Université Paris Diderot – Paris 7
Introduction à l'informatique et à la programmation (IF1)
Corrigé de l'examen du 7 janvier 2008

Exercice 1. *Conversion Farenheit/Celsius*

```
i port fr.  ssie .script.*
c ass Con ersion {
    p   ic static  oid  ain(String[  args) {
       do   e ce si s  farenheit  // f oat est ega e ent possi  e
       De g.print(" e perat re en degres Farenheit ? ");
       farenheit = De g.readDo   e();
       // po r o tenir  a te perat re en degres Ce si s  on retranche 3
       // p is on    tip ie par  ;
       //  e parenthsage et  e type do   e de  a  aria  e farenheit
       // ass re q e  es oprations sont faites s r des do   e
       ce si s = ((farenheit - 3) * 5 / 9;
       De g.print n("La te perat re est " + ce si s + " degres Ce si s");
    }
}
```

Exercice 2. *Calcul (simplifié) de l'impôt sur le revenu en 2008*

```
i port fr.  ssie .script.*
c ass  pots 00  {
   p   ic static  oid  ain(String[  args) {
       do   e re en   parts  i pots  reste
       int a atte ent  i posa  e
       int qf  res  tat
       De g.print("Re en  dec are ? ");
       re en  = De g.readF oat();
       De g.print("No  re de parts ? ");
       parts = De g.readF oat();
       a atte ent =(int) (re en / 10);
       i posa  e = (int) (re en  - a atte ent);
       qf = (int) (i posa  e / parts);
       i pots = 0
       if (qf > 8 2)
          if (qf > 113 ) {
              i pots += 66 * 0.055;
              if (qf >  5 ){
                  i pots += 13 1 * 0.1
                  if (qf > 6 6 ) {
                       i pots +=  3 1 * 0.3
                       reste = qf - 6 6 6 + 1
                       i pots += reste * 0.
                       }
                  e se {
                       reste = qf -  5 + 1
                       i pots += reste * 0.3
                       }
                   }
              e se {
                  reste = qf - 113  + 1
                  i pots += reste * 0.1
                   }
              }
          e se {
              reste = qf - 8  + 1
              i pots += reste * 0.055
               }
       res  tat = (int) (i pots * parts);
       De g.print n("\nMontant de  'i pot : " + res  tat);
    }
}
```

Exercice 3. *Autour de l'indicateur de chemins de fer*

3.1. *Affichage en clair d'une heure exprimée en secondes écoulées depuis 0h*

```
p   ic static  oid he re(int date) {
    if(date < 0 || date >= 1 0) De g.print n("he re incorrecte");
    e se {
       int  in tes = date%60
       De g.print(date/60 + "h");
       if( in tes < 10) De g.print("0" +  in tes);
       e se De g.print( in tes);
    }
}
```

3.2. *Recherche du premier train partant après une heure donnée*

```
p   ic static int prochain rain(int[]  depart  int[]  arri ee  int date) {
//  e para etre arri ee est in ti e
    if(date  < 0 || date >= 1 0) ret rn -1
    for(int train  = 0  train < depart. ength  train++)
        if (depart[train  >= date) ret rn train
    ret rn -1

}
```

3.3. *Recherche de tous les trajets possibles avec une correspondance*

```
p   ic static  oid to s ra ets(int[]  depart1  int[]  arri ee1  int[]  depart   int[]  arri ee2) {
      int train1  train   correspondance
      int p  sCo rt1 = -1
      int p  sCo rt  = -1
      for(train1 = 0  train1 < depart1. ength  train1 ++) {
          correspondance = arri ee1[train1  + 1]
          train  = prochain rain(depart   arri ee   correspondance)
          if(train  == -1) ret rn
          he re(depart1[train1])
          De g.print(" --> ")
          he re(arri ee1[train1])
          De g.print("/ ")
          he re(depart [train ])
          De g.print(" --> ")
          he re(arri ee [train ])
          De g.print(" **** e ps tota  ")
          he re(arri ee [train   - depart1[train1])
          De g.print(" **** e ps d'attente ")
          he re(depart [train   - arri ee1[train1])
          De g.print n()
      }
}
```

