# *In Memory Data Base Project*

## *Java and DevOps Training*

***Prepared by: Omar Abdel Majeed Ababneh***

***Email: omarababneh1010@gmail.com***

## *Introduction*:

In this Project I will explain How to build In memory data base and how to implement principles in my own code such as design pattern and solid principle and more.

In this Report I will try to ask some question and I will answer about it.

I Build Student Data base.

### *1- What is In memory data base (Caching Layer):*

It is layer to store data in memory usually we store an expansive request in memory and we try to access in data quickly so we must used in memory data base intend HDD because it is slowly.

### *2-Why we needing caching layer:*

1- Faster response

The cache in memory data store, the results are faster than the result which we get from disk also saves an entire network round trip to database and complex queries.

2- Availability of data during network interruption:

When there are network fluctuations between application and database user can still view data in cache database.

### *3-how to Create In memory database (How to build Caching Layer):*

I used Jedis Library (Redis as a cache using java).

Why did I use Jedis?

Because it is small and considerably faster than the other Libraries. Besides, it is the client library of choice of the Spring Framework developers, and it has the biggest community of all three.

I want to explain cache data adding strategy There are certain ways how we can add data in our cache data base depends on our requirements.

I have will mention what I used in my own code.

I tried to implement write through caching strategy:

Storing at two places increase the latency at the time of write but data is consistent(updated) at every read.

Why I used this strategy?  because I guarantee data found on disk.

At any time the server stops for some reason, all the data will be on the hard disk I achieved Durability.

I want to explain cache eviction strategy and max memory limit:

An eviction policy enables a cache to ensure that the size of the cache does not exceed the maximum limit to achieve this, existing element are removed from a cache depending on the eviction policy but it can be customized as per application requirements there are multiple ways through which data can be evicted from the cache I used LRU algorithm to eviction policy.

Why I did to use LRU algorithm?

Because it is One of the most used strategy.

In most caching use cases application access the same data again and again.

For LFU algorithm it is not used that often as it does not account for an item that had an initially high access rate and then was not accessed for a long time.

Selecting the cache size is important.

Because if it is set to too small of value, Project's performance will suffer from too much disk I/O.

On the other hand If cache is too large, then Project will use more memory than it actually needs.

So the best way to determine how large cache needs to be is to put Project into production environment and watch to see how much disk I/O is occurring

If it is going to disk quite a lot to retrieve database records, then we should increase the size of our cache.

## *4-why did I Used Mysql not file to store data as primary database?*

difference between file system and MYSQL:

1- A file processing system is a collection of programs that store and manage files in computer hard-disk. On the other hand, A database management system is collection of programs that enables to create and maintain a database.

2-File processing system has more data redundancy, less data redundancy in MYSQL.

3-File processing system provides less flexibility in accessing data, whereas MYSQL has more flexibility in accessing data.

4-File processing system does not provide data consistency, whereas MYSQL provides data consistency through normalization.

5-File processing system is less complex, whereas MYSQL is more complex.

## *6- How to wrote my Code and Used Multithreading and non-blocking data structure:*

In this section I am using my code to explain some points

```
public class Client {
    private static ClientCommunication clientCommunication=null;
    private static Display  display=null;
    public static void main(String[] args) {
        display=new Display();
        clientCommunication=ClientCommunication.getClientCommunication();
        clientCommunication.init();

        if(isClient()){
            System.out.println("Hello You present in the Student
Information");
            display.SelectOperation();
        }
        else {
            System.out.println("Iam Sorry you are not register in this
service");
        }
       clientCommunication.close();
    }
    public static Boolean isClient(){
        return clientCommunication.CommunicateServerInOrderLogin();
    }

}
```

