****

**ﺭﺎﻧﻣﻟﺍ ﺲﻧﻭﺗ ﺔﻌﻣﺎﺠ**

**Université de Tunis El Manar**



Faculté de Sciences de Tunis  
Département de sciences Informatique

Mini-Projet JAVA   
«  Algorithme: Priorité avec Partitionnement fixe »

**Mini\_Projet Présenté par**

**Ben Mhamed Aymen AMPHI B\_G3**

**Rekaya Ayda AMPHI A\_G3**

**Année universitaire  
2011/2012**

### 1) Objectif :

- Connaître le principe de gestion de mémoire en Multiprogrammation avec partitions fixes :

•Partition fixes de tailles égales :

la taille de partition est supérieure à l’espace recueilli par un certain processus le reste de cette partition restera vide ce qui cause une perte d’espace mémoire.

Le sous système chargé de la gestion de mémoire met en place une structure des données appelée table des partitions ayant pour rôle l’indication de l’état (libre ou occupée) de chaque partition en identifiant chacune soit par son adresse soit par son numéro.

•Partition fixes de tailles inégales  :

Dans ce cas la mémoire est subdivisée en partitions de tailles inégales. Chaque nouveau processus est placé dans la file d’attente de la plus petite partition qui peut le contenir. Cette façon de faire peut conduire à faire attendre un processus dans une file, alors qu’une autre partition pouvant le contenir est libre.

-Implémenter l’algorithme «  Priorité avec Partitionnement fixe » d'allocation de processus.

### 2)Introduction à la gestion de la MC :

La mémoire est une ressource rare. Il n'en existe qu'un seul exemplaire et tous les processus actifs doivent la partager Selon l’un des mécanismes de gestion de ce partage suivant :

Gestion sans recouvrement ni pagination :

• La monoprogrammation : Le modèle de base

• La multiprogrammation : Multiprogrammation avec partitions fixes

Partition fixes de tailles égales

Partition fixes de tailles inégales

Multiprogrammation avec partitions variables

Gestion avec recouvrement sans pagination :

• Le va-vient (Swap In/Out)

• Opérations sur la mémoire (Compactage Logique /Physique)

• Gestion de la mémoire par table de bits (First-fit/Best-fit /Worst- fit)

* Gestion avec recouvrement, avec pagination ou segmentatio : Notion de MV

La pagination

### 3) Package

**3.1)Introduction**

L’importation de package en Java se fait via l’instruction *import* en en-tête de fichier. Le séparateur ‘*.’* est utilisé pour séparer les noms de packages et de sous-packages. Le suffixe ‘*\*’* définit que toutes les classes du package doivent être importées. Le suffixe ‘*\*’* peut également être substitué par un nom de classe, afin de filtrer individuellement les classes importées du package. L’importation de toutes les classes d’un package n’augmente ni la taille du byte code Java, ni le temps d’exécution du programme. Le filtrage des classes lors de l’importation d’un package n’a d’utilité que de limiter les ambiguïtés en cas de classes de noms similaires entre différents packages importés. Dans ce cas, on utilise le nom du package en plus du nom de la classe pour lever ces ambiguïtés. L’exemple suivant résume ces différents principes ,pour l’utilisation de deux classes *Vector* en parallèle :

import java.lang.\*;

import java.util.Vector;

class Vector {}

class Stack {

private java.util.Vector v1;

private Vector v2;

String stackName;

}

Dans cet exemple, l’importation complète (*\**) du package *java.lang* est effectuée. Cette directive est en fait facultative (nous l’utilisons ici à titre d’exemple), en effet ce package est obligatoirement importé par défaut dans les programmes Java. Implémenter le code suivant afin de vérifier cette importation par défaut. Vous pouvez vérifier que la classe *String* est définie dans le package *java.lang*, vous l’utilisez cependant ici sans avoir recours à l’instruction *import java.lang.String*.

class Default {

String s1;

}

**3.2)Définition de package**

L’API specification SDK contient plusieurs packages standards. Il est cependant possible pour un programmeur Java de définir ses propres packages à l’aide du mot clé *package*. Créer un répertoire (package), dans ce répertoire (package) créer un sous-répertoire (myp). Dans ce sous-répertoire (myp), créer un fichier (MyClass.java). Implémentez le code suivant dans ce fichier, et compilez-le :

package myp;

public class MyClass {

public void print()

{System.out.println("first package");}

}

Retourner dans le répertoire (package), implémenter le code suivant dans un fichier (App.java) et le mettre en œuvre :

import myp.\*;

public class App {

public static void main(String args[])

{

MyClass my = new MyClass();

my.print();

}

}

Dans cet exemple, vous avez défini votre premier package *myp* constitué d’une classe *MyClass*. Vous avez importé et mis en oeuvre ce package dans votre programme principal à l’aide de l’instruction *import myp.\**.

