- Sym.java

```
public class Sym {

    private String type;

    /**
     * constructor  initialize the Sym to have the given type
     * @param type
     */
    public Sym(String type) {
        this.type = type;
    }

    /**
     * Return this Sym's type.
     * @return
     */
    public String getType() {
        return type;
    }

    /**
     * Return this Sym's type.
     * @return
     */
    @Override
    public String toString() {
        return type;
    }
}
```

- SymTable.java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * @author
 * @date 2019-01-28 13:30
 */
public class SymTable {
    private List<HashMap<String, Sym>> tables;
```

```
    /**
     * constructor  initialize the SymTable's List field to contain a
single, empty HashMap.
     */
    public SymTable(){
        tables = new ArrayList<>();
        tables.add(new HashMap<>());
    }

    /**
     * If this SymTable's list is empty, throw an
EmptySymTableException.
     * If either idName or sym (or both) is null, throw a
WrongArgumentException
     * If the first HashMap in the list already contains the given id
name as a key, throw a DuplicateSymException.
     * add the given idName and sym to the first HashMap in the list
     * @param idName
     * @param sym
     * @throws DuplicateSymException
     * @throws EmptySymTableException
     * @throws WrongArgumentException
     */
    public void addDecl(String idName, Sym sym) throws
DuplicateSymException,EmptySymTableException,WrongArgumentException{
        if (tables.isEmpty()) {
            throw new EmptySymTableException();
        }
        if (null == idName && null != sym) {
            throw new WrongArgumentException("Id name is null.");
        }
        if (null != idName && null == sym) {
            throw new WrongArgumentException("Sym is null.");
        }
        if (null == idName && null == sym) {
            throw new WrongArgumentException("Id name and sym are
null.");
        }
        for (HashMap<String, Sym> map : tables) {
            if (map.get(idName) != null) {
                throw new DuplicateSymException();
            }
        }
        HashMap<String, Sym> map = tables.get(0);
        map.put(idName, sym);
    }

    /**
```

```
     * Add a new, empty HashMap to the front of the list.
     */
    public void addScope() {
        tables.add(0, new HashMap<>());
    }

    /**
     * If this SymTable's list is empty, throw an
EmptySymTableException.
     * if the first HashMap in the list contains id name as a key,
return the associated Sym; otherwise, return null.
     * @param idName
     * @return
     * @throws EmptySymTableException
     */
    public Sym lookupLocal(String idName) throws EmptySymTableException
{
        if (tables.isEmpty()) {
            throw new EmptySymTableException();
        }
        HashMap<String, Sym> firstMap = tables.get(0);
        return firstMap.get(idName);
    }

    /**
     * If this SymTable's list is empty, throw an
EmptySymTableException.
     * If any HashMap in the list contains idName as a key, return the
first associated Sym (i.e., the one from the HashMap that is closest to
the front of the list); otherwise, return null.
     * @param idName
     * @return
     * @throws EmptySymTableException
     */
    public Sym lookupGlobal(String idName) throws EmptySymTableException
{
        if (tables.isEmpty()) {
            throw new EmptySymTableException();
        }
        for (HashMap<String, Sym> map : tables) {
            if (map.get(idName) != null) {
                return map.get(idName);
            }
        }
        return null;
    }

    /**
```

```
     * If this SymTable's list is empty, throw an
EmptySymTableException.
     * remove the HashMap from the front of the list.
     * @throws EmptySymTableException
     */
    public void removeScope() throws EmptySymTableException {
        if (tables.isEmpty()) {
            throw new EmptySymTableException();
        }
        tables.remove(0);
    }

    /**
     * print the elements in the tables
     */
    public void print() {
        System.out.println("\n=== Sym Table ===\n");
        for (HashMap<String, Sym> M : tables) {
            System.out.println(M.toString());
        }
    }
}
```

- DuplicateSymException.java

```
public class DuplicateSymException extends Exception{
}
```

- EmptySymTableException.java

