Systems Analysis and Design

Instructor : Huang, Chuen-Min

Teamwork ver.2

Group 1

ID	Name
B10423002	Leon
B10423003	Kurumi
B10423009	Jerry
B10423015	Justin
B10423032	Kevin
B10423041	Dan
B10423045	Rong
W10423301	Ben
A10523050	Ian

Date 2017/12/25

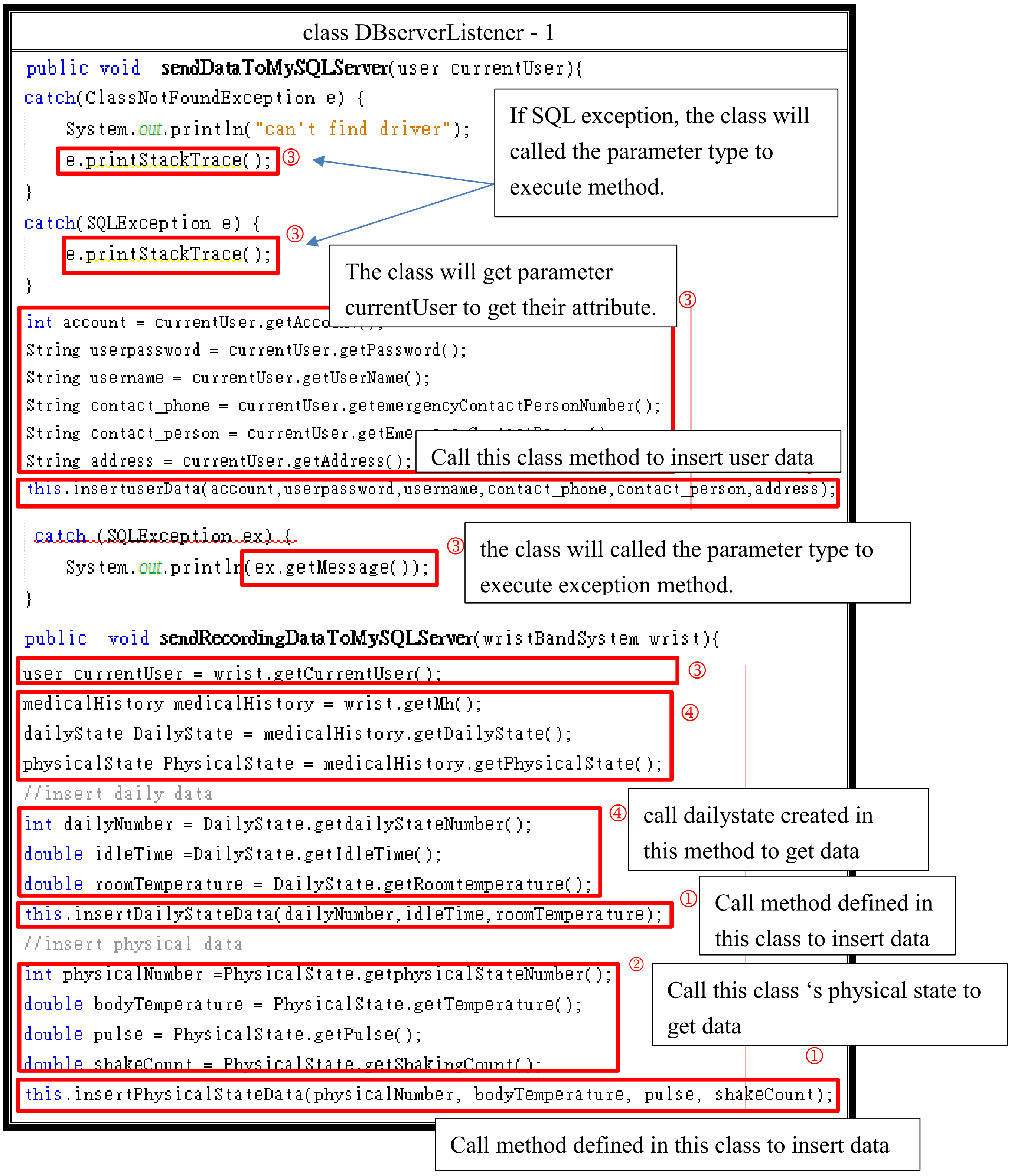
Content

Content

1) Please explain the <u>Law of Demeter (LoD)</u> by using any piece of your project.

Law of Demeter (LoD)	symbol
(1) to itself (O itself)	①
(2) to objects contained in attributes of itself or a superclass (Any objects created/instantiated within M)	②
(3) to an object that is passed as a parameter to the method (M’s parameters)	③
(4) to an object that is created by the method (O's direct component objects)	④

class DBserverListener - 1



```
public void  sendDataToMySQLServer(user currentUser){
catch(ClassNotFoundException e) {
    System.out.println("can't find driver");
    e.printStackTrace(); ③
}
catch(SQLException e) { ③
    e.printStackTrace();
}
int account = currentUser.getAcco
String userpassword = currentUser.getPassword();
String username = currentUser.getUserName();
String contact_phone = currentUser.getemergencyContactPersonNumber();
String contact_person = currentUser.getEme
String address = currentUser.getAddress();
this.insertuserData(account,userpassword,username,contact_phone,contact_person,address);

 catch (SQLException ex) { ③
    System.out.println(ex.getMessage());
}

public  void sendRecordingDataToMySQLServer(wristBandSystem wrist){
user currentUser = wrist.getCurrentUser(); ③
medicalHistory medicalHistory = wrist.getMh(); ④
dailyState DailyState = medicalHistory.getDailyState();
physicalState PhysicalState = medicalHistory.getPhysicalState();
//insert daily data
int dailyNumber = DailyState.getdailyStateNumber(); ④
double idleTime =DailyState.getIdleTime();
double roomTemperature = DailyState.getRoomtemperature();
this.insertDailyStateData(dailyNumber,idleTime,roomTemperature); ①
//insert physical data
int physicalNumber =PhysicalState.getphysicalStateNumber(); ②
double bodyTemperature = PhysicalState.getTemperature();
double pulse = PhysicalState.getPulse();
double shakeCount = PhysicalState.getShakingCount();
this.insertPhysicalStateData(physicalNumber, bodyTemperature, pulse, shakeCount);
```

If SQL exception, the class will called the parameter type to execute method.