Retourner dans votre répertoire (myp) et créer un sous-répertoire (mypp). Dans ce sous-répertoire (mypp), créer un fichier (MyPClass.java) et implémenter le code suivant, compiler :

package myp.mypp;

public class MyPClass {

public void print()

{System.out.println("first sub-package");}

}

Retourner dans le répertoire (package), redéfinir le fichier (App.java) de la façon suivante et le mettre en œuvre :

import myp.\*;

import myp.mypp.\*;

public class App {

public static void main(String args[])

{

MyClass my = new MyClass(); my.print();

MyPClass myp = new MyPClass(); myp.print();

}

}

Dans cet exemple, vous avez défini votre premier sous-package *mypp* du package *myp* constitué d’une classe *MyPClass*. Vous avez importé et mis en oeuvre ce package dans votre programme principal à l’aide de l’instruction *import myp.mypp.\**.

**3.3)Package local**

Cependant, sans le savoir vous avez déjà défini des packages à plusieurs reprises dans vos programmes Java. En effet, lorsque vous compilez différentes classes stockées en (.class) dans un même répertoire, vous constituez un package local (sans nom). Implémenter l’exemple suivant et le mettre en oeuvre.

class Random {}

public class App {

public static void main(String args[])

{System.out.println(new Random());}

}

Placer ensuite la définition de la classe *Random* en commentaire, sans supprimer le fichier (Random.class) du répertoire local, recompiler l’ensemble, commenter. Supprimer ensuite le fichier (Random.class) du répertoire local et recompiler l’ensemble.

Vous voyez dans cet exemple qu’il n’est pas nécessaire d’avoir les sources correspondantes aux définitions des classes pour les exploiter. Durant la compilation d’un fichier source, les (.class) des classes utilisées dans ce fichier source sont recherchées dans le répertoire local par le compilateur. De cette manière le répertoire local constitue un package local. Reprendre le fichier de la manière suivante, recompiler l’ensemble. Que pouvez vous conclure sur les priorités de recherche à la compilation entre le package local et les packages externes.

import java.util.\*;

class Random {}

public class App {

public static void main(String args[])

{

System.out.println(new Random());

System.out.println(new java.util.Random());

}

}

**3.4)Droit d’accès**

Un package constitue un niveau d’encapsulation supplémentaire à celui de la classe. En Java, il existe quatre déclarations de droit d’accès définissables dans une classe pour les données membres et les méthodes : *private*, pas de déclaration, *protected*, et *public*. Ces déclarations sont utilisées pour définir les accès aux données membres et aux méthodes entre classes, au sein d’un même package et entre différents packages. Les déclarations *private* et pas de déclaration sont utilisées « majoritairement » pour les relations d’héritage. Les déclarations *protected* et *public* sont utilisées majoritairement pour les droits d’accès aux classes des packages.

On se propose d’illustrer la déclaration *public* dans un premier temps, reprendre le fichier (MyClass.java) du package *myp* de la façon suivante :

package myp;

public class MyClass {

public void print()

{System.out.println("public class, public method");}

void print2()

{System.out.println("public class, no public method");}

public void print3()

{print2();new MyClass2().print();}

}

class MyClass2 {

void print()

{System.out.println("no public class, no public method");}

}

Dans le fichier (App.java) le mettre en oeuvre de la façon suivante. Tenter d’exécuter les lignes en commentaire, et commenter l’utilisation du mot clé *public*. D’après vous, quel sens aurait la déclaration *public* d’une méthode dans la classe *MyClass2* ?