```
public class EmptySymTableException extends Exception {
}
```

- WrongArgumentException.java

```
public class WrongArgumentException extends Exception{
    public WrongArgumentException(String message) {
        super(message);
    }
}
```

- P1.java

```java
/**
 * Test SymTable
 * @author
 * @date 2019-01-28 13:56
 */
public class P1 {
    public static void main(String[] args) {
        SymTable symTable = new SymTable();
        // print the initialize SymTable
        symTable.print();
        System.out.println("-----------------------");

        // addDecl throw WrongArgumentException when idName and sym is
null
        try {
            symTable.addDecl(null, null);
        } catch (EmptySymTableException | WrongArgumentException |
DuplicateSymException e) {
            printException(e);
        }

        // addDecl throw WrongArgumentException when idName is null
        try {
            symTable.addDecl(null, new Sym("a"));
        } catch (EmptySymTableException | WrongArgumentException |
DuplicateSymException e) {
            printException(e);
        }

        // addDecl throw WrongArgumentException when sym is null
        try {
            symTable.addDecl("a", null);
        } catch (EmptySymTableException | WrongArgumentException |
DuplicateSymException e) {
            printException(e);
        }

        try {
            symTable.addDecl("a", new Sym("A"));
            symTable.addDecl("b", new Sym("B"));
            // the line code will throw DuplicateSymException
            symTable.addDecl("a", new Sym("A"));
        } catch (EmptySymTableException | WrongArgumentException |
DuplicateSymException e) {
            printException(e);
        }
```

```
        symTable.print();
        System.out.println("-----------------------------------");

        // test lookupLocal
        try {
            Sym symA = symTable.lookupLocal("a");
            //will print A
            System.out.println(symA);
            Sym symEmpty = symTable.lookupLocal("abc");
            //will print null
            System.out.println(symEmpty);
        } catch (EmptySymTableException e) {
            printException(e);
        }

        System.out.println("--------------------------");

        // test add Scope
        symTable.addScope();
        symTable.print();
        System.out.println("--------------------------");

        // test lookupGlobal
        try {
            Sym symA = symTable.lookupGlobal("a");
            // will print A
            System.out.println(symA);
            Sym symEmpty = symTable.lookupGlobal("aa");
            // will print null
            System.out.println(symEmpty);
        } catch (EmptySymTableException e) {
            printException(e);
        }

        System.out.println("-------------------------------");

        try {
            symTable.removeScope();
        } catch (EmptySymTableException e) {
            printException(e);
        }

        symTable.print();
        System.out.println("-------------------------------");

        // remove again and the tables in SymTable is empty
        try {
            symTable.removeScope();
        } catch (EmptySymTableException e) {
```

```
            printException(e);
        }

        // test addDecl EmptySymTableException
        try {
            symTable.addDecl("c", new Sym("C"));
        } catch (EmptySymTableException | WrongArgumentException |
DuplicateSymException e) {
            printException(e);
        }

        // test lookupLocal EmptySymTableException
        try {
            symTable.lookupLocal("a");
        } catch (EmptySymTableException e) {
            printException(e);
        }

        // test lookupGlobal EmptySymTableException
        try {
            symTable.lookupGlobal("a");
        } catch (EmptySymTableException e) {
            printException(e);
        }

        // test removeScope EmptySymTableException
        try {
            symTable.removeScope();
        } catch (EmptySymTableException e) {
            printException(e);
        }
    }

    private static void printException(Exception e) {
        if (e instanceof EmptySymTableException) {
            System.out.println("Sym Table is empty");
        }
        if (e instanceof WrongArgumentException) {
            System.out.println(e.getMessage());
        }
        if (e instanceof DuplicateSymException) {
            System.out.println("duplicate elements in Sym Table");
        }
    }
}
```

- To test my SymTable implementation, run the P1.java.

```
javac DuplicateSymException.java
javac EmptySymTableException.java
javac WrongArgumentException.java
javac Sym.java
javac SymTable.java
javac P1.java
java P1
```