DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

SIGLE DU COURS: IFT 1010 (H001)

NOM DU PROFESSEUR: Philippe Langlais

TITRE DU COURS: Programmation I

EXAMEN INTRA

Date : 22 février
Heure : 8:30 - 10:30
Salle : 0030 / 1360

DIRECTIVES PÉDAGOGIQUES:

- Toute documentation est permise.
- **Inscrivez tout de suite votre nom et code permanent dans l'encadré.**
- Répondez **sur le questionnaire** en utilisant l'espace libre qui suit chaque question.

NOTATION:

Exercice 1:	/20	A)	B)	C)	D)	
Exercice 2:	/30	A)	B)	C)	D)	E)
Exercice 3:	/20					
Exercice 4:	/30					
Total:	/100					

Inscrivez votre nom et votre code permanent ici

CONSEIL:
Les deux premiers exercices contrôlent vos connaissances générales de Java. N'y passez pas un temps déraisonnable (vous connaissez la réponse ou pas). Les deux exercices qui suivent vérifient votre aptitude à la programmation structurée (prenez le temps de réfléchir).

Exercice 1 (20 points):

Dans cet exercice, il vous est demandé de trouver des erreurs de compilation et/ou des erreurs d'exécution. Les erreurs de compilation sont des erreurs qui rendent impossible la compilation (javac). Les erreurs logiques ou erreurs d'exécution sont des erreurs qui ne sont pas détectées à la compilation, mais qui se manifestent à l'exécution par un comportement non prévu.

Les exemples de code qui sont donnés ne sont pas mis en contexte (dans une classe particulière), ceci afin de réduire la taille de l'énoncé, les erreurs qu'on vous demande de trouver ne sont pas liées à ce fait.

Question A (5 points)

Ce code comporte une erreur qui empêche sa compilation. Expliquez clairement cette erreur.

```
boolean res;
if (i > 4) res = true;
else if (i<4) res = false;

if (res) System.out.println(i + " est superieur a 4 ");
else System.out.println(i + " n'est pas superieur a 4");
```

Solution: L'erreur vient du fait que *res* n'est pas initialisée dans le cas où *i* vaut 4. JAVA refuse la compilation dans une telle situation. La réponse consistant à dire que *i+* n'est pas déclarée n'est pas la bonne, compte-tenue de la remarque faite en début d'exercice.

Question B (5 points)

Ce code contient une erreur logique. Expliquez l'erreur et corrigez-là directement dans le code.

```
int somme = 0;
for (int n=0; n<=10; n++)  if ( (n % 2) == 0) somme += n;
System.out.println("1 + 3 + 5 + 7 + 9 = " + somme);
```

Solution: Ce programme calcule la somme 0+2+4+6+8+10. Plusieurs solutions possible. La plus simple consiste à changer *==* dans le test par *!=*.

Question C (5 points)

Quel est l'affichage provoqué par l'exécution de ce code ?

```
int x= 6;
int y= 4;
if (x > 5)
    if (y>5) System.out.println("x et y sont supérieurs à 5");
else
  if (y<5) System.out.println("x et y sont inférieurs à 5");
```

Solution: x et y sont inférieurs à 5

Question D (5 points)

Quels affichages sont produits par les instructions suivantes ?

```
System.out.println(" 3 * 4 = " + 3 * 4);
System.out.println(" 3 + 4 = " + 3 + 4);
System.out.println(" A = " + (3 / 2 * 4 -2) );
System.out.println(" B = " + ((14<15)? (2 + 7/8 - 1 - 2): 4));
System.out.println(" C = " + 4 % 2 + 3 );
```

Solution: 3 * 4 = 12
Solution: 3 + 4 = 34
Solution: A = 2
Solution: B = -1
Solution: C = 03

Exercice 2 (30 points):

Répondez à chaque question sachant les deux déclarations suivantes:

```
float f;
int i;
```

Question A (5 points)