MyClass my = new MyClass();

my.print();

//MyClass2 my2 = new MyClass2();

//my.print2();

my.print3();

Afin d’illustrer l’utilisation de la déclaration *protected*, rajouter à la définition de la classe *MyClass* la donnée membre suivante :

protected int v;

Dans le répertoire (package), définir une classe *MyClassE* de la façon suivante :

class MyClassE extends MyClass {

MyClassE()

{System.out.println(v);}

}

Mettre en œuvre les classes *MyClassE* et *MyClass* de la façon suivante, quelle erreur de compilation relevez-vous sur l’accès à la donnée membre *v* de la classe *MyClass* :

new MyClassE();

MyClass my = new MyClass();

//System.out.println(my.v);

Dans cette partie du TP vous avez illustré l’utilisation des droits d’accès *protected* et *public* dans le cadre de la définition de packages. Dans les TP précédents, vous avez illustré les déclarations *private* et sans déclaration dans le cadre des relations d’héritage. Le tableau suivant résume l’utilisation des droits d’accès en Java.

|  |  |
| --- | --- |
| **Déclarations** | **Accessibilité** |
| private | la classe |
| sans | la classe et les classes du même package |
| protected | la classe, les classes du même package, et les classes dérivées |
| public | partout |

### 4) Multi-Threading :

**4.1) Définition :**

Disposer de plusieurs threads dans même processus. Un Thread ne peut s’executer que via un processus.

**4.2) Multi-Threading sous Java :**

**Methode de creation:**

**\***Par heritage de la classe Thread et par surcharge de la méthode **run()**

Class MonThread extends Thread{

MonThread() {

…code du constructeur…

}

Public void run(){

…code a executer dans le thread…

}

}

\*Declarer un classe qui imlimente l’interface **Runnable**

class MonThread2 imliment Runnable {

MonThread2() {

…code du constructeur…

}

public void run() {

…code a éxecuter dand le thread…

}

}

public static void main(String[] args){

MonThread2 p=new MonThread2();

Thread t =new thread (p);

t.start();

}

### 5) Outils SDK

**5.1)Introduction**

La plate forme SDK est basée sur l’utilisation de différents outils standards. Plusieurs outils existent, parmi eux javac, java, jar, et javadoc sont couramment utilisés. Vous pouvez accéder à la documentation des outils SDK depuis la page principale J2SDK Documentation (C:\j2sdk1.4.1\_02\docs\index.html), en ouvrant le lien Tool Documentation. Dans cette partie nous introduisons l’utilisation de ces outils dans un cadre général, et plus particulièrement pour l’exploitation de package.

**5.2)javac et java**

Sans le savoir, vous avez utilisé à plusieurs reprises ces outils standards SDK lors de la mise en oeuvre de vos différents programmes Java. En effet, à chaque compilation/exécution de vos programmes (dans votre environnement de programmation) vous avez fait appel respectivement à javac et java.

### 6) Code Source :

class Processus

package testJava;

public class Processus {

private String nom;

private int Duree;

private int taille ;

private int TR = 0;

private int TA=0;

private int Date\_deb;

private int priorité;

public Processus()

{

}

public Processus (String x1,int D,int prio,int tail)

{

nom=x1;

Duree =D;

priorité = prio ;

taille = tail ;

}

public String get\_Nom()

{

return (nom);

}

public void Set\_nom (String x)

{

nom=x;

}

public int get\_Duree()

{

return(Duree);

}

public void set\_Duree(int D1)

{

Duree=D1;

}

public int get\_TR ()

{

return(TR);

}

public void set\_TR (int TR1 )

{

TR=TR1;

}

public int get\_TA ()

{

return(TA);

}

public void set\_TA (int TA1)

{

TA=TA1;

}

public int get\_Date\_deb ()

{

return(Date\_deb);

}

public void set\_Date\_deb(int Date\_deb1)

{

Date\_deb=Date\_deb1;

}

public int get\_priorité()

{

return(priorité);

}

public void set\_get\_priorité(int prioritééé)

{

priorité=prioritééé ;

}

public int get\_taille()

{

return(taille) ;

}

public void set\_get\_taille(int tailleeee)

{

taille = tailleeee ;

}

Public String ToString ()

{return («le nom du processeur :" +nom+"la date de début de processeur est:" +Date\_deb+ "le duré de processus est "+Durée);

}

}

Class Ordonnanceur

package testJava;

public class Ordonnanceur extends Thread{

//tableau des proccessus

public Processus pro[];

//horloge

int horloge=0;

int trr=0,taa=0;

public Ordonnanceur(Processus[] pro) {

super();

this.pro = pro;

}

public void run(){

for(int i=0;i<pro.length;i++){

//parcour des tableau

pro[i].set\_Date\_deb(horloge);

pro[i].set\_TA(horloge);

taa+=horloge;

horloge=horloge+pro[i].get\_Duree();

pro[i].set\_TR(horloge);

trr+=horloge;