this class communicate with server initially verification if user has authorization using function (is Client) if yes navigate user to (select Operation) to choose certain operation

```
public class Display {
    private ClientCommunication clientCommunication;
    private DataInput dataInput;
    public Display(){
        clientCommunication=ClientCommunication.getClientCommunication();
        dataInput=new DataInput();
    }
    public void SelectOperation(){
        Boolean isFound=true;
        while (isFound){
            ShowMenu();
            int OperationNumber=dataInput.getOperationNumber();
            switch (OperationNumber){
                case
1:clientCommunication.CommunicateServerInOrderInsert(dataInput.CreateStudentI
nformation());break;
                case
2:clientCommunication.CommunicateServerInOrderDelete(dataInput.CreateStudentI
d());break;
                case
3:clientCommunication.CommunicateServerInOrderUpdate(dataInput.CreateStudentI
nformation(),dataInput.CreateStudentId());break;
                case
4:clientCommunication.CommunicateServerInOrderSearch(dataInput.CreateStudentI
d());break;
                case
5:clientCommunication.CommunicateServerInOrderCreateUser();break;

default:clientCommunication.CommunicateServerInOrderLogout();isFound=false;
            }
        }
        System.out.println("Good Bay");
    }
    public void ShowMenu(){
        System.out.println("----------------Welcome To DataBase Student------
----------");
        System.out.println();
        System.out.println("Please Choose One of Operations :");
        System.out.println();
        System.out.println("1- Insert Record.");
        System.out.println("2- Delete Record.");
        System.out.println("3- Update Record.");
        System.out.println("4- Search on Record.");
        System.out.println("5-Create New User");
        System.out.println("5- LogOut");
    }

}
```

In this class communicate with server to carry out certain operation as you show And there are simple interface in order client choose one of operations

```java
public class DataInput {
    private Scanner input;
    public DataInput(){
        input=new Scanner(System.in);
    }
    public StudentInfo CreateStudentInformation(){
        input=new Scanner(System.in);
        System.out.println("Please Enter The Following information : ");
        System.out.print("Student Id : ");
        int StudentId=input.nextInt();
        System.out.print("Student First Name : ");
        String StudentFirstName=input.next();
        System.out.print("Student Last Name : ");
        String StudentLastName=input.next();
        System.out.print("GPA : ");
        Double GPA=input.nextDouble();
        System.out.print("Number Of Course : ");
        int NumberOfCourse=input.nextInt();
        return new
StudentInfo(StudentId,StudentFirstName,StudentLastName,GPA,NumberOfCourse);
    }
    public int CreateStudentId(){
        System.out.print("Student Id : ");
        int StudentId=input.nextInt();
        return StudentId;
    }
    public int getOperationNumber(){
        int OperationNumber=input.nextInt();
        return OperationNumber;
    }
    public UserInfo CreateUserInformation(){
       UserInfo User=new UserInfo();
        input=new Scanner(System.in);
        System.out.println("Please Enter UserName :");
        String UserName=input.nextLine();
        System.out.println("Please Enter Password :");
        String Password=input.nextLine();
        System.out.println("Please Enter Id :");
        int Id=input.nextInt();
        User.setUserName(UserName);
        User.setPassword(Password);
        User.setUserId(Id);
        return User;
    }
}
```

This class is for data input operations.

```java
public class ClientConnection {
    private  Socket socket = null;
    private  ObjectOutputStream out = null;
    private  ObjectInputStream in = null;

    public  void OpenConnection(){
        try {
            socket = new Socket("localhost", 8000);
            System.out.println("Connected");
        }catch (IOException e){
            System.out.println(e);
        }
    }
    public void OpenResource(){
        try {
            out = new ObjectOutputStream(socket.getOutputStream());
            in = new ObjectInputStream(socket.getInputStream());
        }catch (Exception e){
            System.out.println(e);
        }
    }
    public void CloseResource(){
        try {
            System.out.println("Close Resource...");
            in.close();
            out.close();
        }catch (IOException e){
            System.out.println(e);
        }

    }
    public void CloseConnection(){
        try {
            System.out.println("Close Connection...");
            socket.close();
        }
        catch (Exception e){
            System.out.println(e);
        }
    }

    public  Socket getSocket() {
        return socket;
    }

    public  void setSocket(Socket socket) {
        this.socket = socket;
    }

    public  ObjectOutputStream getOut() {
        return out;
    }

    public  void setOut(ObjectOutputStream out) {
        this.out = out;
    }
```

```
    public  ObjectInputStream getIn() {
        return in;
    }

    public  void setIn(ObjectInputStream in) {
        this.in = in;
    }
```

