Java et Mascopt

Jean-François Lalande, Michel Syska, Yann Verhoeven

Projet Mascotte, I3S-INRIA
Sophia-Antipolis, France

Formation Mascotte 09 janvier 2004

Java

Java - Compiler et Exécuter

```
> ls
Lancement.java Tutoriel.java
> javac Tutoriel.java
> javac Lancement.java
> ls
Lancement.class Lancement.java Tutoriel.class Tutoriel.java
> export CLASSPATH=./:$CLASSPATH
> java Lancement
Nombre de vertex: 0
> cd /net/ailleurs/pas-ici/a-l-autre-bout-du-monde
> java Lancement
Nombre de vertex: 0
```

Notion de Classe

```
class Tutoriel                                                    1
{                                                                 2
                                                                  3
   public int nb_vertex = 0;                                      4
   private int nb_tutoriel = 0;                                   5
                                                                  6
   public void displayNbVertex()                                  7
   {                                                              8
      System.out.println("Nombre de vertex: " + nb_vertex);       9
   }                                                              10
                                                                  11
}                                                                 12
```

```
public class Lancement                                  1
{                                                       2
                                                        3
  public static void main(String args [])              4
  {                                                     5
    Tutoriel tutoriel = new Tutoriel();                 6
                                                        7
    tutoriel.displayNbVertex();                         8
  }                                                     9
                                                        10
}                                                       11
```

Packages

```
package tutoriel.essai;

public class Tutoriel
{
   public int nb_vertex = 0;
   private int nb_tutoriel = 0;

   public void displayNbVertex()
   {
      System.out.println("Nombre de vertex: " + nb_vertex);
   }
}
```

```
ls tutoriel/essai
Tutoriel.java Tutoriel.class
```

```
//import tutoriel.essai.*;
import tutoriel.essai.Tutoriel;

public class Lancement
{

  public static void main(String args [])
  {
    Tutoriel tutoriel = new Tutoriel();

    tutoriel.displayNbVertex();
  }

}
```

Constructeur

```
package tutoriel.essai;

public class Tutoriel
{
   public int nb_vertex = 0;
   private int nb_tutoriel = 0;

   public Tutoriel()
   {
      System.out.println("Je me construits...");
   }
   public void displayNbVertex()
   {
      System.out.println("Nombre de vertex: " + nb_vertex);
   }
}
```

Structures de controle

```
if (condition)
{ }
else
{ }
```

```
while (condition)
{ }
```

```
for(int i=0; i<4; i++)
{
        System.out.println("i= "+i);
}
```

```
switch(k){
        case 5: System.out.println("Case k=5");
        break;
        case 15: System.out.println("Case k=15");
        break;
        default: System.out.println("case default");
}
```

Iterateur sur des ensembles

- Vector, List, ArrayList
- Set, HashSet, HashTable

```
Vector v = new Vector();                                          1
v.add(new Integer(9));                                            2
v.add(new Integer(4));                                            3
                                                                  4
Iterator it = v.iterator();                                       5
while (v.hasNext())                                               6
{                                                                 7
   Integer courant = (Integer)v.next();                           8
   System.out.println("courant=" + courant);                       9
}                                                                 10
```

Mascopt

Récuperer Mascopt

- Mascopt par CVS

```
export CVSROOT=cvs-sop.inria.fr:/CVS/mascotte
export CVS_RSH=ssh
cvs checkout mascoptDev
cd mascoptDev
```

```
export CVS_RSH=ssh
cvs -d :ext:login@cvs-sop.inria.fr:/CVS/mascotte checkout mascoptDev
cd mascoptDev
```

- Architecture
  - ⋆ bin which contains scripts which launch applications
  - ⋆ docs wich contains the documentation
  - ⋆ files which contains some files of graphs or networks
  - ⋆ jar which contains the jar files
  - ⋆ launch which contains the code of applications
  - ⋆ licences all the licences of each part of provided code

- ⋆ src which contains the source code of algorithms developed by users. This is the place where you can put the code of your algorithm.
- ⋆ samples which contains some samples references of classes of mascopt.
- ⋆ tests wich contains bad code of developers, where they test their algorithms

- Ou aller ?