}

}

}

Class Mémoire

package testJava;

import java.io.\*;

import javax.swing.\*;

public class Memoire {

public Processus p[];

public void donnermemoire(Processus[] p)

{

//declaration de variable

JOptionPane jop3 = new JOptionPane();

JOptionPane jop4 = new JOptionPane();

//saisie de donnée

//Processus p[]=new Processus[0];

String b = jop3.showInputDialog(null, "DONNER LA TAILLE DE LA MEMOIRE EN ko", "tp", JOptionPane.QUESTION\_MESSAGE);

int taille = Integer.parseInt(b);

String c = jop3.showInputDialog(null, "DONNER LE NOMBRE DE PARTITIONNEMENT", "tp", JOptionPane.QUESTION\_MESSAGE);

int nbp = Integer.parseInt(c);

jop4.showMessageDialog(null,"LE NOMBRE DE PROCESSUS:" + p.length +"\n" + "LA TAILLE DE LA MEMOIRE EN ko:" + taille +"\n" + "DONNER LE NOMBRE DE PARTITIONNEMENT:"+ nbp ,"LECTURE DE DONNER", JOptionPane.INFORMATION\_MESSAGE);

int tp[][]=new int[100][100];

for (int i=1;i<=nbp;i++)

{

String tailpart = jop3.showInputDialog(null, "DONNER LA TAILLE DE LA PARTITION Numero " + i, "tp", JOptionPane.QUESTION\_MESSAGE);

int X = Integer.parseInt(tailpart);

tp[i][1] = X ; //0 si l'espace est libre sinon contient le numero de processus

tp[i][2] = 0 ;

}

/\*Une autre vision de voir le chose est de conciderer une matrice dans le nombre de ligne est le nombre de

\* processus et le nombre de colonne est 6 :

\* 1ere col : Nom de processus

\* 2eme col : taille de processus

\* 3eme col : temps d'arrivée

\* 4eme col : temps d'execution

\* 5eme col :temps estimer pour la fin de processus

\* 6eme col : priorité

\*

int info[][]=new int[100][100];

for (int i=1;i<=p.length;i++)

{

info[i][1] = i ;

//jop4.showMessageDialog(null,"Processus Numero " + i, JOptionPane.INFORMATION\_MESSAGE);

String tail1 = jop3.showInputDialog(null, "TAILLE EN ko" , "Processus" + i, JOptionPane.QUESTION\_MESSAGE);

int XX = Integer.parseInt(tail1);

info[i][2] = XX ;

String tail2 = jop3.showInputDialog(null, "DATE D ARRIVER : " , "Processus" + i, JOptionPane.QUESTION\_MESSAGE);

int XXX = Integer.parseInt(tail2);

info[i][3] = XXX;

String tail3 = jop3.showInputDialog(null, "LE TEMPS D EXCUTION : ", "Processus" + i, JOptionPane.QUESTION\_MESSAGE);

int XXXX = Integer.parseInt(tail3);

info[i][4] = XXXX ;

info[i][5] = info[i][3]+info[i][4] ;

String tail4 = jop3.showInputDialog(null, "PRIORITEE " , "Processus" + i, JOptionPane.QUESTION\_MESSAGE);

int XXXXX = Integer.parseInt(tail4);

info[i][6] = XXXXX ;

jop4.showMessageDialog(null,"TAILLE DE PROCESSUS:" + XX +"\n" + "DATE D'ARRIVEE:" + XXX +"\n" + "TEMPS D'EXCUTION :"+ XXXX + "\n" + "TEMPS DE FIN D'EXCUTION :"+ info[i][5] + "\n" + "PRIORITEE :" + XXXXX ,"Information sur le Processus" + i, JOptionPane.INFORMATION\_MESSAGE);

}

//s:Temps total de Connexion

int s=0;

for (int i=1;i<=p.length;i++)

{

s+=info[i][4];

}

\*/

int s=0 ;

for (int ii=0 ; ii <= p.length ;ii++)

{

s= s+ p[ii].get\_Duree() ;

}

//System.out.println(s);

int L=1;

int k=1;

int memoireattend[]=new int[100];

int tabproc[]=new int[100];

for (int t=0;t<s;t++)

{

for (int i=1;i<= p.length;i++)

{

if ( p[i].get\_Date\_deb() == t)

{

for (int j=1;j<=nbp;j++)

{

if (tp[j][1]<p[i].get\_taille())