This class is for opening connection with server and opening resources and using resource in classes client.

```
public class ClientCommunication {
    private static ClientCommunication clientCommunication;
    private ClientConnection connection;
    private DataInput dataInput;
    private ClientCommunication(){
        connection=new ClientConnection();
        dataInput=new DataInput();
    }
    public static ClientCommunication getClientCommunication(){
        if (clientCommunication==null)
            clientCommunication=new ClientCommunication();
        return clientCommunication;
    }
    public void init(){
        connection.OpenConnection();
        connection.OpenResource();
    }
     public Boolean CommunicateServerInOrderLogin(){
         UserInfo User=dataInput.CreateUserInformation();
         ObjectOutputStream out=connection.getOut();
         ObjectInputStream in=connection.getIn();
         Boolean isUser=false;
         try{
             out.writeObject("Log in");
             out.writeObject(User);
             out.flush();
              isUser=in.readBoolean();
         }catch (Exception e){
             System.out.println(e);
         }finally {
             return isUser;
         }
     }
```

```
    public void CommunicateServerInOrderInsert(StudentInfo studentInfo) {
        try {
            ObjectOutputStream out=connection.getOut();
            ObjectInputStream in=connection.getIn();
            out.writeObject("Insert");
            out.writeObject(studentInfo);
            out.flush();
            Boolean isInsert = in.readBoolean();
            if (isInsert) {
                System.out.println("Insert Record Successfully...");
            } else {
                System.out.println("Insert Record Not Successfully...");
            }
        } catch (Exception e) {
            System.out.println();
            System.out.println(e);
        }
    }

    public void CommunicateServerInOrderDelete(int StudentId) {

        try {
            ObjectOutputStream out=connection.getOut();
            ObjectInputStream in=connection.getIn();
            out.writeObject("Delete");
            out.writeInt(StudentId);
            out.flush();
            Boolean isDelete = in.readBoolean();
            if (isDelete) {
                System.out.println("Delete Record Successfully...");
            } else {
                System.out.println("Delete Record Not Successfully...");
            }
        } catch (Exception e) {
            System.out.println(e);
        }


    }

    public void CommunicateServerInOrderUpdate(StudentInfo studentInfo, int
StudentId) {
        try {
            ObjectOutputStream out=connection.getOut();
            ObjectInputStream in=connection.getIn();
            out.writeObject("Update");
            out.writeObject(studentInfo);
            out.writeInt(StudentId);
            out.flush();
            Boolean isUpdate = in.readBoolean();
            if (isUpdate) {
                System.out.println("Update Record Successfully...");
            } else {
                System.out.println("Update Record Not Successfully...");
            }
        } catch (Exception e) {
```

```
            System.out.println(e);
        }
    }

    public void CommunicateServerInOrderSearch(int StudentId) {
        try {
            ObjectOutputStream out=connection.getOut();
            ObjectInputStream in=connection.getIn();
            out.writeObject("Search");
            out.writeInt(StudentId);
            out.flush();
            StudentInfo studentInfo = (StudentInfo) in.readObject();
            //check  student  info to ensure if it is null
            System.out.println("Student Id : " + studentInfo.getStudentId());
            System.out.println("Student First Name : " +
studentInfo.getStudentFirstName());
            System.out.println("Student Last Name : " +
studentInfo.getStudentLastName());
            System.out.println("Student GPA : " + studentInfo.getGPA());
            System.out.println("Student Number Of Course : " +
studentInfo.getNumberOfCourses());
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public Boolean CommunicateServerInOrderCreateUser(){
        UserInfo User=dataInput.CreateUserInformation();
        ObjectOutputStream out=connection.getOut();
        ObjectInputStream in=connection.getIn();
        Boolean isUser=false;
        try{
            out.writeObject("CreateNewUser");
            out.writeObject(User);
            out.flush();
            isUser=in.readBoolean();
        }catch (Exception e){
            System.out.println(e);
        }finally {
            return isUser;
        }
    }

    public void CommunicateServerInOrderLogout() {
        try {
            ObjectOutputStream out=connection.getOut();
            out.writeObject("Log Out");
            out.flush();
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    public void close(){
        connection.CloseResource();
        connection.CloseConnection();
    }
```

```
}
```