3.4. *Recherche de trains permettant d'arriver le plus tard possible avant une heure donnée*

```
p   ic static  oid tra etAdapte(int[]  depart1  int[]  arri ee1  int[]  depart   int[]  arri ee   int h) {
      int train1  train   correspondance
      // recherche d  train s r  a seconde partie d  tra et
      for(train  = arri ee . ength - 1  train  >= 0  train --)
          if(arri ee [train   <= h  rea )
      if(train  == -1) {
              De g.print n("  oyage i possi  e (pas de train seconde partie d  tra et) ")
              ret rn }
      // recherche d  train s r  a pre iere partie d  tra et
      // he re d'arri ee d  pre ier train : depart d  second - 1]
      int h  = depart [train  - 1]
      for(train1 = arri ee1. ength - 1  train1 >= 0  train1 --)
          if(arri ee1[train1  <= h  rea )
      if(train1 == -1) {
              De g.print n("  oyage i possi  e (pas de train pre iere partie d  tra et) ")
              ret rn }
      De g.print n("Pre ier train : " + train1)
      De g.print n("Second train : " + train2)
}
```

Exercice 4. *Création d'une pyramide*

```
i port fr.  ssie .script.*
c ass Pyra ide {
    p   ic static  oid espace( { De g.print(" " }
    p   ic static  oid entier(int n { De g.print(n  }
    p   ic static  oid  igne( { De g.print n(  }
    p   ic static  oid  ain(String[  args) {
       int  ig = 1
       int n Espaces = 0
        hi e( ig < 1) {
               for(int i=0  i <n Espaces  i+) espace(
               for(int i=0  i< ig  i+) entier(( ig+i) % 0
               for(int i= ig-1  i>0  i-) entier(( ig+i-) % 0
                igne(
                ig++  n Espaces--
       }
    }
}
```

Exercice 5. *Autour du jeu de la vie*

5.1. *Construction du tableau des voisins d'un point donné*

On travaille modulo la dimension concernée et pour éviter les valeurs négatives on ajoute $n-1$ plutôt que -1.

```
    p   ic static int[ [   oisins( oo ean[ [  t  int i  int ) {
        int[ [    = ne  int[ [
         [0 [0  = (t. ength + i - ) % t. ength   [0 [1  = (t[0 . ength +   - ) % t[0 . ength
         [1 [0  = (t. ength + i - ) % t. ength   [1 [1  =
         [  [0  = (t. ength + i - ) % t. ength   [  [1  = (  + ) % t[0 . ength
         [3 [0  = i   [3 [1  = (t[0 . ength +   - ) % t[0 . ength
         [  [0  = i   [  [1  = (t[0 . ength +   + ) % t[0 . ength
         [ 0  = (i + ) % t. ength   [ 1  = (t[0 . ength +   - ) % t[0 . ength
         [6 [0  = (i + ) % t. ength   [6 [1  =
         [ 1  = (i + ) % t. ength   [ 1  = (  + ) % t[0 . ength
        ret rn
    }
```

5.2. *Calcul de l'état suivant d'une case donnée*

```
    p   ic static  oo ean etatS i ant( oo ean[ [  t  int i  int ) {
        int[ [    =  oisins(t  i  )
        int n = 0  // co pte r d  no  re de  oisins contenant  ne ce    e
        for(int  = 0  < 9  +)
           if(t[ [  [0  [ [  [1 ) n++
        if(n == 3) ret rn tr e
        if(n ==      t[i [ ) ret rn tr e
        ret rn fa se
    }
```

5.3. *Calcul de la génération suivante*

```
    p   ic static  oid generationS i ante( oo ean[ [  t) {
         oo ean[ [  t  = ne   oo ean[t. ength [t[0 . ength
        for(int i = 0  i < t. ength  i+)
           for(int   = 0    < t[0 . ength   +)
               t [i [   = etatS i ant(t  i  )
        for(int i = 0  i < t. ength  i+)
           for(int   = 0    < t[0 . ength   +)
               t[i [   = t [i [
    }
```