{

jop4.showMessageDialog(null,i+ "Ne peut pas étre allouer dans la partition " + j + "de la memoire à cause de la taille" ,"traitement", JOptionPane.INFORMATION\_MESSAGE);

//System.out.println( i+ "Ne peut pas étre executé");// i sera differer

}

}

boolean test=true ;

for (int j=1;j<=nbp && test ;j++)

{

if((tp[j][1]>=p[i].get\_taille()) && (tp[j][2]==0))

{

tp[j][2]=i;

test= false ;

}

}

}

}

int lib ;

for (int i1=1;i1<=p.length;i1++)

{

if ( p[i1].get\_Date\_deb() == t)

{

for (int j1=1;j1<=nbp;j1++)

{

lib = tp[j1][2] ;

if (lib!=0)

{

if ((p[i1].get\_taille()<=tp[j1][1]) && (p[lib].get\_priorité() < p[i1].get\_priorité()))

{

memoireattend[L]=tp[j1][2];

tp[j1][2]=i1 ;

L++ ;

}

}

}

}

for (int i2=1;i2<=p.length;i2++)

{

for (int compt=1;compt<=nbp;compt++)

{

if ((tp[compt][2]==i2))

{

if (t==p[i2].get\_TR())

{

System.out.println( i2+ " va étre allouer par le processeur ");

tabproc[k]=i2 ;

k++;

}

}

}

}

for (int c0=1;c0<=p.length;c0++)

{

for (int c1=1;c1<=nbp;c1++)

{

if ((tp[c1][2]==0) && (tp[c1][1]>=p[c0].get\_taille()))

{

tp[c1][1]=memoireattend[c0];

//L++;

}

}

}

}

}

jop4.showMessageDialog(null,"Le Processus par ordre d'excution dans le Processeur :","traitement", JOptionPane.INFORMATION\_MESSAGE);

//System.out.println("Le Processus par ordre d'excution dans le Processeur :");

for (int c2=1;c2<=p.length;c2++)

{

//System.out.println(tabproc[c2]);

jop4.showMessageDialog(null,tabproc[c2],"traitement", JOptionPane.INFORMATION\_MESSAGE);

}

}

}

Class Priorité

package testJava;

public class Priorité

{

public Priorité (Processus[] pro)

{

//super()

int Horloge=0;

//this.pro = pro;

for(int i=0;i<pro.length;i++){

for(int j=1;j<pro.length;j++)

{

if(pro[i].get\_priorité() < pro[j].get\_priorité()){

System.out.println(i+ "est plus prioritaire que" + pro);

}

int a=pro[i].get\_Duree();

{

Horloge=Horloge+a;

int TR=Horloge+a;

int TA=TR-a;

pro[i].set\_TA(TA);

pro[i].set\_TR(TR);

System.out.println(pro[i].toString());}}}

}

Class Interface

package testJava;

import java.awt.HeadlessException;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class Interface extends JFrame implements ActionListener {

JPanel p1;

Processus p[]=new Processus[0];

JTable t1=new JTable();

JTable t2=new JTable();

JButton ajout=new JButton("Ajouter");

JTextField nom=new JTextField();

JTextField dure=new JTextField();

JTextField priorité=new JTextField();

JTextField taille=new JTextField();

JLabel ne=new JLabel("Nouveau Processus: ");

JLabel n=new JLabel("Nom: ");

JLabel d=new JLabel("Durée:");

JLabel k=new JLabel("Priorité: ");

JLabel l=new JLabel("Taille:");

JButton fifo=new JButton("Gestion file d'attente Memoire FIFO");

JLabel ta=new JLabel("Temps Moyenne d'attente:");

JLabel tr=new JLabel("Temps Moyenne de Repponse:");

DefaultTableModel model1;

DefaultTableModel model2;

public Interface() throws HeadlessException {

super();

p1=(JPanel)this.getContentPane();

p1.setLayout(null);

this.setSize(700,800);

this.setTitle("Projet");

this.setLocationRelativeTo(null);

// \*

model1=(DefaultTableModel)t1.getModel();

model1.setRowCount(0);

model1.setColumnCount(4);

Object[] row=new Object[4];

row[0]="Nom";

row[1]="Durée";

row[2]="Priorité";

row[3]="taille";

model1.addRow(row);

t1.setModel(model1);

