Laboratory 10
week 10 (4-8 December 2023)

TASKS:
A. Please submit the assignment A4.
B. Please continue to work on the assignment the assignment A5.
The deadline of the assignment A5 is week 11 (11- 15 December 2023).
B. After the Seminar 10, Please start to work on the assignment A6. The deadline of the assignment A6 is week 12 (18 - 22 December 2023).

Assignment 6

You must extend the JAVA project from the assignment A5. You have to implement a Type Checker for your ToyLanguage programs using the rules discussed in Seminar 10. Therefore you have to do the following modifications:

1. **Interface Exp:** please add the following method to it:
Type typecheck(MyIDictionary<String,Type> typeEnv) throws MyException

2. **All Expression Classes must implement the method typecheck**:

```
class ValueExp implements Exp{
        Value e;
       ....
  Type typecheck(MyIDictionary<String,Type> typeEnv) throws MyException{
          return e.getType();
   }
}

class VarExp implements Exp{
        String id;
        ...
   Type typecheck(MyIDictionary<String,Type> typeEnv) throws MyException{
          return typeEnv.lookup(id);
   }
}

class ArithExp  implements Exp{
  Exp e1;
  Exp e2;
   ....
   Type typecheck(MyIDictionary<String,Type> typeEnv) throws MyException{
          Type typ1, typ2;
          typ1=e1.typecheck(typeEnv);
          typ2=e2.typecheck(typeEnv);
```

```
                if typ1.equals(new IntType()) {
                        if typ2.equals(new IntType()) {
                                return new IntType();
                        } else
                            throw new MyException("second operand is not an integer");
                }else
                        throw new MyException("first operand is not an integer");

        }
}
```

Please implement in a similar way the method typecheck for the LogicExp class and RelationalExp class.

```
class RHExp  implements Exp{
        Exp e;
                ...
         Type typecheck(MyIDictionary<String,Type> typeEnv) throws MyException{
                Type typ=e.typecheck(typeEnv);
                if (typ instanceof RefType) {
                        RefType reft =(RefType) typ;
                         return reft.getInner();
                } else
                    throw new MyException("the rH argument is not a Ref Type");
        }

}
```

3. **Interface Istmt:** please add the following method to it:

```
        MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException
```

4. **All Statement Classes must implement the method typecheck**:

```
class CompStmt implements IStmt {
         IStmt first;
          IStmt snd;
          .....
         MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException{
                //MyIDictionary<String,Type> typEnv1 = first.typecheck(typeEnv);
                 //MyIDictionary<String,Type> typEnv2 = snd.typecheck(typEnv1);
                 //return typEnv2;
                 return snd.typecheck(first.typecheck(typeEnv));
         }
}
```

```
class PrintStmt implements IStmt{
   Exp exp;
    ....
        MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException{
                exp.typecheck(typeEnv);
                return typeEnv;
        }
}


 class AssignStmt implements IStmt{
            String id;
   Exp exp;
    ....
        MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException{
                Type typevar = typeEnv.lookup(id)
                Type typexp = exp.typecheck(typeEnv);
                if (typevar.equals(typexp))
                        return typeEnv;
                else
                        throw new MyException("Assignment: right hand side and left hand side
have different types ");
        }
}


 class VarDeclStmt implements IStmt{
            string name;
             Type typ;
            .....
MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException{

                typeEnv.add(name,typ);
                return typeEnv;
}
}



class IfStmt implements IStmt{
   Exp exp;
   IStmt thenS;
   IStmt elseS;
        .....
        MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException{
                Type typexp=exp.typecheck(typeEnv);
                if (typexp.equals(new BoolType())) {
                        thenS.typecheck(clone(typeEnv));
```

```
                elseS.typecheck(clone(typeEnv));
                return typeEnv;
            }
             else
                throw new MyException("The condition of IF has not the type bool");
        }
}
```

Please implement in the same manner forkStmt and whileStmt.

```
class NewStmt implements IStmt{
            String id;
   Exp exp;
    ....
        MyIDictionary<String,Type> typecheck(MyIDictionary<String,Type> typeEnv) throws
MyException{
            Type typevar = typeEnv.lookup(id)
            Type typexp = exp.typecheck(typeEnv);
            if (typevar.equals(new RefType(typexp)))
                return typeEnv;
            else
                throw new MyException("NEW stmt: right hand side and left hand side have
different types ");
        }
}
```

Please implement the method typecheck for all the other statements.

5. **Please call the method typecheck for the input program before you create its associated PrgState.** The execution is done only if the program passes the typechecker.