Programming Languages

Homework Assignment 2

Announced: 3/1/2016

Due: Tuesday, 3/22/2016 5pm

Submit via Blackboard

(zip file with antlr code)

Intro

- **The assignment is to:**
  - **Design and implement a pre-processor/code injector bringing MAP/FILER/FOLD functionality into Java 7**
    - **Using ANTLR (v4)**

MAP/FILTER/FOLD

- MAP essentially takes on input a list/collection, and a piece of Java code, and returns another list, resulting from applying the code to each element of the input list.
  - E.g. take a list of strings, and a code to convert a string to upper case. The result should be a list of uppercase strings

MAP/FILTER/FOLD

- FILTER essentially takes on input a list/collection, and Java code for an expression that returns a Boolean type, and returns another list, containing only those elements of the input list for which the expression evaluates to TRUE.
  - E.g. take a list of strings, and an expression that returns true if length of a string is >6. The result should be a list of strings of lengths >6

MAP/FILTER/FOLD

- FOLD essentially takes on input a list/collection, and a piece of Java code, and returns a single value, composed of applying the code to every element in the input list.
  - E.g. take a list of string, and a code that adds length of a string to some variable that was initialized with 0. In result, that variable should contain sum of lengths of strings in the input list

Java code injection

- Your parser should:
  - Read in a java file
  - Produce a modified java file
    - If the line in input java file starts with # then inject some code (see below)
    - Other lines in the input java – just copy the to output java

Example

Input JAVA:

```
import java.util.List;
import java.util.LinkedList;
class hw2
{
  public static void main(String []args)
  {
                List<String> inCollection = new LinkedList<String>();
                inCollection.add("my");inCollection.add("simple");inCollection.add("example");

                List<String> outList1=null; //declares output list
                #outList1=MAP[inCollection,String,inElem,String,outElem,{outElem=inElem.toUpperCase();}]
                // new functionality: MAP applied to inCollection, result is in the allocated outList1
                System.out.println(outList1);
 }
}

// inElem is needed so that you have a name variable for the code, same with outElem;
// also, types of those variables (both String) are needed to be specified
// the injected code should be a block, so that whatever variables you use, they are local to the MAP code
```

Example

Output JAVA, a possibility, it's up to you how to implement MAP/FILTER/FOLD, as long as it works (efficiently):

```
import java.util.List;
import java.util.LinkedList;
class hw2
{
  public static void main(String []args)
  {
                List<String> inCollection = new LinkedList<String>();
                inCollection.add("my");inCollection.add("simple");inCollection.add("example");

                List<String> outList1=null;
                //#outList1=MAP[inCollection,String,inElem,String,outElem,{outElem=inElem.toUpperCase();}]
                {
                                List<String> outListInternal=new LinkedList<String>();
                                for (String inElem : inCollection)
                                {
                                                String outElem;
                                                outElem=inElem.toUpperCase();
                                                outListInternal.add(outElem);
                                }
                                outList1=outListInternal;
                }
                System.out.println(outList1); // should print [MY, SIMPLE, EXAMPLE]
  }
}
```

Java code injection

- Syntax of MAP/FILTER/FOLD:
- #out_list_name=MAP[in_collection_name,in_type,in_name,out_type,out_name,{java_code}]
- e.g.
- #outList1=MAP[inCollection,String,inElem,String,outElem,{outElem=inElem.toUpperCase();}]
- #outList1=MAP[inList,String,s,Integer,len,{len=s.length();}]

- #out_list_name=FILTER[in_collection_name,in_type,in_name,{java code that returns boolean}] e.g.:
- #outList2=FILTER[inCollection,String,inElem,{inElem.length()>10}]

- #out_name=FOLD[in_collection_name,in_type,in_name,{java_code}]
- e.g.:
- #outValue=FOLD[inCollection,String,inElem,{outValue+=inElem.length();}]

- In all cases the programmer would need to declare/initialize outList1, outList2, outValue. Your pre-processor doesn't have to check that – so you don't need to parse Java, only the lines starting with #

Example

Input JAVA:

```
import java.util.List;
import java.util.LinkedList;
class hw2
{
  public static void main(String []args)
  {
                List<String> inCollection = new LinkedList<String>();
                inCollection.add("my");inCollection.add("simple");inCollection.add("example");

                List<Integer> outList1=null; //declares output list
                #outList1=MAP[inCollection,String,s,Integer,len,{len=s.length();}]
                // new functionality: MAP applied to inCollection, result is in the allocated outList1
                System.out.println(outList1);
 }
}
```

Example

Output JAVA, a possibility, it's up to you how to implement MAP/FILTER/FOLD, as long as it works (efficiently):

```
import java.util.List;
import java.util.LinkedList;
class hw2
{
  public static void main(String []args)
  {
                List<String> inCollection = new LinkedList<String>();
                inCollection.add("my");inCollection.add("simple");inCollection.add("example");

                List<Integer> outList1=null;
                //#outList1=MAP[inCollection,String,s,Integer,len,{len=s.length();}]
                {
                                List<Integer> outListInternal=new LinkedList<Integer>();
                                for (String s : inCollection)
                                {
                                                Integer len;
                                                len=s.length();
                                                outListInternal.add(len);
                                }
                                outList1=outListInternal;
                }
                System.out.println(outList1); // should print [2, 6, 7]
  }
}
```

More examples

- See Blackboard for example involving FILTER and FOLD