# TD 3

## Bibliothèque d'exécution pour LF en Java

Soit la bibliothèque d'exécution Java `runtime.java` suivante décrite en cours :

```
abstract class MLvalue extends Object {
  abstract void  print();
}

class MLunit extends MLvalue {
  private int val;
  MLunit(){val=0;}

  public void print(){System.out.print("()");}
  public int MLaccess(){return val;}
}

class MLbool extends MLvalue {
  private boolean val;
  MLbool(boolean a){val=a;}

  public void print(){if (val) System.out.print("true");
                      else System.out.print("false");}
  public boolean MLaccess(){return val;}
}

class MLint extends MLvalue {
   private int val;
   MLint(int a){val=a;}

   public void print(){System.out.print(val);}
   public int MLaccess(){return val;}
}

class MLdouble extends MLvalue {
   private double val;
   MLdouble(double a){val=a;}

   public void print(){System.out.print(val);}
   public double MLaccess(){return val;}
}

class MLstring extends MLvalue {
  private String val;
  MLstring(String a){val=a;}

  public void print(){System.out.print("\""+val+"\"");}
  public String MLaccess(){return val;}
}

class MLpair extends MLvalue {
  private MLvalue MLfst;
  private MLvalue MLsnd;
  MLpair(MLvalue a, MLvalue b){MLfst=a; MLsnd=b;}

  public MLvalue MLaccess1(){return MLfst;}
```

```
  public MLvalue MLaccess2(){return MLsnd;}
  public void print(){System.out.print("(");
                      MLfst.print();
                      System.out.print(",");
                      MLsnd.print();
                      System.out.print(")");}
}

class MLlist extends MLvalue {
  private MLvalue MLcar;
  private MLlist  MLcdr;

  MLlist(MLvalue a, MLvalue b){MLcar=a; MLcdr=(MLlist)b;}
  MLlist(MLvalue a, MLlist b){MLcar=a; MLcdr=b;}

  public MLvalue MLaccess1(){return MLcar;}
  public MLlist  MLaccess2(){return MLcdr;}
  public void print(){if (MLcar == null) {System.out.print("[]");}
                      else {MLcar.print();
                            System.out.print("::");
                            MLcdr.print();}}
}

abstract class MLfun extends MLvalue {
  public int MLcounter;
  protected MLvalue[] MLenv;

  MLfun(){MLcounter=0;}
  MLfun(int n){MLcounter=0;MLenv = new MLvalue[n];}

  public  void  MLaddenv(MLvalue []O_env,MLvalue a)
   { for (int i=0; i< MLcounter; i++) {MLenv[i]=O_env[i];}
     MLenv[MLcounter]=a;MLcounter++;}

 abstract public MLvalue invoke(MLvalue x);

  public void print(){
     System.out.print("<fun> [");
     for (int i=0; i< MLcounter; i++)
           MLenv[i].print();
     System.out.print("]");
  }
} //

class MLprimitive extends MLfun {
  String name="";
  MLprimitive(String n){name=n;}

  public MLvalue invoke(MLvalue l) {
    if (name.equals("hd")) return MLruntime.MLhd_real((MLlist)l);
    else if (name.equals("tl")) return MLruntime.MLtl_real((MLlist)l);
    else if (name.equals("fst")) return MLruntime.MLfst_real((MLpair)l);
    else if (name.equals("snd")) return MLruntime.MLsnd_real((MLpair)l);
    else {System.err.println("Unknown primitive "+name); return l;}
  }
} //

class MLruntime {

// booleens
  public static MLbool MLtrue = new MLbool(true);
  public static MLbool MLfalse = new MLbool(false);
```

```
// unit
  public static MLunit MLlrp = new MLunit();
// nil
  public static MLlist MLnil = new MLlist(null,null);

// arithmetique sur les entiers
  public static MLint MLaddint(MLint x, MLint y) {
       return new MLint(x.MLaccess()+y.MLaccess());
  }
  public static MLint MLsubint(MLint x, MLint y) {
       return new MLint(x.MLaccess()-y.MLaccess());
  }
  public static MLint MLmulint(MLint x, MLint y) {
       return new MLint(x.MLaccess()*y.MLaccess());
  }
  public static MLint MLdivint(MLint x, MLint y) {
       return new MLint(x.MLaccess()/y.MLaccess());
  }
// fonction equal
  public static MLbool MLequal(MLvalue x, MLvalue y) {
       return new MLbool((x == y) || (x.equals(y)));
  }
// inegalites sur les entiers
  public static MLbool MLltint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()<y.MLaccess());
  }

  public static MLbool MLleint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()<=y.MLaccess());
  }

  public static MLbool MLgtint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()>y.MLaccess());
  }

  public static MLbool MLgeint(MLint x, MLint y) {
       return new MLbool(x.MLaccess()>=y.MLaccess());
  }

// paire
  public static MLpair MLpair(MLvalue x, MLvalue y) {
        return new MLpair(x,y);
  }

// liste
  public static MLlist MLlist(MLvalue x, MLvalue y) {
        return new MLlist(x,y);
  }

//
  public static MLvalue MLconcat(MLstring x, MLstring y) {
        return new MLstring(x.MLaccess()+y.MLaccess());
  }

// acces aux champs des paires
  public static MLvalue MLfst = new MLprimitive("fst");
  public static MLvalue MLfst_real(MLpair p) {
      return p.MLaccess1();
  }

  public static MLvalue MLsnd = new MLprimitive("snd");
  public static MLvalue MLsnd_real(MLpair p) {
```

```
      return p.MLaccess2();
  }

// acces aux champs des listes
  public static MLvalue MLhd = new MLprimitive("hd");
  public static MLvalue MLhd_real(MLlist l) {
      return l.MLaccess1();
  }

  public static MLvalue MLtl = new MLprimitive("tl");
  public static MLvalue MLtl_real(MLlist l) {
      return l.MLaccess2();
  }

// la fonction d'affichage
  public static MLvalue MLprint(MLvalue x) {
     x.print();
     System.out.println();
     return MLlrp;
  }
}
```

on cherche à implanter une bibliothèque équivalente pour C. Pour débuter le TD, on cherche à se familiariser avec cette bibliothèque d'exécution pour bien comprendre la réprésentationd es données, principalement des valeurs fonctionnelles.

1. Déterminer la hiérarchie de classes correspondant à la représentation des données choisie.
2. Construire directement en Java les données suivantes :
   (a) `(37.2, true)`
   (b) `"un" :: "jour" :: []`
   (c) `"un" :: 2 :: []`
   (d) `ref 3`
   (e) `()`
3. Ajouter un type tableau à cette bibliothèque pour des tableaux à la ML, avec les constructeurs et accesseurs habituels.
4. Chaque fonction ML sera traduite par une sous-classe de la classe `MLfun`.
   (a) En reprenant le code engendré pour la fonction `fib`, simplifiez-le à la main.
   (b) Soit la fonction `map` suivante :

```
let rec map f l =
   if l = [] then []
   else (f (hd l)) :: (map f (tl l));;
```

   écrivez à la main le code produit par le compilateur suivant les schémas de compilation indiqués dans le cours.
   (c) Que construit l'appel `map fib`?
   (d) Que construit l'appel `map fib [1;2;3]`?

## Bibliothèque d'exécution pour LF en C

On cherche à construire une bibliothèque d'exécution similaire en C.