Basic:

There have 5 classes to implement the basic Database usage

- Record: With a String type array list
    1. **Record(String... data)**: new an array of String list to records
    2. **getRecords()**: To return record String Array Record_size(): get how many columns
- Table: with a Field array and Attrs array with Record type
    1. **Table(String name, Record field)**: new a table name and field to the table
    2. **getFields()**: To get the field's array and only can have one record
    3. **getAttrs(int num)**: to get the key of Attrs
    4. **getAttrsSize(String... values)**: return the number of Attrs
    5. **addRecord**: add new attr and if the size of record in attr is not equal the field and throw error
    6. **removeRecord(int num)**: set the Record to null.
    7. **updateRecord(int num, String... values)**: update the existed record
    8. **printTable()**
    9. **Keys** is going to make sure the data is unique.
- ImportData
    1. **ImportData(String name)**: open the file and add to record array. The format is `id, name, student` which is used comma to split each cell and new line to change new row getData():return arraylist of record
- WriteData
    1. **write(String str, String filename)**: Test to write str into file
    2. **writeTable**: write field and attrs into field and followed the same format of the importData's format.
- Database
    1. **Database(String name)**: Create a database folder if database is not existed
    2. **CreateTable(String name, Record field)**: Create a table folder if it's not existed.

Extension

- Type: the constant of different data type
- DataType the method to store data and define the data type
- Record
    1. Change the String Arraylist to DataType Arraylist
    2. add **getRecordsType()** method
    3. add **getIndexofRecord()** and **getIndexofRecordType** method

- Table
    1. addRecord method's name change to **insertRecord**
    2. add check data type method, such as **isInt()** nd **isFloat()**, to make sure data is correct type.

Refactoring

- String ArrayList change to DataType ArrayList
    ```
    ArrayList<String> getRecords() {
        // save string data into array list and return it.
        ArrayList<String> recordData = new ArrayList<String>();
        for(DataType record: this.records){recordData.add(record.getdata());}
        return recordData;
    }
    ```
- Before inserting data, it will be checked type is correct
    ```
    // Insert new Record and Check Type is correct
    void insertRecord(DataType... values){
        if(values.length!=fields.get(0).Record_size()){
            throw new IndexOutOfBoundsException();
        }
        for(DataType e: values){
            if(e.getType()==Type.INT && !isInt(e.getdata())){
                throw new NumberFormatException("Not Integer");
            }
            if(e.getType()==Type.FLOAT && !isFloat(e.getdata())){
                throw new NumberFormatException("Not Float");
            }
        }
        Record attr = new Record(values);
        attrs.add(Keys,attr);
        Keys ++ ;
    }
    ```
- A format to make load data into table and write data into file
    ```
    while(input.hasNextLine())
    {
        String line = input.nextLine();
        String [] list = line.split(",");
        if (i == 0)
        {
            Record fieldTmp = new Record();
            for(String e: list){
                String [] fieldlist = e.trim().replace('(', ' ').replace(')', ' ').split(" ");
                fieldTmp.setRecord(new DataType(fieldlist[0],checkType(fieldlist[1])));
            }
            field.add(fieldTmp);
    ```

```
void writeTable(String filename,ArrayList<Record> field ,ArrayList<Record> attrs){
    try {
        BufferedWriter writer = new BufferedWriter(new FileWriter(filename));
        for (Record line: field){
            for (int i=0; i<line.Record_size(); i++)
            {
                String str = line.getIndexofRecord(i);
                Type type = line.getIndexofRecordType(i);
                if (i==line.Record_size()-1){writer.write(str+"("+type.toString()+")");}
                else{writer.write(str+"("+type.toString()+")"+", ");}
            }
            writer.write("\n");
        }
```

Testing Method

Manually add testing data

- Record Testing

```
public static void main(String[] args) {
    DataType data1 = new DataType("id", Type.INT);
    DataType data2 = new DataType("name", Type.STR);
    DataType data3 = new DataType("score", Type.FLOAT);
    Record program = new Record(new DataType[]{data1,data2,data3});
    program.run();
}
private void run (){
    testGetRecord();
    testGetRecordType();
}
private void testGetRecord()
{
    getIndexofRecord(0).equals("id");
    getIndexofRecord(1).equals("name");
    getIndexofRecord(2).equals("score");
}
private void testGetRecordType()
{
    getIndexofRecordType(0).equals(Type.INT);
    getIndexofRecordType(1).equals(Type.STR);
    getIndexofRecordType(2).equals(Type.FLOAT);
}
```

- Table Testing

```
107   public static void main(String[] args) {
108       DataType data1 = new DataType("id", Type.INT);
109       DataType data2 = new DataType("name", Type.STR);
110       DataType data3 = new DataType("score", Type.FLOAT);
111       Record field = new Record(new DataType[]{data1,data2,data3});
112       Table program = new Table("Student",field);
113       program.run();
114       program.printTable();
115       Record field2 = new Record(new DataType[]{data1,data2,data3});
116       Table program2 = new Table("Student",field2);
117       program2.loadData("data.txt");
118       program2.printTable();
119   }
120   private void run()
121   {
122       addAttrsTest();
123       removeAttrsTest();
124       updateAttrsTest();
125   }
```

1. Insert Testing

```
126   private void addAttrsTest()
127   {
128       DataType data1 = new DataType("1", Type.INT);
129       DataType data2 = new DataType("justin", Type.STR);
130       DataType data3 = new DataType("0.1", Type.FLOAT);
131       DataType datand1 = new DataType("2", Type.INT);
132       DataType datand2 = new DataType("ben", Type.STR);
133       DataType datand3 = new DataType("0.2", Type.FLOAT);
134       insertRecord(new DataType[]{data1,data2,data3});
135       insertRecord(new DataType[]{datand1,datand2,datand3});
136       assert(getAttrsSize()==2);
137   }
```

2. Remove Testing

```
138   private void removeAttrsTest()
139   {
140       removeRecord(1);
141       assert(getAttrsSize()==2);
142       assert(getAttrs(1)==null);
143   }
```

3. Update Testing

```
144   private void updateAttrsTest()
145   {
146       DataType data1 = new DataType("2", Type.INT);
147       DataType data2 = new DataType("ben", Type.STR);
148       DataType data3 = new DataType("1.2", Type.FLOAT);
149       DataType datand1 = new DataType("3", Type.INT);
150       DataType datand2 = new DataType("Den", Type.STR);
151       DataType datand3 = new DataType("1.2", Type.FLOAT);
152       DataType datard1 = new DataType("4", Type.INT);
153       DataType datard2 = new DataType("Jen", Type.STR);
154       DataType datard3 = new DataType("1.2", Type.FLOAT);
155       DataType datachange1 = new DataType("4", Type.INT);
156       DataType datachange2 = new DataType("Jen", Type.STR);
157       DataType datachange3 = new DataType("1.2", Type.FLOAT);
158       insertRecord(new DataType[]{data1,data2,data3});
159       insertRecord(new DataType[]{datand1,datand2,datand3});
160       insertRecord(new DataType[]{datard1,datard2,datard3});
161       updateRecord(3, new DataType[]{datachange1,datachange2,datachange3});
162
163   }
```

Future Improvement

- Separate to Field and Attributes Class
- Probably create generic class
- Extend Key usage by using Map class