Rayez les instructions qui provoquent une erreur de compilation.

```
f = 3;                 OK
f = 3.0;               ERREUR (double vers float)
i = 3.5;               ERREUR (double vers int)

i = 10 / 2.0;          ERREUR (double vers int)
i = (int) 10 / 2.0;    ERREUR (double vers int)
```

Explication: Les deux dernières instructions mettent en jeu des conversions implicites. Le résultat de la division de 10 par 2.0 est un résultat avec décimale (5.0), ce qui ne peut être rangé dans un int. Dans la seconde division, le cast est prioritaire sur la division. Il ne s'applique donc qu'à 10 (qui est déjà un entier). La division en donc encore une division entre un entier et un double. Le résultat est donc encore un double. La bonne solution aurait été d'écrire: i = (int) (10 / 2.0);

Question B (5 points)

Quelle est la valeur de `i` après cette affectation ?

```
i = (int) Math.random() * 15 + 1;
```

Solution: 1

Explication: Encore un problème de priorité entre l'opérateur de cast et les autres opérateurs en présence. Math.random() retourne un double entre 0 et 1 (non inclus). En prenant sa partie entière, on obtient nécessairement 0 auquel est ensuite ajouté 1.

Écrivez une instruction qui affecte à `i` une valeur entière entre 4 et 17 (bornes incluses):

Solution: (par exemple) i = (int) (Math.random()*14+4);

Explication: Math.random() retourne une valeur dans [0,1[. En la multipliant par 14, on a une valeur dans [0,14[en lui ajoutant 4, on a une valeur dans [4,18[. Il ne reste qu'à prendre la partie entière.

Question C (5 points)

Indiquez à même le code (directement après l'instruction), l'affichage produit par ce code. Si une instruction contient une erreur de compilation ou une erreur d'exécution, rayez-là et indiquez à coté (de manière concise) la nature de l'erreur.

```
String s = "bonjour";
System.out.println(s.charAt(0));              b
System.out.println(s.length());               7
System.out.println(s.charAt(s.length()));     PROVOQUE UNE ERREUR À l'EXÉCUTION
```

Explication: L'indice du premier caractère d'une chaîne est 0; l'indice du dernier caractère est length()-1. L'erreur est liée au fait que l'on demande le caractère d'indice 7 qui n'existe pas (indices de 0 à 6).

Question D (10 points)

Indiquez l'affichage produit par ce code:

```
int y=5, x=4;
System.out.println(y++);
if ((x++ > 2) || (++y == 4)) System.out.println("A");
else System.out.println("B");
System.out.println("x=" + x + " y=" + y);
```

Solution:
5
A
x=5 y=6

Explication: y++ est une post-incrémentation. Elle ne prend effet qu'après le premier affichage. Dans le test, x++ ne prend effet qu'après le test, mais de toute facon x est égal à 4, donc supérieur à 2. Le premier test est donc vrai et la partie droite du test n'est pas évaluée (donc y ne change pas). Aussi bien x que y ont été finalement incrémentés une seule fois chacun. Il valent donc respectivement 5 et 6.

Question E (5 points)

La méthode suivante est succeptible de provoquer une erreur d'exécution. Indiquez pourquoi et proposez une correction qui ne change pas la sémantique du code.

```
void methode(int i) {
  if ((i != 0) || ( (15/i) == 3)) System.out.println(i);
}
```

Solution: Comme toujours lorsqu'il y a une division, il y a risque de division par 0, ce qui provoquerait une erreur d'exécution. Ici le risque est réel car si i vaut 0 en entrée de la méthode, alors le premier test (i != 0) est faux et le second est évalué (ce qui provoque l'erreur). Une solution possible consiste à reécrire l'instruction de la manière suivante:

```
  if ((i != 0) || ((i != 0) && ((15/i) == 3) ))
    system.out.println(i);
```

Explication: Vous pouvez également remarquer que le second test ne sert à rien et que si i est différent de 0, alors il vous suffit d'afficher directement i.

