

ITI1520 Section 7 Solutionnaire aux exercices

Mémoire de programme

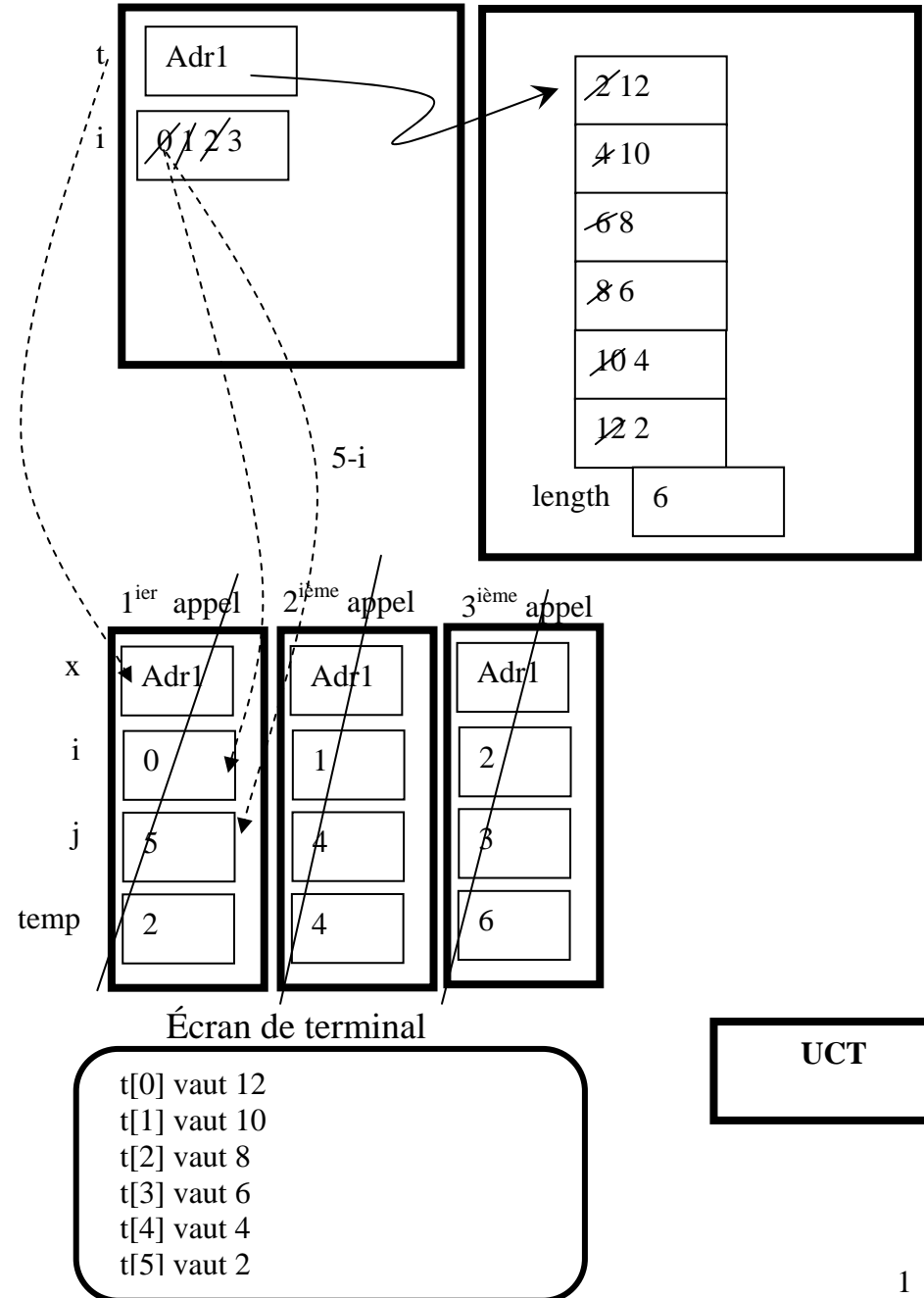
Exercice 7-1 - Échanges de valeurs dans tableau

Mémoire de travail

Mémoire globale

```


class ÉchangeComplet
{
    public static void main (String args[ ])
    { int i = 0;
      int[ ] t;
      t = new int[ ]
          { 2, 4, 6, 8, 10, 12 } ;
      while( i <= 2 )
      {
          échangeTbl(t, i, 5 - i ) ;
          i = i + 1;
      }
      for ( i = 0 ; i <= 5 ; i = i + 1 )
      { System.out.println("t[" + i +
          "]" vaut " + t[i] );
      }
    }
    // échangeTbl : échanges valeurs à i,j
    // Données: x, reference à un tableau,
    //          i,j, 2 indices de x
    public static void échangeTbl(
        int[ ] x,
        int i,int j)
    {
        // DÉCLARE VARIABLES
        int temp ; // Inter: contient x[i]
        // MODULE DE L'ALGORITHMME
        temp = x[i] ;
        x[i] = x[j] ;
        x[j] = temp;
    }
}
    
```