// mise en place

p1.add(ne);

ne.setBounds(10, 10, 500, 30);

p1.add(n);

n.setBounds(10,40, 100, 30);

p1.add(d);

d.setBounds(10, 100, 100, 30);

p1.add(k);

k.setBounds(290,40, 100, 30);

p1.add(l);

l.setBounds(290, 100, 100, 30);

p1.add(nom);

nom.setBounds(120, 40, 100, 30);

p1.add(dure);

dure.setBounds(120, 100, 100, 30);

p1.add(priorité);

priorité.setBounds(350, 40, 100, 30);

p1.add(taille);

taille.setBounds(350, 100, 100, 30);

p1.add(ajout);

ajout.setBounds(10, 160, 100, 30);

p1.add(t1);

t1.setBounds(10, 200, 650, 200);

p1.add(fifo);

fifo.setBounds(360, 400, 289, 30);

p1.add(t2);

t2.setBounds(10, 450, 650, 200);

p1.add(ta);

ta.setBounds(10, 670, 650, 30);

p1.add(tr);

tr.setBounds(10, 720, 650, 30);

ajout.addActionListener(this);

fifo.addActionListener(this);

//\*\*\*

model2=(DefaultTableModel)t2.getModel();

model2.setRowCount(0);

model2.setColumnCount(5);

Object[] row2=new Object[5];

row2[0]="Nom";

row2[1]="Durée";

row2[2]="Date debut";

row2[3]="TR";

row2[4]="TA";

model2.addRow(row2);

t2.setModel(model2);

}

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

if(arg0.getSource()==ajout){

Object[] row=new Object[4];

row[0]=nom.getText();

row[1]=dure.getText();

row[2]=priorité.getText();

row[3]=taille.getText();

model1.addRow(row);

t1.setModel(model1);

Processus pp[]=new Processus[p.length+1];

for(int i=0;i<p.length;i++){

pp[i]=p[i];

}

pp[p.length]=new Processus(row[0].toString(),Integer.parseInt(row[1].toString()),Integer.parseInt(row[2].toString()),Integer.parseInt(row[3].toString()));

p=pp;

}

if(arg0.getSource()==fifo){

Ordonnanceur fi=new Ordonnanceur(p);

fi.run();

model2=(DefaultTableModel)t2.getModel();

model2.setRowCount(0);

model2.setColumnCount(5);

Object[] row2=new Object[5];

row2[0]="Nom";

row2[1]="Durée";

row2[2]="Date debut";

row2[3]="TR";

row2[4]="TA";

model2.addRow(row2);

for(int i=0;i<p.length;i++){

row2=new Object[5];

row2[0]=p[i].get\_Nom();

row2[1]=p[i].get\_Duree();

row2[2]=p[i].get\_Date\_deb();

row2[3]=p[i].get\_TR();

row2[4]=p[i].get\_TA();

model2.addRow(row2);

t2.setModel(model2);

}

ta.setText("Temps Moyenne d'attente: "+fi.taa/p.length);

tr.setText("Temps Moyenne de Repponse:"+fi.trr/p.length);

}

}

public static void main(String[] args) {

Interface i=new Interface();

i.setVisible(true);

}

}

Class Main

package testJava;

public class Main {

public static void main(String[] args) {

Processus p[]=new Processus[4];

p[0]=new Processus("Pross0",3,0,200);

p[1]=new Processus("Pross1",9,2,150);

p[2]=new Processus("Pross2",31,1,100);

p[3]=new Processus("Pross3",5,3,300);

Ordonnanceur fifo = new Ordonnanceur(p);

fifo.run();

Memoire m = new Memoire();

m.donnermemoire(p);

}

}

### Conclusion :

Le langage java est un langage évolué, classé 1er selon le classement TIOBE de langages de programmation pour deux années successives (2011& 2012), le plus demandable dans les divers domaines, les logiciels réalisés par Cobol et VB…

### Sommaire :

### 

Objectif ………………………………………………………………………………………………… Page2

Introduction à la gestion de laMC………………………….…..………………………… Page 2

Introduction au package………………………………………………………………………… Page 3

Définition de package……………………………………………………………………………. Page 4

Package local………………………………………………………………………………………… Page 6

Droits d’accès………………………………………………………………………………………… Page 7

Multi-Threading…………………………………………………………………………………… Page 9

Definition……………………………………………………………………………………………… Page 9

Multi-Threading sous Java…………………………………………………………………… Page 9

Introduction aux outils SDK…………………………………………………………………… Page 10

javac et java………………………………………………………………………………………….. Page 10

Code Source…………………………………………………………………………………………. Page 11

Conclusion……………………………………………………………………………………………. Page 23