The class will get parameter currentUser to get their attribute. ③

Call this class method to insert user data

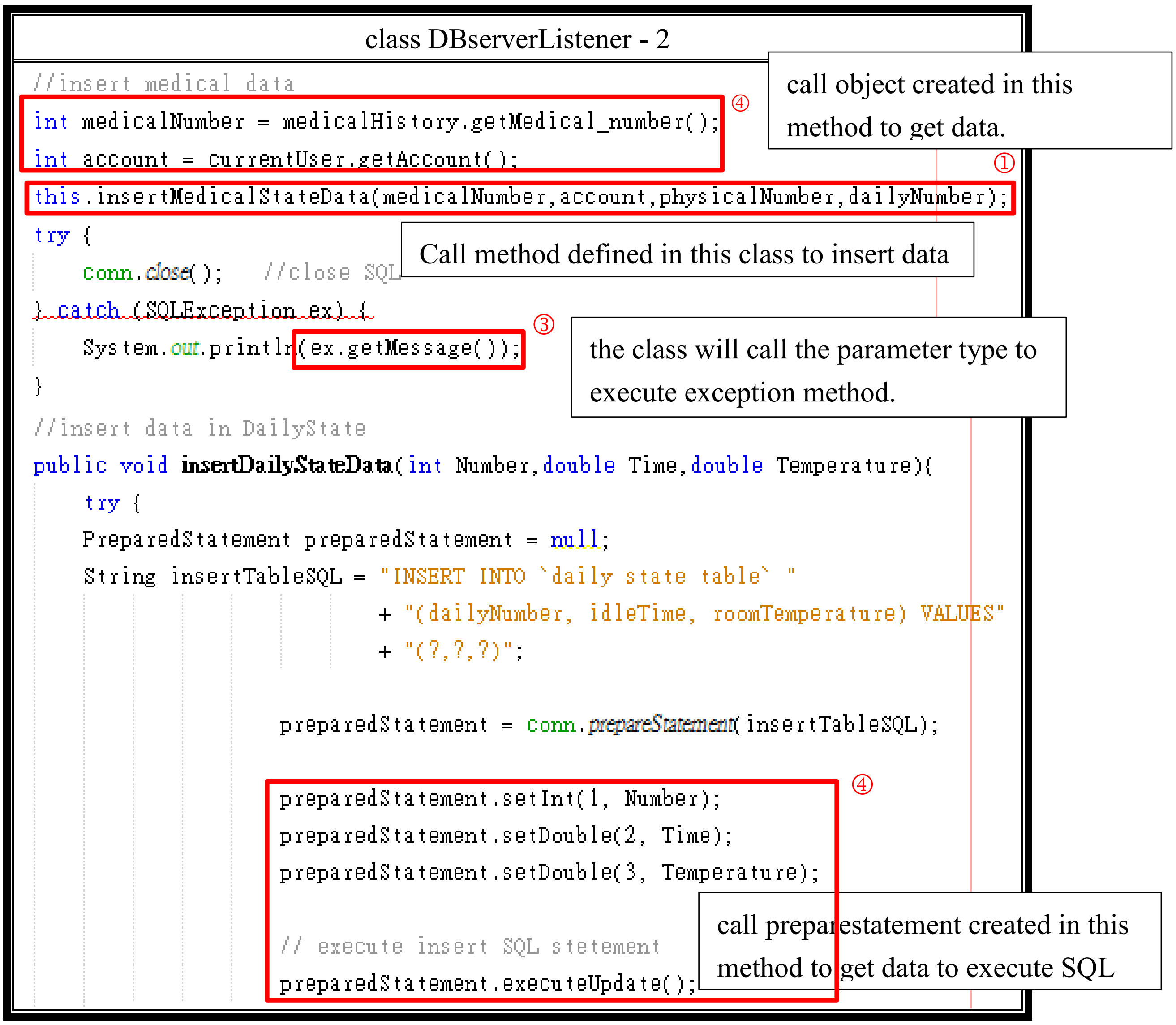
the class will called the parameter type to execute exception method.

call dailystate created in this method to get data

Call method defined in this class to insert data

Call this class ‘s physical state to get data ①

Call method defined in this class to insert data



class DBserverListener - 2

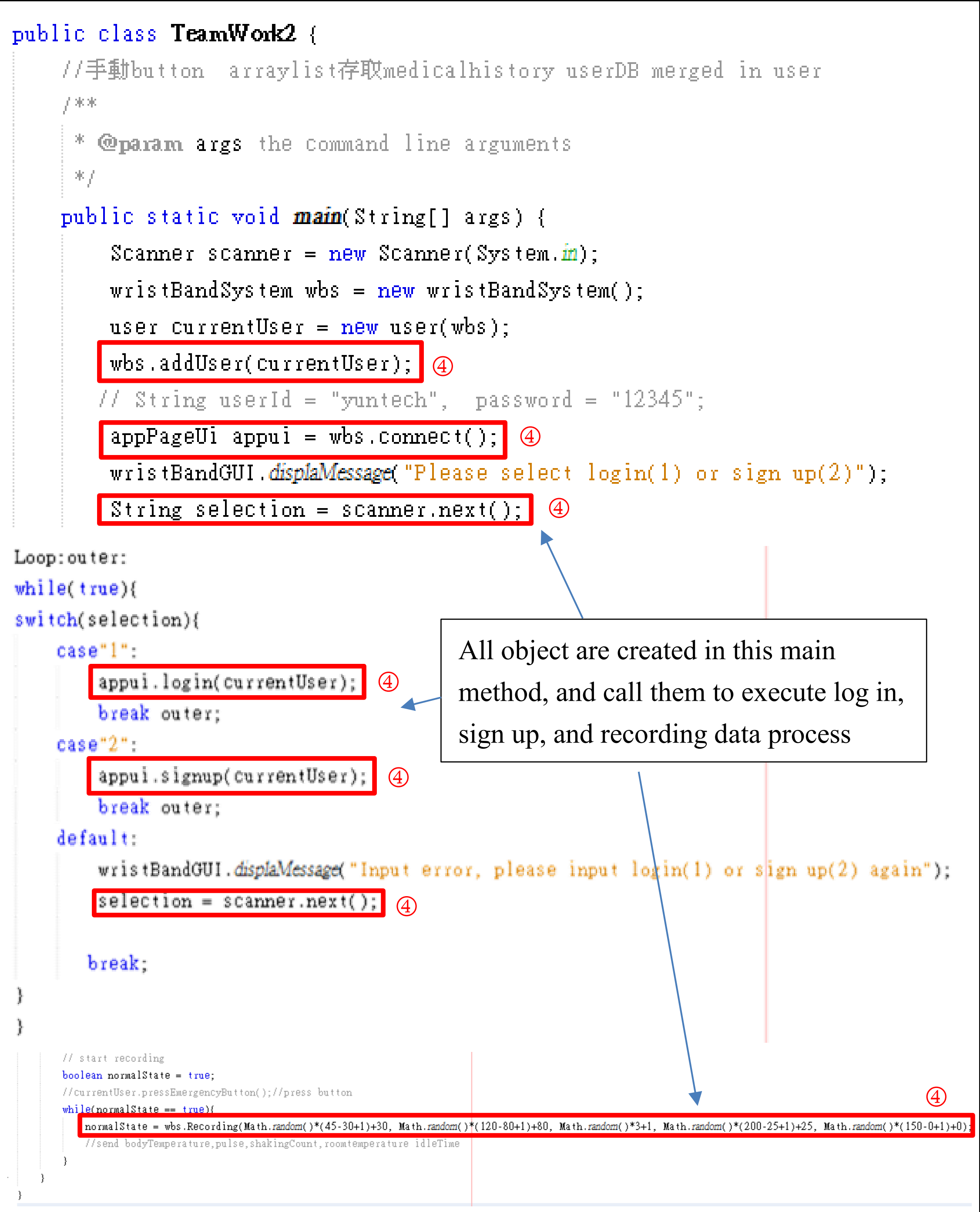
```
//insert medical data
int medicalNumber = medicalHistory.getMedical_number();
int account = currentUser.getAccount();
this.insertMedicalStateData(medicalNumber,account,physicalNumber,dailyNumber);
try {
    conn.close();   //close SQL
} catch (SQLException ex) {
    System.out.println(ex.getMessage());
}
//insert data in DailyState
public void insertDailyStateData(int Number,double Time,double Temperature){
    try {
    PreparedStatement preparedStatement = null;
    String insertTableSQL = "INSERT INTO `daily state table` "
                          + "(dailyNumber, idleTime, roomTemperature) VALUES"
                          + "(?,?,?)";

                preparedStatement = conn.prepareStatement(insertTableSQL);

                preparedStatement.setInt(1, Number);
                preparedStatement.setDouble(2, Time);
                preparedStatement.setDouble(3, Temperature);

                // execute insert SQL stetement
                preparedStatement.executeUpdate();
```

call object created in this method to get data.