This class is for communicate with server in order carry out operation that select from user.

```
package Info;

import java.io.Serializable;

public class StudentInfo implements Serializable {
    private int StudentId;
    private String StudentFirstName;
    private String StudentLastName;
    private double GPA;
    private int NumberOfCourses;
    public StudentInfo(){}
    public StudentInfo(int StudentId,String StudentFirstName,String
StudentLastName,double GPA,int NumberOfCourses){
        this.StudentFirstName=StudentFirstName;
        this.StudentId=StudentId;
        this.StudentLastName=StudentLastName;
        this.GPA=GPA;
        this.NumberOfCourses=NumberOfCourses;
    }

    public int getStudentId() {
        return StudentId;
    }

    public String getStudentFirstName() {
        return StudentFirstName;
    }

    public String getStudentLastName() {
        return StudentLastName;
    }

    public double getGPA() {
        return GPA;
```

```
    }

    public int getNumberOfCourses() {
        return NumberOfCourses;
    }

    public void setStudentId(int studentId) {
        StudentId = studentId;
    }

    public void setStudentFirstName(String studentFirstName) {
        StudentFirstName = studentFirstName;
    }

    public void setStudentLastName(String studentLastName) {
        StudentLastName = studentLastName;
    }

    public void setGPA(double GPA) {
        this.GPA = GPA;
    }

    public void setNumberOfCourses(int numberOfCourses) {
        NumberOfCourses = numberOfCourses;
    }
}
```

This class is for Student Information.

```
package Info;

import java.io.Serializable;

public class UserInfo implements Serializable {
    private int UserId;
    private String UserName;
    private String Password;

    public int getUserId() {
        return UserId;
    }

    public void setUserId(int userId) {
        UserId= userId;
    }

    public String getUserName() {
        return UserName;
```

```
    }

    public String getPassword() {
        return Password;
    }

    public void setUserName(String userName) {
        UserName = userName;
    }

    public void setPassword(String password) {
        Password = password;
    }
}
```

This class is for User Information.

```java
package ServerSide;

import Info.StudentInfo;
import Info.UserInfo;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class Server {
    public static void main(String[] args) {
        new Thread(new HandleAClient()).start();
    }

    private static class HandleAClient implements Runnable {
        private ServerConnection connection = new ServerConnection();
        private Operation operation = new Operation();

        @Override
        public void run() {
            connection.OpenConnection();
            SelectOperationAndCommunicationClient();

        }

        public Boolean CheckAndCommunicationClient() {
            Boolean isUser = false;
            ObjectInputStream in = connection.getIn();
            ObjectOutputStream out=connection.getOut();
            try {
                UserInfo userInfo = (UserInfo) in.readObject();
                isUser = operation.ValidationUser(userInfo);
               out.writeBoolean(isUser);
               out.flush();

            } catch (Exception e) {
                System.out.println(e);
            }
            System.out.println("CheckAndCommunicationClient    "+isUser);
            return isUser;
        }

        public void SelectOperationAndCommunicationClient() {
            Boolean ClientIsActive=true;
            connection.Accept();
            connection.OpenResource();
            ObjectInputStream in = connection.getIn();
            ObjectOutputStream out = connection.getOut();
            operation.init();
            while (ClientIsActive) {
                try {
                    String OperationType = null;
                    try {
                        OperationType = (String) in.readObject();
                    } catch (Exception e) {
                        System.out.println(e);
                    }
```

```
                    switch (OperationType) {
                        case "Log in":
                            try {
                                Boolean isUser;
                                 in = connection.getIn();
                                 out=connection.getOut();
                                UserInfo userInfo = (UserInfo)
in.readObject();
                                isUser = operation.ValidationUser(userInfo);
                                out.writeBoolean(isUser);
                                out.flush();
                            } catch (Exception e) {
                                System.out.println(e);
                            }break;

                        case "Delete":
                            try {
                                try {
                                    int studentId = in.readInt();
                                    Boolean isDelete =
operation.DeleteStudent(studentId);
                                    out.writeBoolean(isDelete);
                                    out.flush();
                                } catch (Exception e) {
                                    e.printStackTrace();
                                }
                            } catch (Exception e) {
                                System.out.println(e);
                            }  break;

                        case "Insert":
                            try {

                                StudentInfo studentInfo = (StudentInfo)
in.readObject();
                                Boolean isInsert =
operation.InsertStudent(studentInfo);
                                out.writeBoolean(isInsert);
                                out.flush();
                            } catch (ClassNotFoundException e) {
                                System.out.println(e);
                            }
                            break;

                        case "Update":
                            try {
                                StudentInfo studentInfo = (StudentInfo)
in.readObject();
                                int StudentId = in.readInt();
                                Boolean isUpdate =
operation.UpdateStudent(studentInfo, StudentId);
                                out.writeBoolean(isUpdate);
                                out.flush();
                            } catch (ClassNotFoundException e) {
                                e.printStackTrace();
                            }
                            break;
```

```
                        case "Search":
                            try {
                                int StudentId = in.readInt();
                                System.out.println(StudentId);
                                StudentInfo studentInfo =
operation.SearchStudent(StudentId);
                                out.writeObject(studentInfo);
                                out.flush();
                            } catch (Exception e) {
                                System.out.println(e);
                            }
                            break;
                       case "Log Out":
                            connection.CloseResource();
                            connection.CloseConnection();
                            ClientIsActive=false;
                            break;
                        case "CreateNewUser":
                            try {
                                UserInfo userInfo = (UserInfo)
in.readObject();
                                System.out.println(userInfo);
                                Boolean isCreateUser=
operation.CreateNewUser(userInfo);
                                out.writeBoolean(isCreateUser);
                                out.flush();
                            } catch (Exception e) {
                                System.out.println(e);
                            }
                            break;
                    }
                } catch (IOException e) {
                    System.out.println(e);
                }
            }


        }
    }
}
```