Exercice 7-1: Trace – Table 1, page 1, main

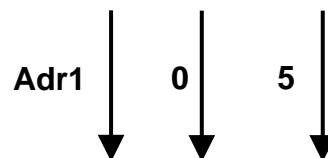
Instructions	i	t	Tableau	Sortie
Valeurs initiales	?	?	?	
1. <code>i = 0</code>	0		?	
2. <code>t = new int[] {2,4,6,8,10,12}</code>			{2,4,6,8,10,12}	
3. <code>while ( i &lt;= 2 ) vrai</code>				
4. <code>échangeTbl( t, i, 5-i )</code> voir Table 2			{12,4,6,8,10,2}	
5. <code>i = i + 1</code>	1			
3. <code>while ( i &lt;= 2 ) vrai</code>				
4. <code>échangeTbl( t, i, 5-i )</code> voir Table 3			{12,10,6,8,4,2}	
5. <code>i = i + 1</code>	2			
3. <code>while ( i &lt;= 2 ) vrai</code>				
4. <code>échangeTbl( t, i, 5-i )</code> voir Table 4			{12,10,8,6,4,2}	
5. <code>i = i + 1</code>				
3. <code>while ( i &lt;= 2 ) faux</code>	3			

Exercice 7-1: Trace – Table 1, page 2, main

Instructions	i	t	Tableau	Sortie
(Valeurs les plus récentes de la page 1)	3		{12,10,8,6,4,2}	
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 )</code>	0			
7. <code>System.out.println( "t["+i+"] = "+t[i] )</code>				t[0] = 12
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 ) vrai</code>	1			
7. <code>System.out.println( "t["+i+"] = "+t[i] )</code>				t[1] = 10
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 ) vrai</code>	2			
7. <code>System.out.println( "t["+i+"] = "+t[i] )</code>				t[2] = 8
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 ) vrai</code>	3			
7. <code>System.out.println( "t["+i+"] = "+t[i] )</code>				t[3] = 6
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 ) vrai</code>	4			
7. <code>System.out.println( "t["+i+"] = "+t[i] )</code>				t[4] = 4
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 ) vrai</code>	5			
7. <code>System.out.println( "t["+i+"] = "+t[i] )</code>				t[5] = 2
6. <code>for ( i = 0; i &lt;= 5; i = i + 1 ) false</code>	6			

Exercice 7-1: Trace – Table 2, échangeTbl(a, 0, 5)

échangeTbl( t, i, 5-i)

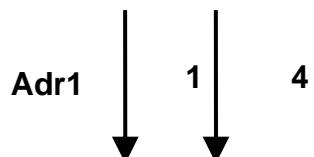


échangeTbl( x, i, j )

Instructions	x	i	j	temp	Tableau de Table 1
Valeurs initiales	2	0	5		{2,4,6,8,10,12}
1. temp = x[i]				2	
2. x[i] = x[j]					{12,4,6,8,10,12}
3. x[j] = temp					{12,4,6,8,10,2}

Exercice 7-1: Trace – Table 3, échangeTbl(a, 1, 4)

échangeTbl( t, i, 5-i)

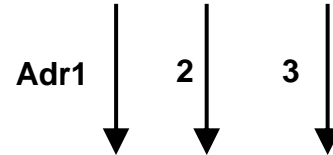


échangeTbl( x, i, j )

Instructions	x	i	j	temp	Tableau de Table 1
Valeurs initiales	12	1	4		{12,4,6,8,10,2}
1. temp = x[i]				4	
2. x[i] = x[j]					{12,10,6,8,10,2}
3. x[j] = temp					{12,10,6,8,4,2}

Exercice 7-1: Trace – Table 4, échangeTbl(a, 2, 3)

