Laboratory 6
week 6 (5 November – 9 November 2018)

TASKS:

A. Please continue to work on the assignment A3 . The deadline of A3 is week 7 (12-16 November 2018).

B. Please consult the file IE-LabAssignmentSchedule.pdf to see the schedule for the entire lab activity.

C. Please start to work on the assignment A4.

The deadline of the assignment A4 is week 8 (19-23 November 2018).

Assignment 4

You must extend the JAVA project from **A3**. Please implement the followings:

1. Heap Management: In order to implement the heap you need to add one more data structure named **Heap** into the program state. Heap is a dictionary of mappings (address, content) where the address is an integer (the index of a location in the heap) while the content is an integer value. Heap must manage the unicity of the addresses, therefore it must have a field that denotes the new free location. The addresses start from 1. The address 0 is considered an invalid address (namely null). You have to implement the Heap in the same manner as ExeStack, SymTable, FileTable and Out namely an interface plus a class that implements that interface (PrgState class will contain a field of type heap interface). Heap must be defined as as HashMap and must be integrated in all the previous functionalities (ex. Printing the prgstate in a log file).

2. Heap Allocation: Define and integrate the following statement which allocates heap memory:

new(var_name, expression)

- it is a statement which takes a variable and an expression, allocates a new cell in the heap, stores the result of the expression evaluation into the heap and returns the address of the new allocated heap into the given variable
- The statement evaluation rule is as follows:

  Stack1={new(var,exp)| Stmt2|...}
  SymTable1
  Out1
  FileTable1
  Heap1
  ==>
  Stack2={Stmt2|...}
  FileTable2=FileTable1
  Out2=Out1
  let be v=eval(exp,SymTable1,Heap1) in
  Heap2 = Heap1 U {newfreelocation ->v}
  if var exists in SymTable1 then SymTable2 = update(SymTable1,var, newfreelocation)
  else SymTable2 = add(SymTable1,var, newfreelocation)

Example: v=10;new(v,20);new(a,22);print(v)
At the end of execution: Heap={1->20, 2->22}, SymTable={v->1, a->2} and Out={1}

3. Heap Reading: Define and integrate the following expression

rH(VariableName)

- it is an expression that takes a variable name and evaluates as follows: (1) gets from the SymTable the value associated to the VariableName. This value represents the heap address. (2) uses the address computed from SymTable in order to access the Heap location and returns the content of the Heap location. If either that heap location is not allocated or VariableName is not in SymTable you have to throw an exception.
- In order to implement the evaluation of the new expression, you have to change the signature of the eval method of the expressions classes as follows

int eval(MyIDictionary<String,Integer> tbl, MyIHeap<Integer,Integer> hp)

Example: v=10;new(v,20);new(a,22);print(100+rH(v));print(100+rH(a))
At the end of execution: Heap={1->20, 2->22}, SymTable={v->1, a->2} and Out={120,122}

4.Heap Writing: Define and integrate the following statement

wH(Var_Name, expression)

- it is a statement which takes a variable and an expression, the variable contains the heap address, the expression represents the new value that is going to be stored into the heap
- The statement evaluation rule is as follows:

  Stack1={wH(var,exp)| Stmt2|...}
  SymTable1
  FileTable1
  Out1
  Heap1
  ==>
  Stack2={Stmt2|...}
  Out2=Out1
  FileTable2=FileTable1
  SymTable2=SymTable1
  let be addr=lookup(SymTable1,var) and v=eval(exp,SymTable1,Heap1) in
  if addr exists in Heap1 then Heap2 = update(Heap1, addr,v)
  else throws invalid address exception

Example: v=10;new(v,20);new(a,22);wH(a,30);print(a);print(rH(a))
At the end of execution: Heap={1->20, 2->30}, SymTable={v->1, a->2} and Out={2,30}

5. Garbage Collector: The garbage collector removes those addresses which are not refferred from the SymTable. However the estimation is very conservative since in SymTable there is not a distinction between integer values and integer addresses. We compute all the values from Sym Table and we consider that all those values can be heap addresses. Garbage collector is implemented by the following method in Controller:

```
Map<Integer,Integer> conservativeGarbageCollector(Collection<Integer> symTableValues,
Map<Integer,Integer> heap){
return heap.entrySet().stream()
        .filter(e->symTableValues.contains(e.getKey())
        .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue));}
```

The garbage collector has to be called in the method allStep() of the Controller class as follows:

```
void allStep(){
        PrgState prg = repo.getCrtPrg();
        try{
                while(true){
                   oneStep(prg);
```

```
                    prg.getHeap().setContent(conservativeGarbageCollector(
                        prg.getSymTable().getContent().values(),
                        prg.getHeap().getContent()));
                    repo. logPrgStateExec();
                }
            }
            catch(MyStmtExecException e) {}
            catch(...) ...
        }
```

Example: v=10;new(v,20);new(a,22);wH(a,30);print(a);print(rH(a));a=0
At the end of execution: Heap={1->20}, SymTable={v->1, a->0} and Out={2,30}