This is class Server multithreading contain class (HandleAClient )to carry out function run() that cary out (selectOperationandcommunicatewithclient)

In each time Client Select Operation

function selectOperationandcommunicatewithclient receive Operation name and comunicate with caching layer and Database found on disk through class (operation).

```java
package ServerSide;
import Info.StudentInfo;
import Info.UserInfo;

public class Operation {
    private CachingLayer cachingLayer;
    private DataBaseDAO dataBaseDAO;
    public void init(){
        cachingLayer=CachingLayer.getCachingLayer();
        dataBaseDAO=DataBaseDAO.getDataBaseDAO(10);
    }
    public Boolean InsertStudent(StudentInfo studentInfo){
        cachingLayer.InsertInMemoryDataBase(studentInfo);
        dataBaseDAO.InsertInDataBasePrimary(studentInfo);
        return true;

    }
    public Boolean DeleteStudent(int StudentId){
        cachingLayer.DeleteInMemoryDataBase(StudentId);
        dataBaseDAO.DeleteInDataBasePrimary(StudentId);
        return true;
    }
    public Boolean UpdateStudent(StudentInfo studentInfo, int StudentId){
        cachingLayer.UpdateInMemoryDataBase(studentInfo,StudentId);
        dataBaseDAO.UpdateInDataBasePrimary(studentInfo,StudentId);
        return true;
    }
    public StudentInfo SearchStudent(int StudentId){
        StudentInfo
studentInfo=cachingLayer.SearchInMemoryDataBase(StudentId);
        if(studentInfo==null){
            studentInfo=dataBaseDAO.SearchInDataBasePrimary(StudentId);
            cachingLayer.InsertInMemoryDataBase(studentInfo);
        }
        return studentInfo;
    }
    public Boolean CreateNewUser(UserInfo userInfo){
        if(ValidationUser(userInfo)){
            System.out.println("User is Found In Data Base");
            return false;
        }
      dataBaseDAO.CreateNewUserInDataBasePrimary(userInfo);
      cachingLayer.CreateNewUserInMemory(userInfo);
      return true;
    }
    public Boolean ValidationUser(UserInfo userInfo){
        Boolean isUser=cachingLayer.ValidationUserInMemoryDataBase(userInfo);
        if(isUser){
            System.out.println("Validation from memory");
            return true;
        }else {
            System.out.println("Validation from HDD");
            return dataBaseDAO.ValidationUserInDataBasePrimary(userInfo);
        }
    }
}
```

This class is Consider Layer Separate between database (In memory and disk)

In the functions writing into DB( Insert,Update) Communicate with Caching layer and Data base primary