  - ⋆ toutes les algorithmes (classes) sans main() $\longrightarrow$ src/
  - ⋆ tous les tests qui utilisent ces algos $\longrightarrow$ tests/
  - ⋆ quand l'algo marche $\longrightarrow$ écrire un main() dans samples/

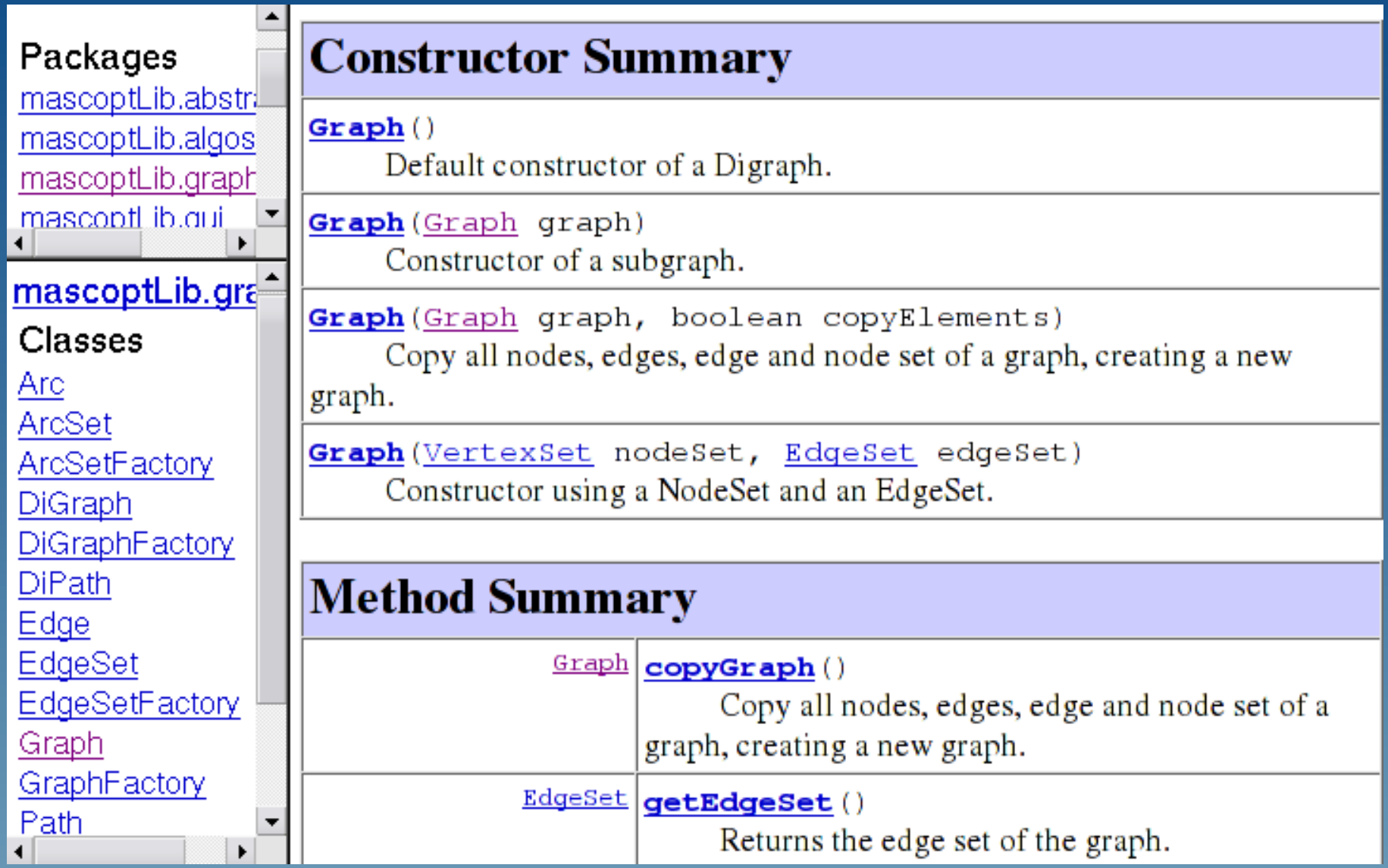
mascoptLib

- Division du code en deux parties
- mascoptLib
  - ⋆ gestion des graphes, GUI
  - ⋆ gestion des entrées sorties
  - ⋆ algorithmes validés et robustes
- mascoptDev
  - ⋆ Espace utilisateurs
  - ⋆ Tests, algorithmes en développement
- dans mascoptDev on trouve mascoptLib.jar