Call method defined in this class to insert data

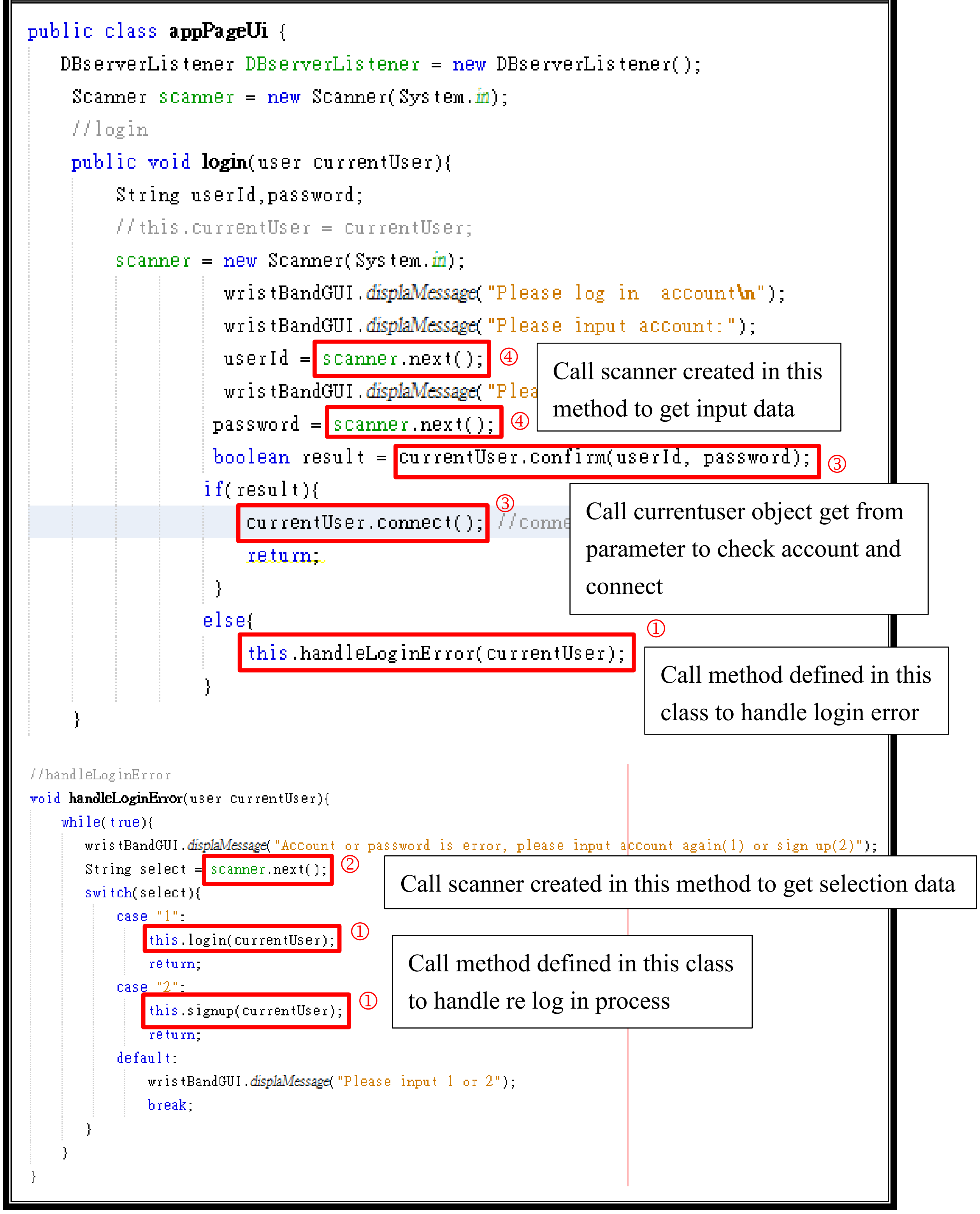
the class will call the parameter type to execute exception method.

call preparestatement created in this method to get data to execute SQL class TeamWork2



```java
public class TeamWork2 {
    //手動button  arraylist存取medicalhistory userDB merged in user
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        wristBandSystem wbs = new wristBandSystem();
        user currentUser = new user(wbs);
        wbs.addUser(currentUser);
       // String userId = "yuntech",  password = "12345";
        appPageUi appui = wbs.connect();
        wristBandGUI.displaMessage("Please select login(1) or sign up(2)");
        String selection = scanner.next();

Loop:outer:
while(true){
switch(selection){
    case"1":
        appui.login(currentUser);
        break outer;
    case"2":
        appui.signup(currentUser);
        break outer;
    default:
        wristBandGUI.displaMessage("Input error, please input login(1) or sign up(2) again");
        selection = scanner.next();

        break;
}
}
        // start recording
        boolean normalState = true;
        //currentUser.pressEmergencyButton();//press button
        while(normalState == true){
            normalState = wbs.Recording(Math.random()*(45-30+1)+30, Math.random()*(120-80+1)+80, Math.random()*3+1, Math.random()*(200-25+1)+25, Math.random()*(150-0+1)+0);
            //send bodyTemperature,pulse,shakingCount,roomtemperature idleTime
        }
    }
}
```

All object are created in this main method, and call them to execute log in, sign up, and recording data process class appPageUi - 1



```
public class appPageUi {
   DBserverListener DBserverListener = new DBserverListener();
    Scanner scanner = new Scanner(System.in);
    //login
    public void login(user currentUser){
        String userId,password;
        //this.currentUser = currentUser;
        scanner = new Scanner(System.in);
                  wristBandGUI.displaMessage("Please log in  account\n");
                  wristBandGUI.displaMessage("Please input account:");
                  userId = scanner.next(); ④
                  wristBandGUI.displaMessage("Plea
                 password = scanner.next(); ④
                 boolean result = currentUser.confirm(userId, password); ③
                if(result){                          ③
                    currentUser.connect(); //conne
                    return;
                 }
                else{
                    this.handleLoginError(currentUser); ①
                }
    }

//handleLoginError
void handleLoginError(user currentUser){
    while(true){
        wristBandGUI.displaMessage("Account or password is error, please input account again(1) or sign up(2)");
        String select = scanner.next(); ②
        switch(select){
            case "1":
                this.login(currentUser); ①
                return;
            case "2":
                this.signup(currentUser); ①
                return;
            default:
                wristBandGUI.displaMessage("Please input 1 or 2");
                break;
        }
    }
}
```

