

DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

SIGLE DU COURS: IFT 1010 (A2001)

NOM DU PROFESSEUR: Pierre Poulin

TITRE DU COURS: Programmation I

EXAMEN INTRA

Date : 22 octobre 2001

Heure : 17:30 - 19:30

Salle : Z110

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.
- Attention, il y a des questions des deux côtés des feuilles.

NOTATION:

Question 1:	/12
Question 2:	/14
Question 3:	/5
Question 4:	/9
Question 5:	/5
Question 6:	/20
Question 7:	/15
Question 8:	/20
Total:	/100

Solutionnaire

Inscrivez votre nom et votre code permanent ici

CONSEIL:

Les cinq premières questions contrôlent vos connaissances générales de Java. N'y passez pas un temps déraisonnable (vous connaissez la réponse ou pas). Les trois dernières questions vérifient votre aptitude à la programmation structurée (prenez le temps de réfléchir).

Question 1 (12 points):

Dans les instructions qui suivent, supposez que toutes les classes et méthodes utilisées sont bien définies. Expliquez le problème principal avec chaque série d'instructions qui suivent.

```
double cout = 5.0;    // cout au Canada
if (pays.equals("Canada"))
    if (province.equals("Quebec"))
        cout = 2.5;    // cout au Quebec
else
    cout = 10.0;        // cout pour les autres pays
```

Le else ici se rattache au deuxième if. Donc si le pays est Canada mais la province est autre que Quebec, la variable cout aura la valeur 10.0. Pour tout autre pays que le Canada, la variable cout aura la valeur 5.0. Ceci n'est pas en accord avec les commentaires.

```
public void Racine (double a, double b, double c)
{
    double d, r;
    if ((d = b * b - 4. * a * c) >= 0.)
        r = Math.sqrt (d);
    System.out.println ("discriminant: " + d + " racine: " + r);
}
```

La variable r ne sera pas initialisée si la variable d est négative. Cette méthode ne compilera pas.

```
public int signe (double x)
{
    if (x < 0) return -1;
    if (x > 0) return +1;
}
```

La méthode signe ne retourne rien si la variable x a la valeur 0. Cette méthode ne compilera pas.

Question 2 (14 points):

De quel type de base doit être x pour être valide dans l'expression

```
x = ( x == x )
```

boolean

Après chaque instruction `System.out.println`, indiquez la valeur qui sera imprimée si elle est exécutée, ou simplement si cette instruction ne compilera pas.

```
int i = 3;          System.out.println((i++ + ++i));
```

8

```
int j = 3;          System.out.println((++(++j)));
```

Ne compile pas.

```
float f = 3;        System.out.println((-f-- - - --f));
```

-2.0

```
char c = 'a';       System.out.println(c.charAt(0));
```

Ne compile pas.

```
int k = 3;          System.out.println((1 <= (int) (k/3.0) <= 3) ? 1 : 3);
```

Ne compile pas.

```
int m = 3, n = 2;   System.out.println(((m == n) == (m == n)));
```

true

Question 3 (5 points):

Dans les instructions qui suivent, encerclez tous les points-virgules (;) qui produiront une erreur de syntaxe:

```
; if ( n % 2 == 1 )  
  
    { ; n = n + 1 ; } ;  
  
else ;  
  
    { ; n = n / 2 ; } ;
```

Seulement le ; devant le else cause une erreur de syntaxe.

Question 4 (9 points):

Expliquez pourquoi se produisent les différences entre ces trois évaluations:

```
float a = 1;  
float b = 100000000; // 100 millions  
System.out.println ("1: " + b + 5);  
System.out.println ("2: " + (b + 5));  
System.out.println ("3: " + (b + a + a + a + a + a));
```

donne

```
1: 1.0E85  
2: 1.00000001E8  
3: 1.0E8
```

1: Concaténation entre un float (1.0E8) et un entier (5).

2: Evaluation de b+5 en float. Les 7 chiffres significatifs du float font que 100000005 est arrondi à 100000010, i.e. 1.00000001E8.

3: Chaque évaluation de b+a s'effectue en float. L'arrondissement à 7 chiffres significatifs produit 100000001: 1.0E8, et donc la valeur après l'addition ne change pas.