Makefile

- source SETENV
- make
  - ⋆ Compile les classes des utilisateurs (src/...)
  - ⋆ Compile les mains des utilisateurs (tests/...)
  - ⋆ Compile les exemples des utilisateurs (samples/...)
- make javadoc
  - ⋆ Construit la javadoc de mascoptDev et mascoptLib

```
popotte:~/unison/mascopt/mascoptDev> source SETENV
popotte:~/unison/mascopt/mascoptDev> make
make -s all_
log Cplex detected.
popotte:~/unison/mascopt/mascoptDev>ls classes/
...
```


Packages
mascoptLib.abstr
mascoptLib.algos
mascoptLib.graph
mascoptLib.gui
mascoptLib.gra
Classes
Arc
ArcSet
ArcSetFactory
DiGraph
DiGraphFactory
DiPath
Edge
EdgeSet
EdgeSetFactory
Graph
GraphFactory
Path
Constructor Summary
Graph()
Default constructor of a Digraph.
Graph(Graph graph)
Constructor of a subgraph.
Graph(Graph graph, boolean copyElements)
Copy all nodes, edges, edge and node set of a graph, creating a new graph.
Graph(VertexSet nodeSet, EdgeSet edgeSet)
Constructor using a NodeSet and an EdgeSet.
Method Summary
Graph
copyGraph()
Copy all nodes, edges, edge and node set of a graph, creating a new graph.
EdgeSet
getEdgeSet()
Returns the edge set of the graph.

Créér un graphe

```
Vertex n0 = new Vertex();                                          1
Vertex n1 = new Vertex();                                          2
Vertex n2 = new Vertex();                                          3
Vertex n3 = new Vertex();                                          4
Vertex n4 = new Vertex();                                          5
Vertex n5 = new Vertex();                                          6
                                                                   7
// creation des arcs entre les neouds                              8
Arc a0 = new Arc(n0, n2);                                          9
Arc a1 = new Arc(n1, n2);                                          10
Arc a2 = new Arc(n2, n3);                                          11
Arc a3 = new Arc(n3, n4);                                          12
Arc a4 = new Arc(n3, n5);                                          13
                                                                   14
// creation d'un ensemble de noeuds V                              15
VertexSet V = new VertexSet();                                     16
                                                                   17
// on ajoue les noeuds ds l'ensemble de noeuds                     18
V.add(n0);                                                         19
```

```
V.add(n1);                                                          20
V.add(n2);                                                          21
V.add(n3);                                                          22
V.add(n4);                                                          23
V.add(n5);                                                          24
                                                                    25
// creation d'un ensemble d'arc E                                   26
ArcSet E = new ArcSet(V);                                           27
E.add(a0);                                                          28
E.add(a1);                                                          29
E.add(a2);                                                          30
E.add(a3);                                                          31
E.add(a4);                                                          32
                                                                    33
// cration d'un Digraph graph=V,E                                   34
DiGraph graph = new DiGraph(V, E);                                  35
                                                                    36
// on l'affiche sur la sortie standart                              37
System.out.println("mon graph = " + graph);                         38
```

- Storing/Getting values
  - ⋆ For storing a String on vertix, arc, node set, edge set, graph:

```
getValue(String name)                                  1
setValue(String name, String value)                    2
```

  - ⋆ For storing an Integer on nodes, arcs and graphs:

```
getIntegerValue(String name)                           1
setIntegerValue(String name, Integer value)            2
```

  - ⋆ For storing an int on nodes, arcs and graphs:

```
getIntValue(String name)                               1
setIntValue(String name, int value)                    2
```

  - ⋆ For storing a Double on nodes, arcs and graphs:

```
getDoubleValue(String name)                            1
setDoubleValue(String name, Double value)              2
```

  - ⋆ For storing a double on nodes, arcs and graphs:

```
getDouValue(String name)                               1
setDouValue(String name, double value)                 2
```

- Exemple:

```
v1.setValue("poids","12");                                  1
v1.getValue("poids");                                       2
v1.setIntegerValue("poids", new Integer(9));                3
v1.setDouValue("poids", 18.34);                             4
```