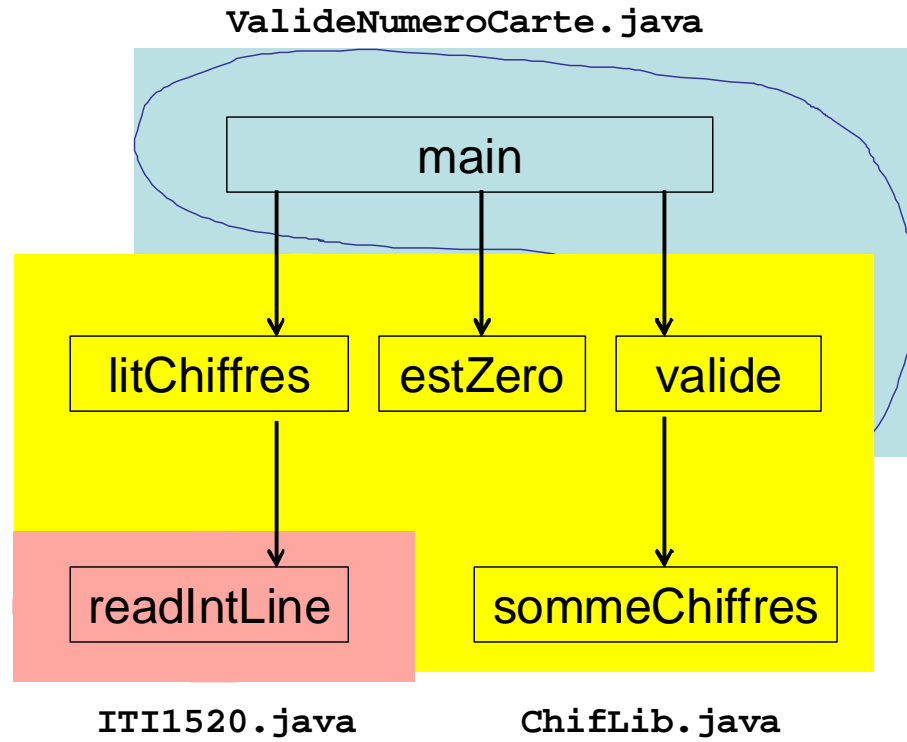
échangeTbl( t, i, 5-i)



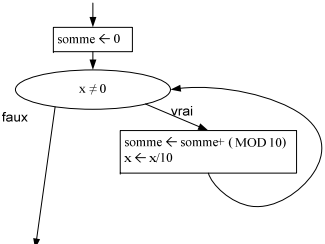
échangeTbl( x, i, j )

Instructions	x	i	j	temp	Tableau de Table 1
Valeurs initiales		2	3		{12,10,6,8,4,2}
1. temp = x[i]				6	
2. x[i] = x[j]					{12,10,8,8,4,2}
3. x[j] = temp					{12,10,8,6,4,2}

Exercice 7-2 : Validation de numéros



Modèle algorithmique	Java
<b>Classe ValideNuméroCarte (ValideNuméroCarte.java)</b>	
<p>DONNÉES: (aucune)  RÉSULTAT: (aucune)  INTERMÉDIAIRES:  chiffres (référence à un tableau d'entiers)  n (nombre d'éléments dans le tableau)  testValide (Indique si la carte est valide)</p> <p>EN-TÊTE: main  MODULE:</p> <pre> graph TD     Start(( )) --&gt; LitChiffres[chiffres,n ← litChiffres()]     LitChiffres --&gt; Decision1{"(n=4) ET (NON estZero(chiffres))?"}     Decision1 -- faux --&gt; AfficheInvalide1[AfficheLigne("Ce numéro est invalide")]     AfficheInvalide1 --&gt; LitChiffres     Decision1 -- vrai --&gt; TestValide[<b>testValide</b> ← valide(chiffres)]     TestValide --&gt; Decision2{testValide?}     Decision2 -- faux --&gt; AfficheInvalide2[AfficheLigne("Ce numéro est invalide")]     AfficheInvalide2 --&gt; LitChiffres     Decision2 -- vrai --&gt; AfficheValide[AfficheLigne("Ce numéro est valide")]     AfficheValide --&gt; LitChiffres   </pre>	<pre> <i>/* La méthode main, tel un chef d'orchestre, invoque les autres méthodes afin de compléter les tâches individuelles. */</i>  public static void main (String [ ] args) {     // invoque litChiffres( ) pour saisir la donnée     int [ ] chiffres = ChifLib.litChiffres( );     while ((chiffres.length == 4) &amp;&amp; (!ChifLib.estZero(chiffres)))     {         // envoie ce nombre à la méthode valide         boolean testValide = valide(chiffres);         // affiche le résultat         if (testValide)             { System.out.println("Ce numéro est valide."); }         else             { System.out.println("Ce numéro est invalide."); }         <b>chiffres = ChifLib.litChiffres( );</b>     } }   </pre>
<p>DONNÉES: chiffres (référence à un tableau de chiffres)  n (nombre d'éléments dans le tableau)  RÉSULTAT: estValide (VRAI si carte est valide, sinon FAUX)  INTERMÉDIAIRES:  troisPremiers (trois premiers chiffres du dernier groupe)  dernierChiffre (tout dernier chiffre du numéro)  somme (somme des 15 chiffres)</p> <p>EN-TÊTE: estValide ← valide(chiffres, n)  MODULE:</p> <pre> troisPremiers ← chiffres[3] / 10 dernierChiffre ← chiffres[3] % 10 somme ← sommeChiffres(chiffres[0]) + sommeChiffres([1])         + sommeChiffres(chiffres[2]) + sommeChiffres(troisPremiers) estValide ← (somme % 10) = dernierChiffre   </pre>	<pre> <i>/* Cette méthode invoque sommeChiffres( ) pour trouver la somme des chiffres d'un entier. Elle compare ensuite les derniers chiffres de la somme des 15 premiers chiffres et du numéro lui-même pour déterminer la validité de la carte. NOTE: Peut être privée!!! */</i> private static boolean valide(int [ ] chiffres) {     // trouve les 3 premiers chiffres du dernier groupe     int troisPremiers = <b>chiffres[3] / 10;</b>     // trouve le tout dernier chiffre du numéro     int dernierChiffre = <b>chiffres[3] % 10;</b>     // trouve la somme des 15 premiers chiffres     int somme = <b>ChifLib.sommeChiffres (chiffres[0])</b>         + <b>ChifLib.sommeChiffres (chiffres[1])</b>         + <b>ChifLib.sommeChiffres (chiffres[2])</b>         + <b>ChifLib.sommeChiffres (troisPremiers);</b>     // détermine la validité     boolean valide = (<b>somme % 10 == dernierChiffre</b>);     return valide; }   </pre>