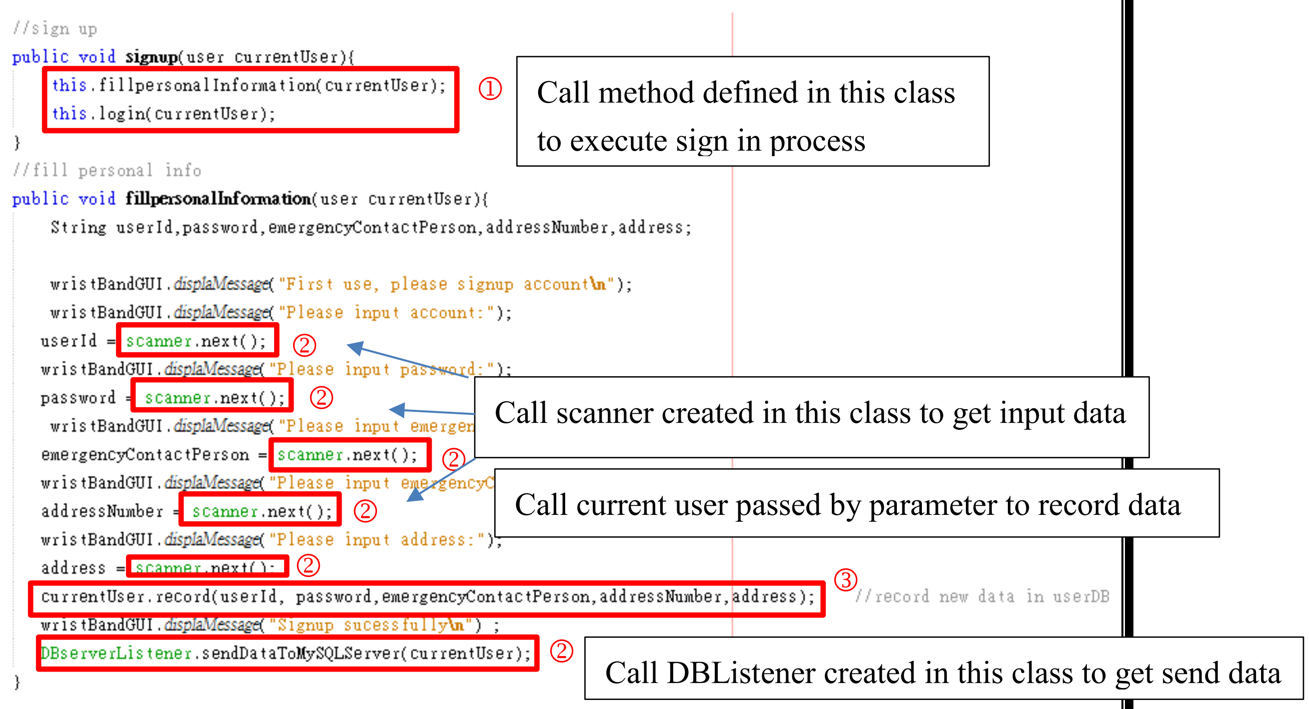
Call scanner created in this method to get input data

Call currentuser object get from parameter to check account and connect

Call method defined in this class to handle login error

Call scanner created in this method to get selection data

Call method defined in this class to handle re log in process class appPageUi - 2



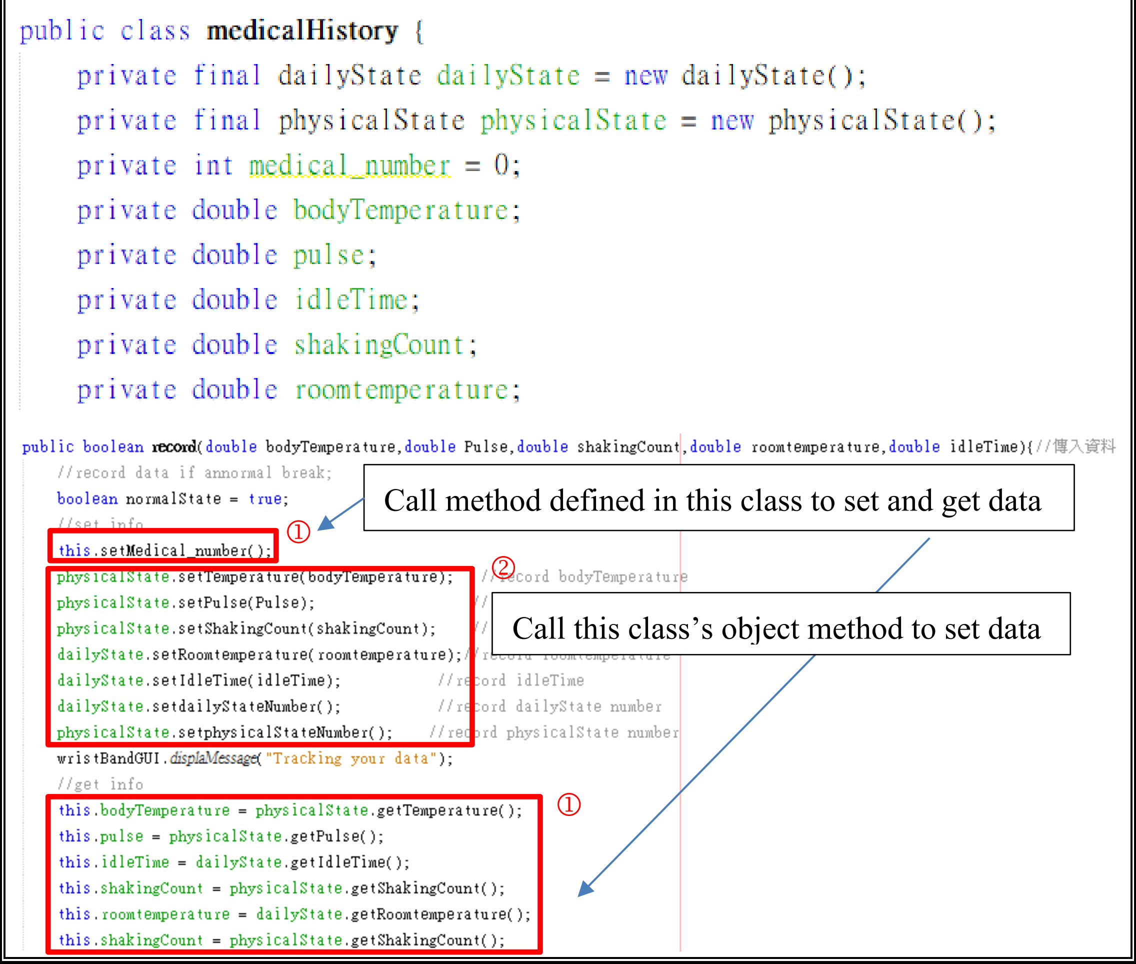
```
//sign up
public void signup(user currentUser){
    this.fillpersonalInformation(currentUser);   ①
    this.login(currentUser);
}
//fill personal info
public void fillpersonalInformation(user currentUser){
    String userId,password,emergencyContactPerson,addressNumber,address;

    wristBandGUI.displaMessage("First use, please signup account\n");
    wristBandGUI.displaMessage("Please input account:");
   userId = scanner.next();   ②
   wristBandGUI.displaMessage("Please input password:");
   password = scanner.next();   ②
    wristBandGUI.displaMessage("Please input emergen
   emergencyContactPerson = scanner.next();   ②
   wristBandGUI.displaMessage("Please input emergencyC
   addressNumber = scanner.next();   ②
   wristBandGUI.displaMessage("Please input address:");
   address = scanner.next();   ②
   currentUser.record(userId, password,emergencyContactPerson,addressNumber,address);   ③ //record new data in userDB
   wristBandGUI.displaMessage("Signup sucessfully\n") ;
   DBserverListener.sendDataToMySQLServer(currentUser);   ②
}
```