Exercice 3 (20 points):

Quel est l'affichage produit par la méthode `exercice3` ?

```
void exercice3() {
   int c;
   final int MAX = 10;

   for (int l=0; l<=MAX/2; l++) {
     for (c=0; c<l; c++) System.out.print(".");
      System.out.print(l);
      while (c<= (MAX-l)) {
         if (c == (MAX/2)) System.out.print(l);
         else System.out.print(".");
         c++;
      }
      System.out.print(l);

      while (c <= MAX) {
         System.out.print(".");
         c++;
      }
      System.out.println();
   }
}
```

0.....0.....0
.1....1....1.
..2...2...2..
...3..3..3...
....4.4.4....
.....555.....

Votre réponse

Exercice 4 (30 points):

Écrivez un programme qui demande à un utilisateur de saisir deux chaînes de caractères (resp. s1 et s2) et qui indique à l'utilisateur si la première chaîne (s1) est une sous-chaîne de la seconde (s2) ou pas. Si s1 est une sous-chaîne de s2, alors votre programme doit également indiquer l'indice (dans s2) à partir duquel commence s1.[1]

Voici trois exemples d'exécution de ce programme:

```
Entrez une chaîne: au
Entrez une chaîne: saumon
La chaîne "au" est une sous-chaîne de "saumon".
Elle commence à la position 1

Entrez une chaîne: auk
Entrez une chaîne: saumon
La chaîne "auk" n'est pas une sous-chaîne de "saumon".

Entrez une chaîne: mon
Entrez une chaîne: saumon
La chaîne "mon" est une sous-chaîne de "saumon".
Elle commence à la position 3
```

Note: Les seules méthodes de la classe *String* que vous devez/pouvez utiliser sont:

`char charAt(int i)` qui retourne le caractère d'indice `i` d'une chaîne
`int length()` qui retourne la longueur de la chaîne

En d'autres termes, n'utilisez pas de méthodes du genre `indexOf` c'est précisément ce qu'on vous demande de programmer !

Solution: L'usage d'une méthode n'était pas demandé. Il existe plein de solutions possibles. Toutes sont basées sur une boucle imbriquée. La boucle la plus extérieure parcourt une chaine tandis que la seconde boucle parcout la seconde tant que les caractères de chacune des chaînes sont égaux. Si la seconde boucle s'effectue jusqu'au bout, alors la secodne chaîne est incluses dans la première. Voir le code pour plus de commentaires.

[1] Si s1 est contenue plusieurs fois dans s2, alors l'indice retourné est l'indice de première apparition de s1 dans s2.

```
import Keyboard;

class Exercice4 {

  // regarde si sc est contenue dans s.
  // si tel est le cas, retourne l'indice de depart de sc dans s
  // sinon, retourne -1

  static int existe(String s, String sc) {
        int j;

        // pour chaque indice i dans s, verifier si sc n'est pas
        // dans s a partir de l'indice i
        for (int i=0; i<=(s.length()-sc.length()); i++) {
            // tant que egalite caractere par caractere, continuer le
            // passage sur la seconde chaine
            for (j=0; (j<sc.length()) && (s.charAt(i+j) == sc.charAt(j)); j++);

            // si tous les caracteres de sc ont ete trouves
            // alors on a trouve sc dans s. Sc commence a l'indice i
            if (j == sc.length()) return i;
        }

        // si on arrive ici, alors c'est que sc n'est pas dans s
        return -1;
    }

  public static main(String [] args) {

    System.out.print("Entrez une chaine:");
    string a = Keyboard.readString();

    System.out.print("Entrez une chaine:");
    string b = Keyboard.readString();

    int indice = existe(a,b);
    if (indice == -1)
      System.out.println("La chaîne \"" + b  + '"'+
                         "n'est pas une sous-chaîne de \"" + a + '"');
    else {
     System.out.println("La chaîne \"" + b  + '"'+
                         "est une sous-chaîne de \"" + a + '"');
     System.out.println("Elle commence à la position "+ indice);
    }

  } //main
}// class
```