the design of the database is following the instruction of the bd assignment.

1.The First Stage

First, the program starts with record class

1.1 The record class

The record is designed to be a String type Arraylist which enables the new elements to be added without limit. The methods in record class include add the string to the end of the record, get the string in the record by specifying the index, set String at the specified index, calculate the record length(i.e the string number in the record) and remove one element or all elements.

1.2 The table class

The table class is also designed to be an ArrayList containing records, which give functions to operate both row elements(record) and column elements(field), the methods related to record include get record, update record, add record, remove record, insert record, there is also method which could operate multiple records at the same time by specifying the row number. The method related to fields includes set fields, add columns, change column name(update columns) . Besides, the method related to columns has also been designed to receive users' inputs, however, these methods seem to be hard to test so these code hasn't been adopted.

1.3 The print class

The print class only contains a method to print the table through a for loop.

1.4 The file class

Files class contains methods of reading and writing. A txt file can be read into a table, and a table could also be written into a txt file, the "writeRecord" method enables a record to be written to the end of an existing file.

By now, a valid program has been designed, however, there are some problems, for example, the table class and record class has the responsibility for printing and the program is only examined by check the printed results, so some code in Record class and Table class should be moved out and tests are needed.

2.The Second Stage

In this part, the main changes are: 1.set a column as key in the record; 2.modified the Print class

2.1set key

In each record, the first element of the record is taken as a key, which should be unique. When adding, inserting a new record, the key valued should be checked, if the key value is the same as the key value of the existing class, the add/insert will be failed. The following are the codes:

```
// add a record to table
void addRecord(Record r){
  String key = r.getString(0);
  for(int i = 0; i < table.size(); i++){
     Record temp = table.get(i);
     if (temp.getString(0)==key){
       return;
     }
  }
  table.add(r);
}
```

```
//insert a reacord by specifying row number, the key(first column) is ckecked when inserting
boolean insertRecord(int index, Record r){
    String key = r.getString(0);
    for(int i = 0; i < table.size(); i++){
        Record temp = table.get(i);
        if (temp.getString(0)==key){
        return false;
     }
  }
    table.add(index, r);
    return true;
}
```

When updating a record, the key value should also be different from that of other records, but the key value could be the same as the original record, the code is as follows:

```java
//update a record by row number, the key(first column) is also checked when updating
boolean updateRecord(int index, Record r){
    String key = r.getString(0);
    for(int i = 0; i < table.size(); i++){
       Record temp = table.get(i);
       if (temp.getString(0)==key && i!=index){
         return false;
       }
    }
    table.set(index, r);
    return true;
}
```

The record can also be select ed or deleted by specifying the key value:

```java
//delect the column name and its contents by specifying its column name
void delectCol(String s){
    int flag = 0;
    Record temp_record = table.get(0);
    for(int i = 0; i < temp_record.getLength(); i++){
        if(s.equals(temp_record.getString(i))){
            delectColumn(i+1);
        }else return;
    }
}
```

```java
    //delect multiple record by specifying the key value
    void delectMultRecordByKey(String [] array){
      ArrayList<Record> collection = new ArrayList<Record>();
      for(String temp_str:array){
          for (int i = 0; i < table.size(); i++){
              Record temp = table.get(i);
              if(temp.getString(0)==temp_str){
                  collection.add(temp);
              }
          }
      }
      table.removeAll(collection);
    }
```

2.2Modified Print class

The Print class is also modified. First, in order to print a table neatly, there should be enough space for each element, especially for very long elements. At first, the space for each element was set as “%10s”, however, it doesn’t work well when the elements

are long. So, in this stage, the printTable method has been modified. First of all, we get the length of the longest elements in the table. Then, when printing, the space for each element is set to the same (the longest string length plus one). So, in the table class, the method of getting the maximum string length is also added. The code is shown as follows:

```
//find the longest string in the table by checking all records and return string length
int getMaxStringLengthInTable(){
    int init =0;
    for(int i =0; i < table.size(); i++){
        Record temp = table.get(i);
        int MaxStringLengthInRecord = temp.getMaxStringLength();
        if(MaxStringLengthInRecord > init){
            init = MaxStringLengthInRecord;
        }
    }
    return init;
}
```

The method calls the method of getting maximum string length in the record, the method getMaxStringLength is shown as follows:

```
//get the longest string in the record and return its length
int getMaxStringLength(){
    int init = 0;
    for(int i = 0; i< record.size(); i++){
        String temp = record.get(i);
        if(temp.length() > init){
            init = temp.length();
        }
    }
    return init;
}
```

The former print table method:

```
public void print_table(Table t){
    for(int i = 0; i < t.count_row(); i++){
        Record temp = t.select_record(i);
        for(int j = 0; j < temp.get_length(); j++){
            String temp_str = temp.get_string(j);
            System.out.printf("%-10s", temp_str);
        }
        System.out.printf("\n");
    }
```

The modified print table method:

```
//print table
void printTable(Table t){
    int maxStringLength = t.getMaxStringLengthInTable();
    for(int i = 0; i < t.countRow(); i++){
        Record temp = t.selectRecord(i);
        for(int j = 0; j < temp.getLength(); j++){
            String tempStr = temp.getString(j);
            System.out.printf(tempStr);
            for(int k =0; k < maxStringLength-tempStr.length(); k++){
                System.out.printf(" ");
            }
        }
        System.out.printf("\n");
    }
}
```

The print record method and help message are also added in the print class

2.3The database class

In this stage, the database class just shows the functions of record, table, files.

The unit tests are added in this stage

3.The Third Stage

In the third stage, the main alternation is in Database class, a hash map is added in the Database to store different tables, each table has its own key, which is a convenience to operate the table. The methods in Database class include add a table into the database, delete a table, get a table and get all keys in the database.

```
Map<String, Table> database = new HashMap<>();

//add table to database
void addTable(String s, Table t){
    database.put(s, t);
}

//print all keys as a string
void getKey(){
    System.out.println(database.keySet().toString());
}

//get table by specifying the key
Table getTable(String s){
    return database.get(s);
}

//delect table by soecifying the key
void delectTable(String s){
    database.remove(s);
}
```

The query class shows a textual interface, which enables the user to input command and gets results. Due to the time limitation, the Query class can only realize a few functions and the the input and output interface will be moved to Print class in the future.