Call method defined in this class to execute sign in process

Call scanner created in this class to get input data

Call current user passed by parameter to record data

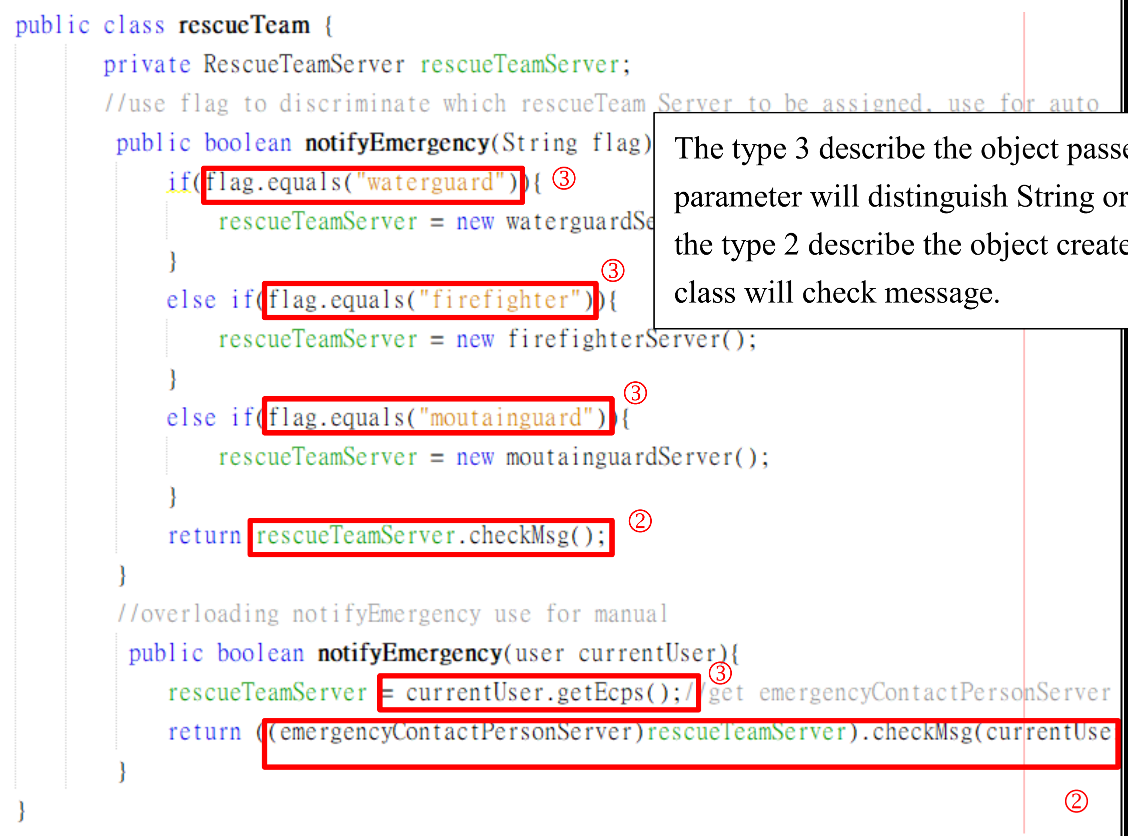
Call DBListener created in this class to get send data class medicalHistory



```java
public class medicalHistory {
    private final dailyState dailyState = new dailyState();
    private final physicalState physicalState = new physicalState();
    private int medical_number = 0;
    private double bodyTemperature;
    private double pulse;
    private double idleTime;
    private double shakingCount;
    private double roomtemperature;

public boolean record(double bodyTemperature,double Pulse,double shakingCount,double roomtemperature,double idleTime){//傳入資料
    //record data if annormal break;
    boolean normalState = true;
    //set info
    this.setMedical_number();
    physicalState.setTemperature(bodyTemperature);  //record bodyTemperature
    physicalState.setPulse(Pulse);                  //
    physicalState.setShakingCount(shakingCount);    //
    dailyState.setRoomtemperature(roomtemperature);//record roomtemperature
    dailyState.setIdleTime(idleTime);           //record idleTime
    dailyState.setdailyStateNumber();           //record dailyState number
    physicalState.setphysicalStateNumber();    //record physicalState number
    wristBandGUI.displaMessage("Tracking your data");
    //get info
    this.bodyTemperature = physicalState.getTemperature();
    this.pulse = physicalState.getPulse();
    this.idleTime = dailyState.getIdleTime();
    this.shakingCount = physicalState.getShakingCount();
    this.roomtemperature = dailyState.getRoomtemperature();
    this.shakingCount = physicalState.getShakingCount();
```

①
Call method defined in this class to set and get data
②
Call this class’s object method to set data
①

class rescueTeam - 1



```
public class rescueTeam {
        private RescueTeamServer rescueTeamServer;
        //use flag to discriminate which rescueTeam Server to be assigned, use for auto
         public boolean notifyEmergency(String flag)
             if(flag.equals("waterguard")){ ③
                 rescueTeamServer = new waterguardSe
             }                                        ③
             else if(flag.equals("firefighter")){
                 rescueTeamServer = new firefighterServer();
             }                                          ③
             else if(flag.equals("moutainguard")){
                 rescueTeamServer = new moutainguardServer();
             }
             return rescueTeamServer.checkMsg(); ②
         }
         //overloading notifyEmergency use for manual
          public boolean notifyEmergency(user currentUser){
             rescueTeamServer = currentUser.getEcps();//③get emergencyContactPersonServer
             return ((emergencyContactPersonServer)rescueTeamServer).checkMsg(currentUse
         }                                                                            ②
}
```