Modèle algorithmique	Java
<b>Classe ChifLib (ChifLib.java)</b>	
<p>DONNÉES: chiffres (référence à un tableau d'entiers)  n (nombre d'éléments dans le tableau)  RÉSULTAT: drapeau (VRAI si le premier chiffre est zéro, sinon FAUX)  INTERMÉDIAIRES: (aucune)  EN-TÊTE: drapeau <math>\leftarrow</math> estZero(chiffres, n)  MODULE:  drapeau <math>\leftarrow</math> (chiffres[0] = 0)</p> <p>2<sup>ème</sup> version:  drapeau <math>\leftarrow</math> (chiffres[0] = 0) ET (chiffres[1] = 0) ET  (chiffres[2] = 0) ET (chiffres[3] = 0)</p>	<pre>//version1: seulement les 4 premiers chiffres doivent être 0 public static boolean estZero(int [ ] chiffres) {     boolean drapeau;     drapeau = chiffres[0] == 0;     return drapeau; }  // version2: tous les 16 chiffres doivent être à 0 public static boolean estZero(int [ ] chiffres) {     boolean drapeau;     drapeau = ( (chiffres[0] == 0) &amp;&amp;                 (chiffres[1] == 0) &amp;&amp;                 (chiffres[2] == 0) &amp;&amp;                 (chiffres[3] == 0) );     return drapeau; }</pre>
<p>DONNÉES: (aucune)  ITERMEDIATE:  RÉSULTAT: tableauEntiers (référence à un tableau d'entiers)  n (nombre d'éléments dans le tableau)  EN-TÊTE: tableauEntiers, n <math>\leftarrow</math> readDigits()  MODULE:  AfficheLigne("Entrez le numéro de carte à l'aide de quatre ")  AfficheLigne("nombres de quatre chiffres, séparés d'espaces")  AfficheLigne("blancs; ou appuyez sur 0 pour terminer.")  tableauEntiers,n <math>\leftarrow</math> LireLigneEntiers()</p>	<pre>/* Cette méthode demande à l'utilisateur d'entrer son numéro de carte de crédit à l'aide de 4 nombres, qui seront placés dans un tableau. Cette méthode fait appel à readIntLine( ) de la classe ITI1520 afin de lire le tableau d'entiers. */ public static int [ ] litChiffres( ) {     int [ ] tableauEntiers;     System.out.println     ("Entrez le numéro de carte à l'aide de quatre ");     System.out.println     ("nombres de quatre chiffres, séparés d'espaces;");     System.out.println     ("blancs, ou appuyez sur 0 pour terminer.");     tableauEntiers = ITI1520.readIntLine( );     return tableauEntiers; }</pre>
<p>DONNÉES: x (nombre entier)  RÉSULTAT: somme (somme des chiffres du nombre entier)  INTERMÉDIAIRES: (aucune)  EN-TÊTE: somme <math>\leftarrow</math> sommeChiffres(x)  MODULE:</p>  <pre>graph TD     Start([somme ← 0]) --&gt; Decision{x ≠ 0}     Decision -- vrai --&gt; Process[somme ← somme + (MOD 10) x ← x/10]     Process --&gt; Decision     Decision -- faux --&gt; Exit[ ]</pre>	<pre>// Retourne la somme des chiffres d'un nombre x public static int sommeChiffres(int x) {     int somme = 0;     while (x != 0)     {         somme = somme + x % 10;         x = x / 10;     }     return somme; }</pre>