In the function reading from DB(Search  and Validation)  initially Communicate with Caching layer if found data (cache Hit)no need communicate with  Data base primary

else if not found data (cache miss)need communicate with  Data base primary

```
package ServerSide;

import Info.StudentInfo;
import Info.UserInfo;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

public class DataBaseDAO {
    private static DataBaseDAO dataBaseDAO;
    private static ConnectionPool connectionPool;

    private DataBaseDAO(){}
    public static DataBaseDAO getDataBaseDAO(int NumberOfConnection){
        if(dataBaseDAO==null) {
            dataBaseDAO = new DataBaseDAO();
            connectionPool=new ConnectionPool();
            connectionPool.init(NumberOfConnection);
        }
        return dataBaseDAO;
    }
    public void InsertInDataBasePrimary(StudentInfo studentInfo){

        Connection connection = null;
           try {
               connection =connectionPool.getConnection();
               Statement statement=connection.createStatement();
               String SqlStatement="insert into Student " +
                       "values
("+"'"+studentInfo.getStudentId()+"'"+","+"'"+studentInfo.getStudentFirstName
```

```
()+"'"+","+

"'"+studentInfo.getStudentLastName()+"'"+","+"'"+studentInfo.getGPA()+"'"+","
+"'"+studentInfo.getNumberOfCourses()+"'"+")";
                statement.executeUpdate(SqlStatement);

            }catch (Exception e){
                System.out.println(e);
            }finally {
                try {
                    connectionPool.ReturnConnection(connection);
                }catch (NullPointerException e){
                    System.out.println(e);
                }
            }
    }
    public void DeleteInDataBasePrimary(int StudentId){

        Connection connection=null;
        try {
            connection=connectionPool.getConnection();
            Statement statement=connection.createStatement();
            String SqlStatement="delete from Student where
StudentId="+StudentId;
            statement.executeUpdate(SqlStatement);
        }catch (Exception e){
            System.out.println(e);
        }finally {
            try {
                connectionPool.ReturnConnection(connection);
            }catch (NullPointerException e){
                System.out.println(e);
            }
        }

    }
    public StudentInfo SearchInDataBasePrimary(int StudentId){
        System.out.println("SearchInDataBasePrimary");
        StudentInfo studentInfo=null;
        Connection connection=null;
        try {
            connection=connectionPool.getConnection();
            Statement statement=connection.createStatement();
            String SqlStatement="select * from Student where
StudentId="+StudentId;
            ResultSet resultSet=statement.executeQuery(SqlStatement);
            if (resultSet.next()){
                System.out.println("StudentId : "+resultSet.getString(1)+"
StudentFirstName : "+resultSet.getString(2)+"        StudentLastName :
"+resultSet.getString(3)+"       GPA : "+resultSet.getString(4)+"
NumberOfCourses : "+resultSet.getString(5));
                studentInfo=new StudentInfo();

studentInfo.setStudentId(Integer.parseInt(resultSet.getString(1)));
                studentInfo.setStudentFirstName(resultSet.getString(2));
                studentInfo.setStudentLastName(resultSet.getString(3));
```

```
studentInfo.setGPA(Double.parseDouble(resultSet.getString(4)));

studentInfo.setNumberOfCourses(Integer.parseInt(resultSet.getString(5)));
            }
        }catch (Exception e){
            System.out.println(e);
        }
        finally {
            try {
                connectionPool.ReturnConnection(connection);
            }catch (NullPointerException e){
                System.out.println(e);
            }
        }
        return studentInfo;

    }
    public void UpdateInDataBasePrimary(StudentInfo studentInfo, int
StudentId){
        DeleteInDataBasePrimary(StudentId);
        InsertInDataBasePrimary(studentInfo);
    }
    public Boolean ValidationUserInDataBasePrimary(UserInfo userInfo) {
        Connection connection=null;
        try {
            connection=connectionPool.getConnection();
            Statement statement=connection.createStatement();
            String SqlStatement="select * from User where
UserId="+userInfo.getUserId();
            ResultSet resultSet=statement.executeQuery(SqlStatement);
            if (resultSet.next()){
                System.out.println("UserId : "+resultSet.getString(1)+"
UserName : "+resultSet.getString(2)+"        Password :
"+resultSet.getString(3));
                if(userInfo.getUserName().equals(resultSet.getString(2))){

if(userInfo.getPassword().equals(resultSet.getString(3))){
                        return true;
                    }
                }
            }
        }catch (Exception e){
            System.out.println(e);
        }
        finally {
            try {
                connectionPool.ReturnConnection(connection);
            }catch (NullPointerException e){
                System.out.println(e);
            }
        }
        return false;
    }
    public void CreateNewUserInDataBasePrimary(UserInfo userInfo){
        Connection connection=null;
        try {
            connection =connectionPool.getConnection();
```

```
            Statement statement=connection.createStatement();
            String SqlStatement="insert into User " +
                    "values
("+"'"+userInfo.getUserId()+"'"+","+"'"+userInfo.getUserName()+"'"+","+
                    "'"+userInfo.getPassword()+"'"+")";
            statement.executeUpdate(SqlStatement);
        }
        catch (Exception e){
            System.out.println(e);
        }
        finally {
            try {
                connectionPool.ReturnConnection(connection);
            }catch (NullPointerException e){
                System.out.println(e);
            }
        }
    }
}
```