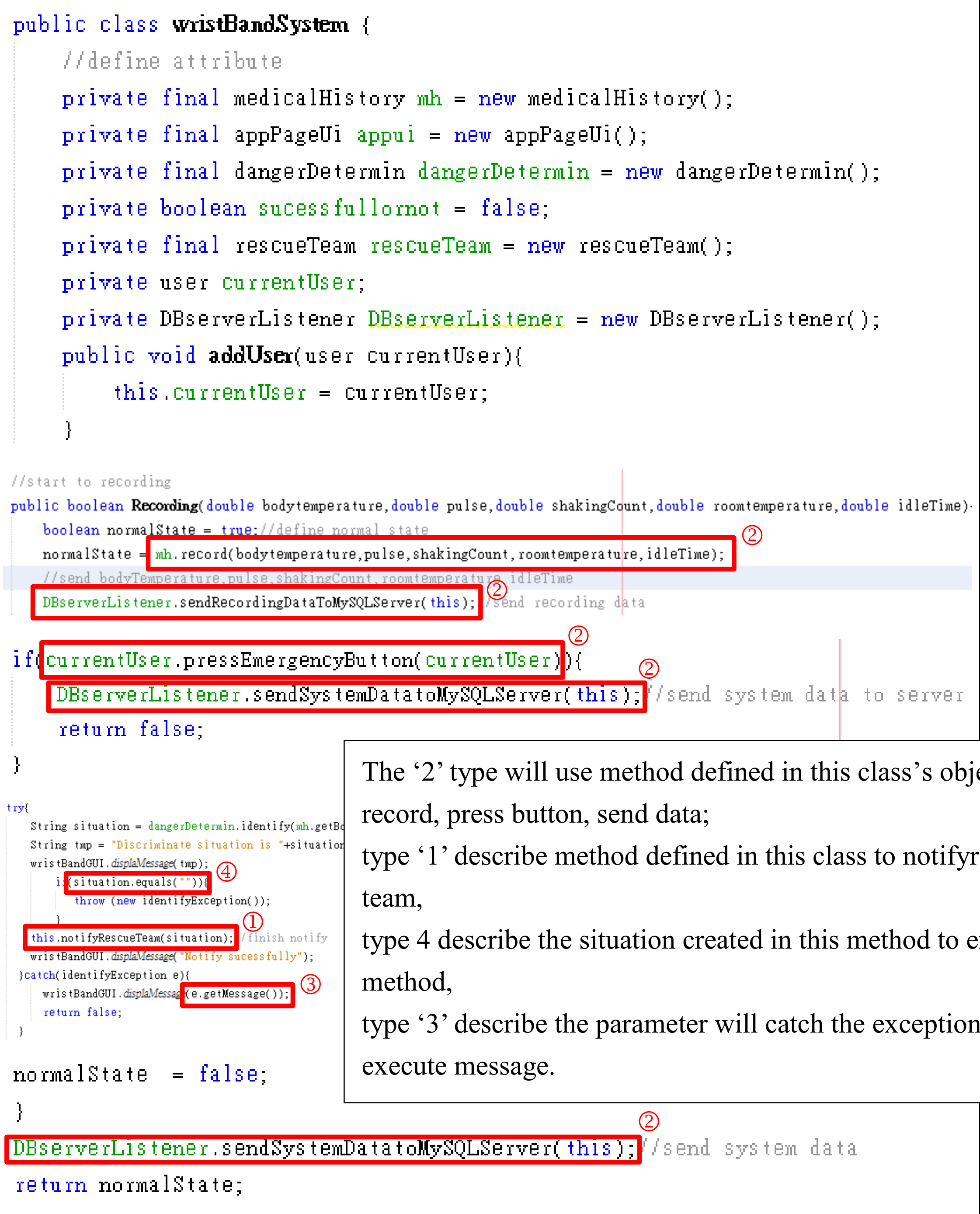
The type 3 describe the object passed by the parameter will distinguish String or getting, the type 2 describe the object created in this class will check message.

class rescueTeam - 2

```
public class user {
    private wristBandSystem wbs ;
    public  String account;//primary key
    private String userName = "Kevin";
    private String password  = "12345";
    private String addressNumber = "";
    private String address = "Dream Mall";
    private String emergencyContactPerson = "default";

private emergencyContactPersonServer ecps;
public user(wristBandSystem wbs){
     this.wbs = wbs;
      ecps = emergencyContactPersonServer.getemergencyContactPersonServer();
}

//press EmergencyButton over 5 times
public boolean pressEmergencyButton(user currentUser){
    double count = Math.random()*(5-0+1)+1; //define count
    //if count >= 5 active notify function
    if(count >= 5){
     wristBandGUI.displaMessage("You press emergency button
     wbs.notifyRescueTeam(currentUser); ②
     return true;
    }
    else{
        return false;
    }
}

public void setEmergencyContactPerson(String emergencyContactPerson) {
     this.emergencyContactPerson = emergencyContactPerson;
     ecps.setName(emergencyContactPerson); ②
}
```

Both type 2 describe the object created in this class will notify or setting.