- Conversions:

```
String s_a = "12";                                          1
String s_b = "13.213";                                      2
String s_c = "32.32";                                       3
String s_d = "14";                                          4
Integer a = new Integer(Integer.parseInt(s_a));             5
double b = Double.parseFloat(s_b);                          6
Double c = new Double(Double.parseFloat(s_c));              7
int d = Integer.parseInt(s_d);                              8
```

Ensemble et sous-ensemble

- Remplir un set

```
NodeSet V0 = new NodeSet();                                          1
                                                                     2
for (int i=0; i<10;i++)                                              3
{                                                                    4
   V0.add(new Node(Math.random(),Math.random()));                    5
}                                                                    6
System.out.println("V0: "+ V0);                                      7
```

- Créér un subset

```
NodeSet V1 = new NodeSet(V0);                                        1
System.out.println("V1: "+ V1);                                      2
```

- Synchronisation des subset par rapport au "super set"

Chemins

- Construction d'un chemin

```
DiGraph g = ...;                                    1
Path p = new Path(g.getEdgeSet());                  2
                                                    3
Arc e1 = ...;                                       4
Arc e2 = ...;                                       5
Arc e3 = ...;                                       6
p.concat(e1);                                       7
p.concat(e2);                                       8
p.concat(e3);                                       9
```

- Parcourir un chemin

```
Vertex current = p.getStart();                      1
                                                    2
while (current != p.getEnd())                       3
{                                                   4
   System.out.println("Current node: " + current);  5
```

```
    Arc e = p.nextArc(current);                          6
    System.out.println("Current edge: " + e);            7
    current = p.nextVertex(current);                     8
}                                                        9
```

- Multi-path

```
Path p1 = ...;                                           1
Path p2 = ...;                                           2
boolean ok = p2.merge(p1);                               3
```

Entrées/Sorties

```
try {                                                                    1
   // creation du writer                                                 2
   MGLWriter writer = new MGLWriter(args[0]);                            3
                                                                         4
   // on ajoute ce qu'on veut ecrire (ici le graph)                      5
   writer.add(graph);                                                    6
   // on pourrai rajouter d'autres objets ...                            7
                                                                         8
   // on ecrit                                                           9
   writer.write();                                                       10
                                                                         11
} catch (java.io.FileNotFoundException fe) { fe.printStackTrace();}      12
// le tout dans un bloc try catch pq ca peut lancer des exceptions.      13
```

```
// on cree le reader.
MGLReader mglR = new MGLReader(args[0]);

try {
   mglR.parse();
} catch (Exception e) {
   System.out.println("Error when parsing file !");
}

// on recupere un iterateur sur les graphs du fichier
Iterator itG = mglR.getAbstractGraphs();

while (itG.hasNext()) {
   System.out.println("Graph read = " + itG.next());
}
```

```
GraphChooser gc = new GraphChooser();                                  1
HashMap my_graphs = null;                                              2
                                                                       3
if (args.length < 1) {                                                 4
   my_graphs = gc.getGraphHashMapGML();                                5
} else {                                                               6
   my_graphs = gc.getGraphHashMapGML(args[0]);                         7
}                                                                      8
                                                                       9
DiGraph G = (DiGraph) my_graphs.get("graph Cable");                    10
DiGraph R = (DiGraph) my_graphs.get("graph Request");                  11
```

Network files

- fichier MGL contenant:
  - ⋆ Graphe nommé “graph Request”
  - ⋆ Graphe nommé “graph Cable”
- Ces deux graphes partagent l'ensemble de nœuds
- Cette convention risque de changer. . .

- The God format

test.graph

```
c Reseau de test: cables                1
c                                       2
n 3                                     3
e 0 1                                   4
e 0 2                                   5
e 1 2                                   6
e 2 0                                   7
```

test.od

```
c Reseau de test: requetes              1
c                                       2
r 0 1 8                                 3
r 1 0 3                                 4
r 0 2 9                                 5
```

- Conversion en mgl
  - ⋆ od2mgl
  - ⋆ programme dans mascoptDev/bin
  - ⋆ utilisation:

```
java Od2MglLauncher
usage: od2mgl fileIn.graph fileIn.od fileOut.mgl [−h] [−−help]
Try 'od2mgl −−help' for more options.
```