SNLJ OPERATOR

```java
package query;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.NoSuchElementException;

import database.Database;
import database.DatabaseException;
import databox.DataBox;
import table.Record;

public class SNLJOperator extends JoinOperator {

  public SNLJOperator(QueryOperator leftSource,
                      QueryOperator rightSource,
                      String leftColumnName,
                      String rightColumnName,
                      Database.Transaction transaction) throws QueryPlanException,
                      DatabaseException {
    super(leftSource,
          rightSource,
          leftColumnName,
          rightColumnName,
          transaction,
          JoinType.SNLJ);
  }

  public Iterator<Record> iterator() throws QueryPlanException, DatabaseException {
    return new SNLJIterator();
  }


  /**
   * An implementation of Iterator that provides an iterator interface for this
   operator.
   * Note that the left table is the "outer" loop and the right table is the
   "inner" loop.
   */
  private class SNLJIterator implements Iterator<Record> {
    private Iterator<Record> leftIterator;
    private Iterator<Record> rightIterator;
    private Record leftRecord;
    private Record nextRecord;

    public SNLJIterator() throws QueryPlanException, DatabaseException {
      this.leftIterator = SNLJOperator.this.getLeftSource().iterator();
      this.rightIterator = null;
      this.leftRecord = null;
      this.nextRecord = null;
    }

    /**
     * Checks if there are more record(s) to yield
     *
     * @return true if this iterator has another record to yield, otherwise false
     */
    public boolean hasNext() {
      if (this.nextRecord != null) {
        return true;
      }
      while (true) {
        if (this.leftRecord == null) {
          if (this.leftIterator.hasNext()) {
            this.leftRecord = this.leftIterator.next();
            try {
              this.rightIterator = SNLJOperator.this.getRightSource().iterator();
```

```
65                } catch (QueryPlanException q) {
66                  return false;
67                } catch (DatabaseException e) {
68                  return false;
69                }
70              } else {
71                return false;
72              }
73            }
74            while (this.rightIterator.hasNext()) {
75              Record rightRecord = this.rightIterator.next();
76              DataBox leftJoinValue =
                this.leftRecord.getValues().get(SNLJOperator.this.getLeftColumnIndex());
77              DataBox rightJoinValue =
                rightRecord.getValues().get(SNLJOperator.this.getRightColumnIndex());
78              if (leftJoinValue.equals(rightJoinValue)) {
79                List<DataBox> leftValues = new
                  ArrayList<DataBox>(this.leftRecord.getValues());
80                List<DataBox> rightValues = new
                  ArrayList<DataBox>(rightRecord.getValues());
81                leftValues.addAll(rightValues);
82                this.nextRecord = new Record(leftValues);
83                return true;
84              }
85            }
86            this.leftRecord = null;
87          }
88        }
89
90        /**
91         * Yields the next record of this iterator.
92         *
93         * @return the next Record
94         * @throws NoSuchElementException if there are no more Records to yield
95         */
96        public Record next() {
97          if (this.hasNext()) {
98            Record r = this.nextRecord;
99            this.nextRecord = null;
100           return r;
101         }
102         throw new NoSuchElementException();
103       }
104
105       public void remove() {
106         throw new UnsupportedOperationException();
107       }
108     }
109   }
110
```