Question 5 (5 points):

Dites ce qu'imprimera exactement (les retours de chariot sont importants) un appel à la méthode `PetiteMethode ()`.

```
public void PetiteMethode ()
{
    int i = 0;
    // petite boucle
    for (; i++ < 3; );
    System.out.print (i);
    /* deuxieme petite boucle
    for (i = 0; i < 3; i++)
        System.out.print (i);
    /* presque fin petite methode */
    System.out.println(++i);
}
```

45(retour de chariot)

Question 6 (20 points):

Ecrivez les instructions de la méthode `ImprimeX()` pour qu'elle affiche un X de la forme spécifiée ci-dessous pour toute valeur positive de `nbLignes`.

```
1: X      2: XX      3: X X      4: X  X      5: X   X      ...
           XX          X          XX          X  X
                   X X          XX          X
                           X  X      X  X
                               X  X      X  X
                                   X   X
```

```
public void ImprimeX (int nbLignes)
{
    for (int i = 0; i < nbLignes; i++)
    {
        for (int j = 0; j < nbLignes; j++)
        {
            if (j == i || j == nbLignes-i-1 )
                System.out.print ("X");
            else
                System.out.print (" ");
        }
        System.out.println ();
    }
}
```

Question 7 (15 points):

Ecrivez les instructions de la méthode `EvaluePi()` pour qu'elle calcule et retourne une valeur approximative pour π selon la formule de Leibnitz:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$$

Comme il s'agit d'une série infinie, assurez-vous que son évaluation se termine lorsque la valeur de type double pour π ne change plus.

```
public double EvaluePi ()
{
    // multiplie chaque terme par 4 pour que le test en type double
    // sur la valeur de pi soit coherent. Le test sur pi/4 est aussi
    // acceptable.

    double valeurPi = 4.;
    double ancienneValeurPi = 0.;
    int signe = -1;
    int denominateur = 3;

    // signe et denominateur pourraient aussi etre declarees en double
    // pour eviter les conversions de type et le risque des divisions entieres

    do
    {
        ancienneValeurPi = valeurPi;
        valeurPi += signe * (4. / denominateur);
        signe = -signe;
        denominateur += 2;
    } while (Math.abs (valeurPi - ancienneValeurPi) > 0.);

    return (valeurPi);
}
```

Question 8 (20 points):

Ecrivez une classe Carte qui ne contient seulement que deux méthodes. La première méthode est un constructeur qui génère aléatoirement une valeur initiale pour la carte parmi toutes les valeurs possibles. Vous n'avez pas à vous soucier que le jeu ait 52 cartes uniques. L'autre méthode, toString(), retourne une String telle que: (as, deux, trois, ... , dix, valet, dame, roi) suivie d'un trait d'union (-) et de son atout (coeur, carreau, trèfle, pique). Par exemple: "dix-pique", "as-carreau". Même si votre constructeur ne devrait générer que des valeurs valides, assurez-vous que dans la méthode toString(), si la valeur de la carte n'est pas valide, il faut retourner dans la String "carte non valide". Utilisez la méthode Math.random() pour générer une carte aléatoirement.

```
public class Carte ()
{
    private int valeur;
    private int atout;

    public Carte ()
    {
        valeur = (int) (Math.random () * 13) + 1;
        atout = (int) (Math.random () * 4) + 1;
    }

    public String toString ()
    {
        String strValeur, strAtout;
        boolean carteValide = true;

        // on pourrait aussi se contenter de faire un return de la string
        // non valide a la place, mais plusieurs return a differents endroits
        // est moins recommandable...

        switch (valeur)
        {
            case 1:
                strValeur = "as";
                break;
            case 2:
                strValeur = "deux";
                break;
            case 3:
                strValeur = "trois";
                break;
            case 4:
                strValeur = "quatre";
                break;
            case 5:
                strValeur = "cinq";
                break;
```

```

case 6:
    strValeur = "six";
    break;
case 7:
    strValeur = "sept";
    break;
case 8:
    strValeur = "huit";
    break;
case 9:
    strValeur = "neuf";
    break;
case 10:
    strValeur = "dix";
    break;
case 11:
    strValeur = "valet";
    break;
case 12:
    strValeur = "dame";
    break;
case 13:
    strValeur = "roi";
    break;
default:
    strValeur = "valeur inconnue";
    carteValide = false;
    break;          // pas necessaire, mais bonne habitude
}

switch (atout)
{
case 1:
    strAtout = "coeur";
    break;
case 2:
    strAtout = "carreau";
    break;
case 3:
    strAtout = "trefle";
    break;
case 4:
    strAtout = "pique";
    break;
default:
    strAtout = "atout inconnu";
    carteValide = false;
    break;          // pas necessaire, mais bonne habitude
}

```

```
    }

    if (carteValide)
        return (strValeur + "-" + strAtout);
    else
        return ("carte non valide");
}

}
```