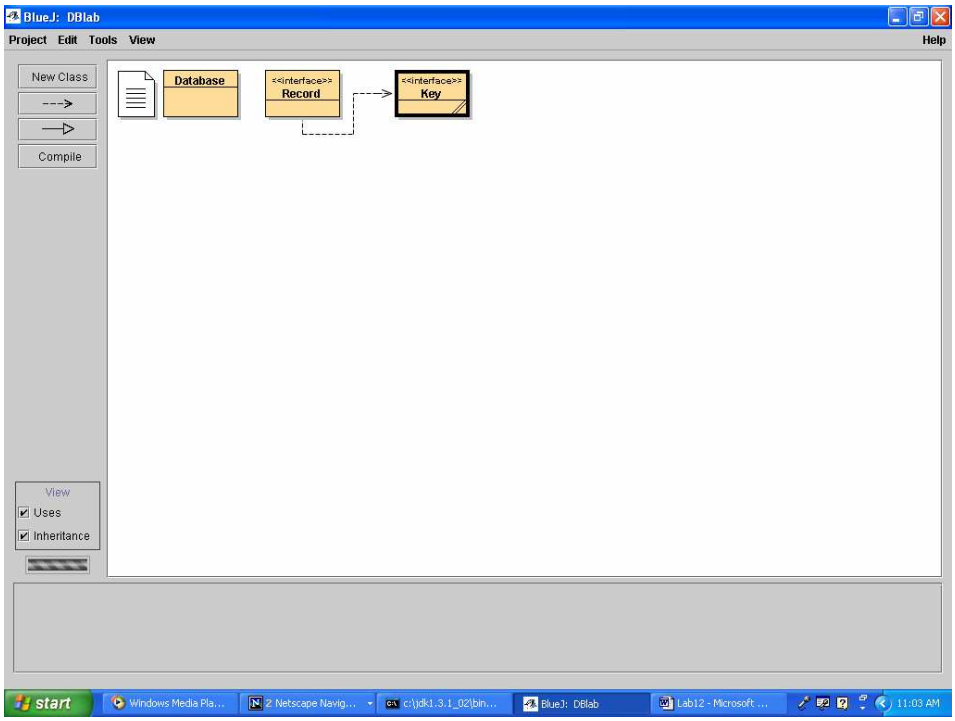
Lab Assignment 12 (week 12)

In this lab, we will:

1. Review the contents of the Database.jar file
2. Cover the usage of arrays to create a database structure
3. Show how the Database class is used and why
4. Have some time for help with the homework

Specification:

The creation of a new project, ***DBlab***, will look like the following:



The lab instructor will go through each class and explain the flow and essentially what is happening and how it relates to other parts of the class.

Record Interface

```
/**
 * Record is a data item that can be stored in a database
 *
 */
public interface Record {

        /**
         * keyOf returns the key that uniquely identifies the record
         * @return      the key          */
```

```
        Key keyOf();
}
```

Key Interface

```
/**
 * Key is an identification, or "key", value
 */

public interface Key {
        /**
         * equals compares itself with another key, m, for equality
         * @param m  - the other key
         * @return  true, if this key and m have the same key value;
         * otherwise, return false.       */
        boolean equals(Key m);

        /**
         * lessthan compares itself with another key, m, for less than
         * @param m  - the other key
         * @return  true, if this key is a lesser key value than m;
         * otherwise, return false.       */
        boolean lessthan(Key m);
}
```

Database Class

```
/** Database  implements a database of records */
public class Database
{ private Record[] base;        // the collection of records
  private int count;            // how many records are stored in the database
                                // invariant:  0 <= item_count <= base.length


  /** Constructor  Database  initializes the database
    * to a size of 10 unless otherwise specified */
  public Database()  {
        this(10);
  }

  /** Constructor  Database  initializes the database
    * @param initial_size - the size of the database */
  public Database(int initial_size)
  { if ( initial_size > 0 )
       { base = new Record[initial_size]; }
    else { base = new Record[1]; }
    count = 0;
```

```
}

/** insert inserts a new record into the database.
  * @param r - the record
  * @return true, if record added; return false if record not added because
  *   another record with the same key already exists in the database */
public boolean insert(Record r)
{ boolean success = false;
  if ( locationOf(r.keyOf()) == -1 )  // ok to add record with this key?
     { boolean found_empty_place = false;
       int i = 0;
       while ( !found_empty_place  &&  i != base.length )
            // so far, all of  base[0]..base[i-1]  are occupied
            { if ( base[i] == null )   // is this element empty?
                 { found_empty_place = true; }
              else { i = i+1; }
            }
       if ( found_empty_place )
           { base[i] = r; }
       else { // array is full!  So, create a new one to hold more records:
              Record[] temp = new Record[base.length * 2];
              for ( int j = 0;  j != count;  j = j+1 )
                 // copying contents of  base  into  temp
                 { temp[j] = base[j]; }
              base = temp;   // change  base  to hold address of  temp
              base[count] = r;   // insert  r  in first free element
            }
       count = count + 1;    // remember that we added a record
       success = true;
     }
  return success;
}

/** find  locates a record in the database based on a key
  * @param key - the key of the desired record
  * @return (the address of) the desired record;
  *  return  null if record not found.  */
public Record find(Key k)
{ Record answer = null;
  int index = locationOf(k);
  if ( index != -1 )
     { answer = base[index]; }
  return answer;
}

/** delete removes a record in the database based on a key
```

```
   * @param key - the record's key (identification)
   * @return true, if record is found and deleted; return false otherwise  */
  public boolean delete(Key k)
  { boolean result = false;
    int index = locationOf(k);
    if ( index != -1 )
       { base[index] = null;
         count = count - 1;  // remember that we deleted a record
         result = true;
       }
    return result;
  }

  /** locationOf returns the index in base where a record with  k  appears*/
  private int locationOf(Key k)
  { int result = -1;
    boolean found = false;
    int i = 0;
    while ( !found  &&  i != base.length )
          { if ( base[i] != null  &&  (base[i].keyOf().equals(k)))
                { found = true;
                  result = i;
                }
            else { i = i+1; }
          }
    return result;
  }

  /** getDatabase returns the database so the user can sort it, etc.
   * @returns base
   */
  public Record[] getDatabase() {
          return base;
  }
}
```

Now that the instructor has explained the details of the Database class, take time to work through this week's homework assignment!