In this class is responsible communicate directly with data base found on disk

It is getting on connection from class pool connection.

I implement singleton in this class because I need one instance and there number Of connection in order support multithreading.

```
package ServerSide;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.ConcurrentSkipListMap;
import java.util.concurrent.TransferQueue;
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
```

```
import java.util.concurrent.locks.ReentrantLock;

public class ConnectionPool {
    ConcurrentSkipListMap<Integer,Connection>Pool=new
ConcurrentSkipListMap<>();
    private static Lock lock = new ReentrantLock();
    private static Condition notEmpty = lock.newCondition();
    private static Condition notFull = lock.newCondition();
    public void init(int number){
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Connected With Driver");
          for (int i=0;i<number;i++){
              Connection connection=
DriverManager.getConnection("jdbc:mysql://localhost:3306/javabook","Omar","12
34");
              Pool.put(i,connection);
          }
        }catch (Exception e){
            System.out.println("yyytuu");
            System.out.println(e);
        }
    }
    public void close(){

            for(int i=0;i<Pool.size();i++){
                try {
                   Pool.get(i).close();
                }catch (Exception e){
                    System.out.println(e);
                }
            }
    }

    public  Connection getConnection() {

        if(Pool.isEmpty()){
            try {
                notEmpty.await();
            }catch (Exception e){
                System.out.println(e);
            }
        }

        Connection connection= Pool.remove(Pool.size()-1);

        return connection;
    }
    public  void ReturnConnection(Connection connection){

       Pool.put(Pool.size(),connection);

        notFull.signalAll();

    }
```

```
}
```