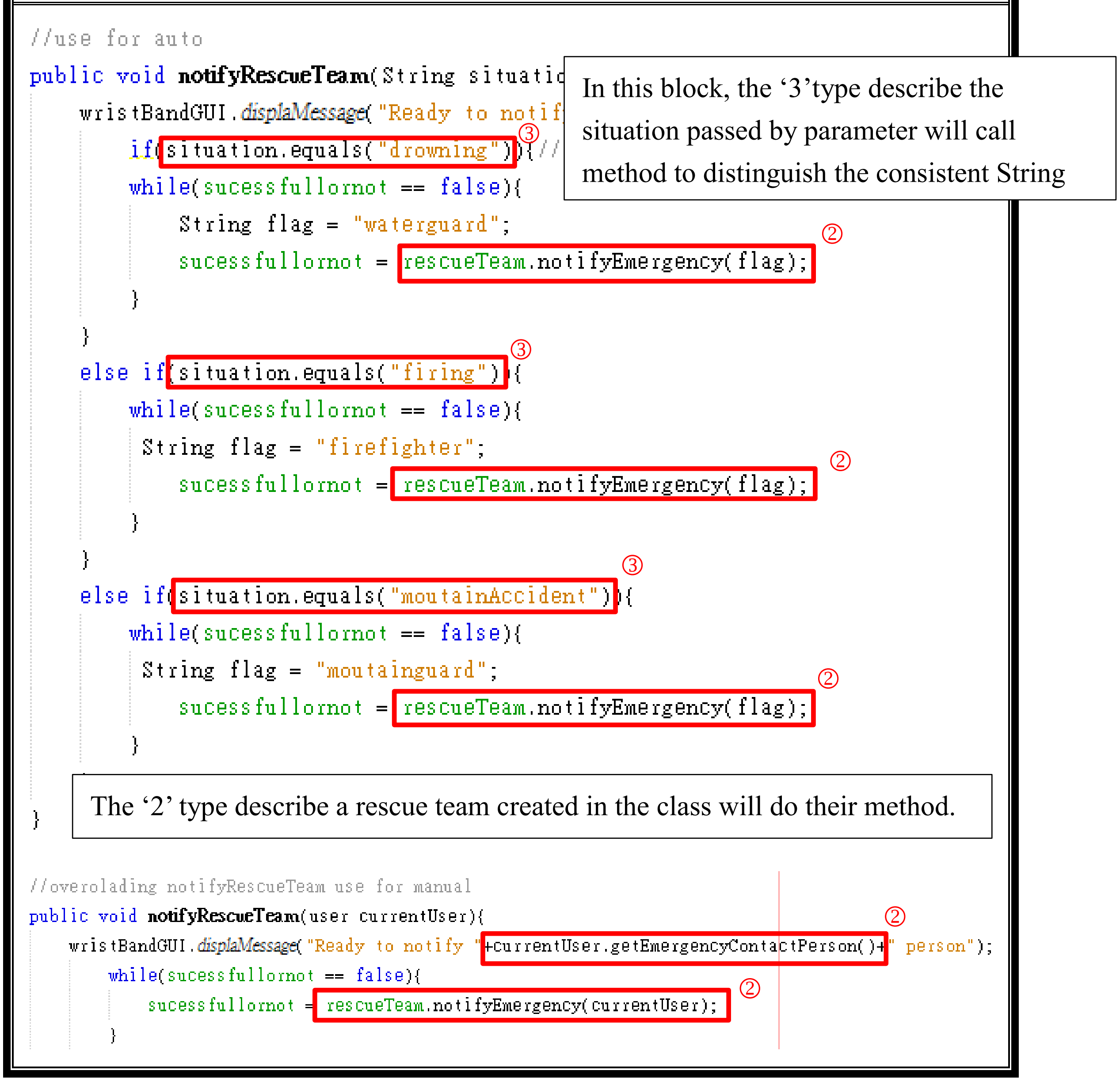
class wristBandSystem - 1



```
public class wristBandSystem {
    //define attribute
    private final medicalHistory mh = new medicalHistory();
    private final appPageUi appui = new appPageUi();
    private final dangerDetermin dangerDetermin = new dangerDetermin();
    private boolean sucessfullornot = false;
    private final rescueTeam rescueTeam = new rescueTeam();
    private user currentUser;
    private DBserverListener DBserverListener = new DBserverListener();
    public void addUser(user currentUser){
        this.currentUser = currentUser;
    }

//start to recording
public boolean Recording(double bodytemperature,double pulse,double shakingCount,double roomtemperature,double idleTime)
    boolean normalState = true;//define normal state
    normalState = mh.record(bodytemperature,pulse,shakingCount,roomtemperature,idleTime); ②
    //send bodyTemperature,pulse,shakingCount,roomtemperature,idleTime
    DBserverListener.sendRecordingDataToMySQLServer(this); ② //send recording data

if(currentUser.pressEmergencyButton(currentUser)){ ②
    DBserverListener.sendSystemDatatoMySQLServer(this); ② //send system data to server
    return false;
}

try{
    String situation = dangerDetermin.identify(mh.getBo
    String tmp = "Discriminate situation is "+situation
    wristBandGUI.displaMessage(tmp); ④
        if(situation.equals("")){
            throw (new identifyException());
        }
    this.notifyRescueTeam(situation); ① //finish notify
    wristBandGUI.displaMessage("Notify sucessfully");
}catch(identifyException e){
    wristBandGUI.displaMessage(e.getMessage()); ③
    return false;
}

normalState  = false;
}
DBserverListener.sendSystemDatatoMySQLServer(this); ② //send system data
return normalState;
```

The ‘2’ type will use method defined in this class’s object to record, press button, send data;
type ‘1’ describe method defined in this class to notifyrescue team,
type 4 describe the situation created in this method to execute method,
type ‘3’ describe the parameter will catch the exception to execute message.

class wristBandSystem - 2

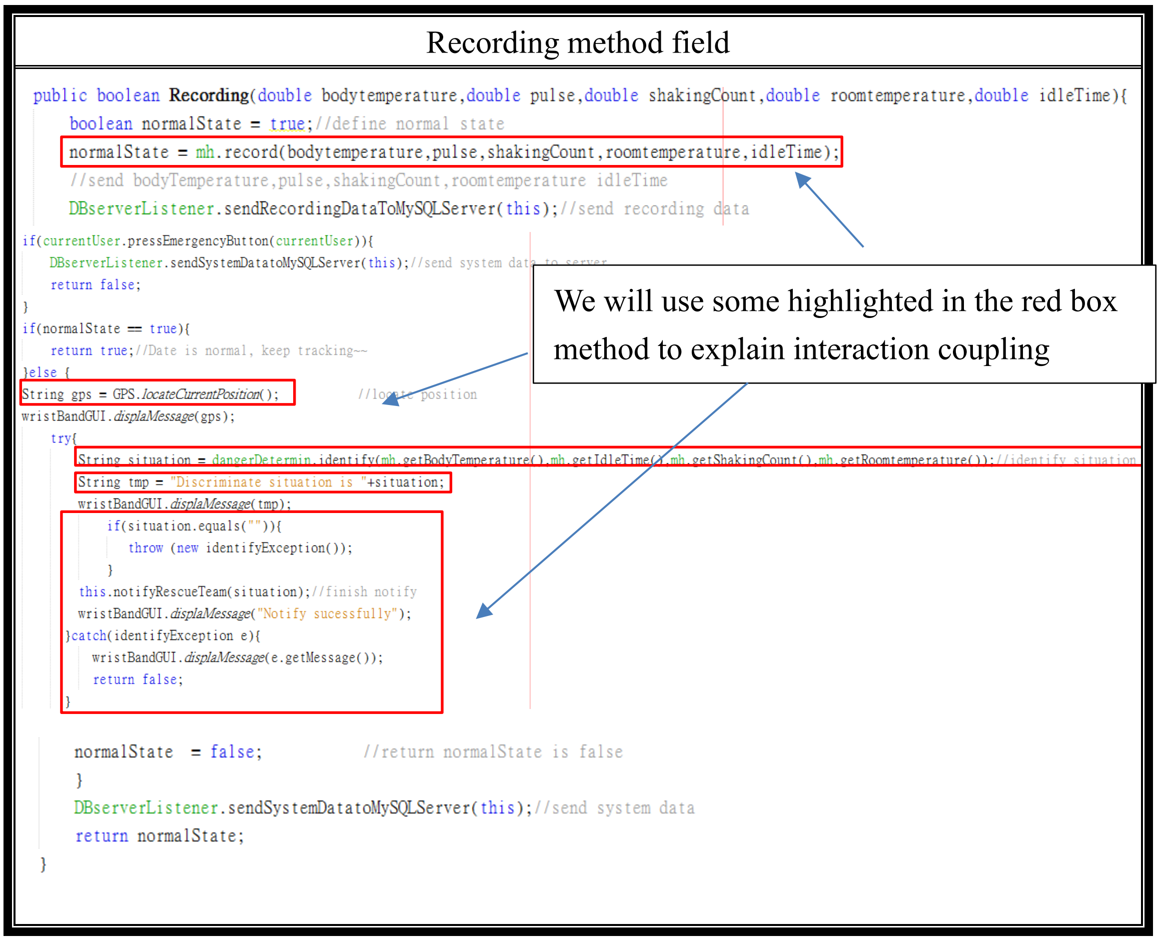


```
//use for auto
public void notifyRescueTeam(String situatic
    wristBandGUI.displaMessage("Ready to notif
        if(situation.equals("drowning")){//
        while(sucessfullornot == false){
            String flag = "waterguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("firing")){
        while(sucessfullornot == false){
         String flag = "firefighter";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("moutainAccident")){
        while(sucessfullornot == false){
         String flag = "moutainguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }

}

//overolading notifyRescueTeam use for manual
public void notifyRescueTeam(user currentUser){
    wristBandGUI.displaMessage("Ready to notify "+currentUser.getEmergencyContactPerson()+" person");
        while(sucessfullornot == false){
            sucessfullornot = rescueTeam.notifyEmergency(currentUser);
        }
```

In this block, the ‘3’type describe the situation passed by parameter will call method to distinguish the consistent String

The ‘2’ type describe a rescue team created in the class will do their method.

