suivre l'examen en ligne