In this class Implement Non-Blocking Concurrent Data Structure ConcurrentSkipListMap This construct allows us to create thread-safe logic in a lock-free way. This means no need synchronize.

```
package ServerSide;

import Info.StudentInfo;
import Info.UserInfo;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;
import redis.clients.jedis.Transaction;

import java.util.List;

public class CachingLayer {

    private static CachingLayer Instance;

    private static JedisPoolConfig poolConfig;
    private  static JedisPool jedisPool ;
    private CachingLayer(){}
    public static CachingLayer getCachingLayer(){
        if(Instance==null){
            Instance = new CachingLayer();
            poolConfig=ConfigurationCache.buildPoolConfig();
            jedisPool= new JedisPool(poolConfig, "localhost");
        }
        return Instance;
    }
    public void InsertInMemoryDataBase(StudentInfo studentInfo){
        try (Jedis jedis = jedisPool.getResource()) {
            Boolean
isFound=jedis.exists("STUDENT:"+studentInfo.getStudentId());
            if(isFound){
                System.out.println("Not allow duplicate key");
            }


jedis.lpush("STUDENT:"+studentInfo.getStudentId(),studentInfo.getStudentFirst
Name());

jedis.lpush("STUDENT:"+studentInfo.getStudentId(),studentInfo.getStudentLastN
ame());
```

```
            jedis.lpush("STUDENT:"+studentInfo.getStudentId(),
String.valueOf(studentInfo.getGPA()));

jedis.lpush("STUDENT:"+studentInfo.getStudentId(),String.valueOf(studentInfo.
getNumberOfCourses()));
        }




    }
    public void DeleteInMemoryDataBase(int StudentId){
        try (Jedis jedis = jedisPool.getResource()) {
            jedis.del("STUDENT:"+StudentId);
        }

    }
    public StudentInfo SearchInMemoryDataBase(int StudentId){
        try (Jedis jedis = jedisPool.getResource()) {
            System.out.println("SearchInMemoryDataBase");
            List<String> Info = jedis.lrange("STUDENT:" + StudentId, 0, 4);
            System.out.println(jedis.info("memory"));
            StudentInfo studentInfo = new StudentInfo();
            if (Info.isEmpty()) {
                System.out.println("The record not found in the data
base....");
                studentInfo = null;
            } else {
                System.out.println("StudentId  StudentFirstName
StudentLastName  GPA  NumberOfCourses");
                System.out.println(StudentId + "          " + Info.get(3) + "
" + Info.get(2) + "          " + Info.get(1) + "          " + Info.get(0) + "
");
                studentInfo.setStudentId(StudentId);
                studentInfo.setStudentFirstName(Info.get(3));
                studentInfo.setStudentLastName(Info.get(2));
                studentInfo.setGPA(Double.parseDouble(Info.get(1)));

studentInfo.setNumberOfCourses(Integer.parseInt(Info.get(0)));
            }
            return studentInfo;
        }
        }
    public void UpdateInMemoryDataBase(StudentInfo studentInfo, int
StudentId){
        DeleteInMemoryDataBase(StudentId);
        InsertInMemoryDataBase(studentInfo);

    }
    public Boolean ValidationUserInMemoryDataBase(UserInfo userInfo) {
        try (Jedis jedis = jedisPool.getResource()) {
            List<String> Info = jedis.lrange("USER:" + userInfo.getUserId(),
0, -1);
            System.out.println(Info);
            if (Info.isEmpty()) {
                return false;
            } else if (userInfo.getPassword().equals(Info.get(0))) {
```

```
                if (userInfo.getUserName().equals(Info.get(1))) {
                    return true;
                }
            }
            return false;
        }
    }
    public void CreateNewUserInMemory(UserInfo userInfo){
        try (Jedis jedis = jedisPool.getResource()) {
            jedis.lpush("USER:" + userInfo.getUserId(),
userInfo.getUserName());
            jedis.lpush("USER:" + userInfo.getUserId(),
userInfo.getPassword());
        }
    }

}
```

In this class communicate directly with memory

redis it is single-threaded so nothing will run until the Command has completed.

Also it is getting on connection from class ConfigurationCache

a pool that is thread safe and reliable as long as you return the resource to the pool when you are done with it.

```
package ServerSide;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerConnection {
    ServerSocket serverSocket=null;
```

```
    private  Socket socket = null;
    private ObjectOutputStream out = null;
    private ObjectInputStream in = null;
    public void OpenConnection(){
        try {
            serverSocket=new ServerSocket(8000);
            System.out.println("Connected");
        }catch (IOException e){
            System.out.println(e);
        }
    }
    public void Accept(){
        try {
            socket=serverSocket.accept();
        }catch (Exception e){
            System.out.println(e);
        }

    }
    public void OpenResource(){
        try {
            out = new ObjectOutputStream(socket.getOutputStream());
            in = new ObjectInputStream(socket.getInputStream());
        }catch (Exception e){
            System.out.println(e);
        }
    }
    public void CloseResource(){
        try {
            System.out.println("Close Resource...");
            in.close();
            out.close();
        }catch (IOException e){
            System.out.println("llllll");
            System.out.println(e);
        }

    }
    public void CloseConnection(){
        try {
            System.out.println("Close Connection...");
           socket.close();
           serverSocket.close();
        }
        catch (Exception e){
            System.out.println(e);
        }
    }
    public ServerSocket getServerSocket() {
        return serverSocket;
    }

    public void setServerSocket(ServerSocket serverSocket) {
        this.serverSocket = serverSocket;
    }

    public Socket getSocket() {
```

```
        return socket;
    }

    public void setSocket(Socket socket) {
        this.socket = socket;
    }

    public ObjectOutputStream getOut() {
        return out;
    }

    public void setOut(ObjectOutputStream out) {
        this.out = out;
    }

    public ObjectInputStream getIn() {
        return in;
    }

    public void setIn(ObjectInputStream in) {
        this.in = in;
    }

}
```

This class is for opening connection with client and opening resources and using resource in classes Server.