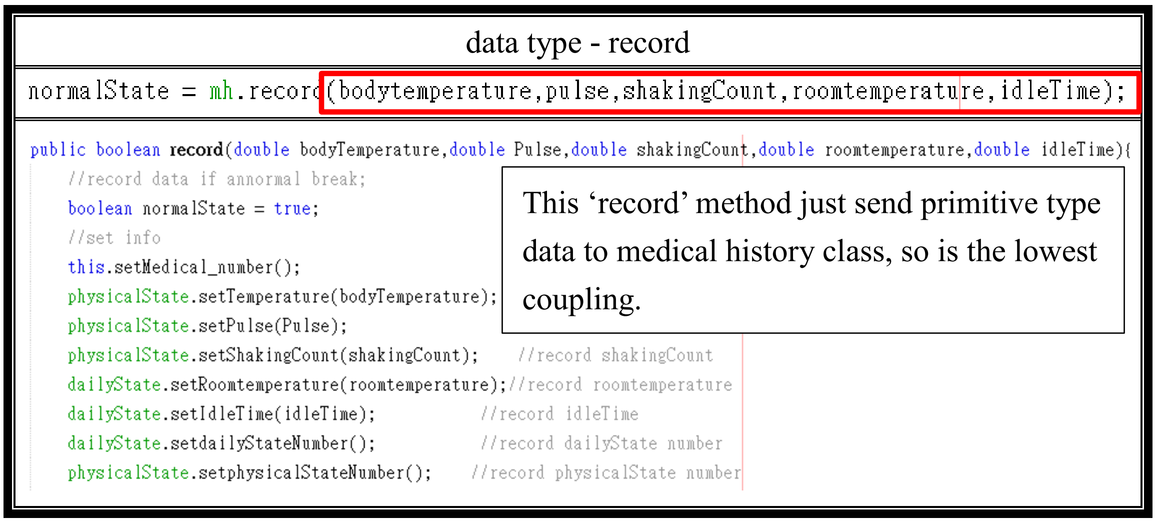
2) Here are six (or seven) types of interaction coupling, each falling on different on different parts of a good-to-bad continuum. Choose three pieces of your project to describe what types of the coupling they belong to.



Recording method field

```
public boolean Recording(double bodytemperature,double pulse,double shakingCount,double roomtemperature,double idleTime){
    boolean normalState = true;//define normal state
    normalState = mh.record(bodytemperature,pulse,shakingCount,roomtemperature,idleTime);
    //send bodyTemperature,pulse,shakingCount,roomtemperature idleTime
    DBserverListener.sendRecordingDataToMySQLServer(this);//send recording data
if(currentUser.pressEmergencyButton(currentUser)){
    DBserverListener.sendSystemDatatoMySQLServer(this);//send system data to server
    return false;
}
if(normalState == true){
    return true;//Date is normal, keep tracking~~
}else {
String gps = GPS.locateCurrentPosition();          //locate position
wristBandGUI.displaMessage(gps);
    try{
        String situation = dangerDetermin.identify(mh.getBodyTemperature(),mh.getIdleTime(),mh.getShakingCount(),mh.getRoomtemperature());//identify situation
        String tmp = "Discriminate situation is "+situation;
        wristBandGUI.displaMessage(tmp);
            if(situation.equals("")){
                throw (new identifyException());
            }
        this.notifyRescueTeam(situation);//finish notify
        wristBandGUI.displaMessage("Notify sucessfully");
    }catch(identifyException e){
        wristBandGUI.displaMessage(e.getMessage());
        return false;
    }

    normalState  = false;          //return normalState is false
    }
    DBserverListener.sendSystemDatatoMySQLServer(this);//send system data
    return normalState;
}
```

We will use some highlighted in the red box method to explain interaction coupling



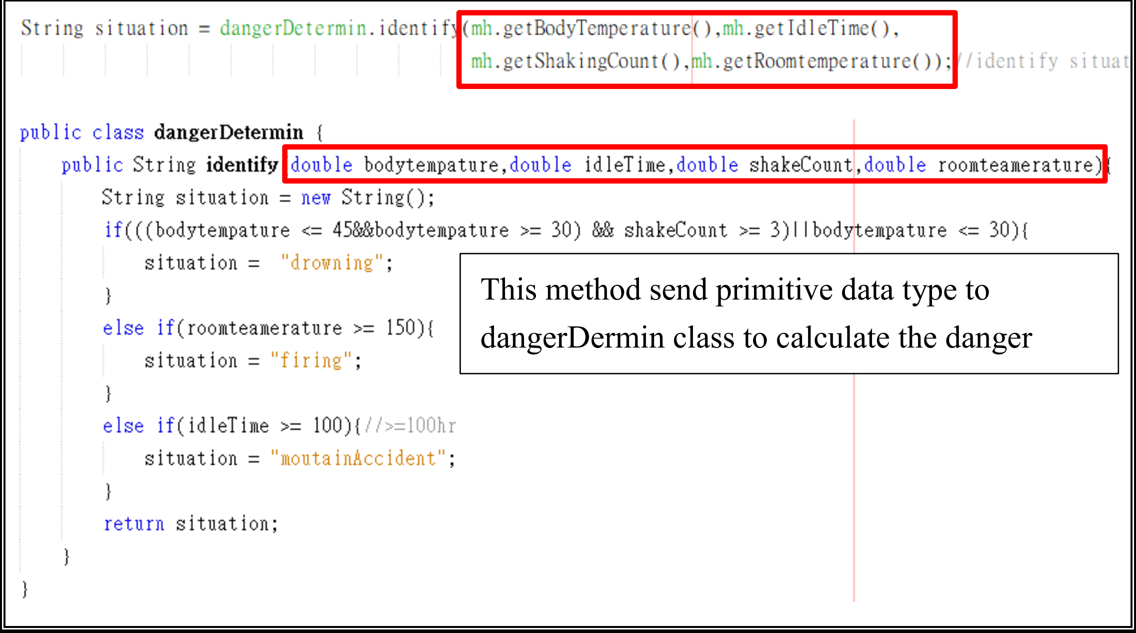
data type - record

```
normalState = mh.record(bodytemperature,pulse,shakingCount,roomtemperature,idleTime);
```

```
public boolean record(double bodyTemperature,double Pulse,double shakingCount,double roomtemperature,double idleTime){
    //record data if annormal break;
    boolean normalState = true;
    //set info
    this.setMedical_number();
    physicalState.setTemperature(bodyTemperature);
    physicalState.setPulse(Pulse);
    physicalState.setShakingCount(shakingCount);    //record shakingCount
    dailyState.setRoomtemperature(roomtemperature);//record roomtemperature
    dailyState.setIdleTime(idleTime);          //record idleTime
    dailyState.setdailyStateNumber();          //record dailyState number
    physicalState.setphysicalStateNumber();    //record physicalState number
```

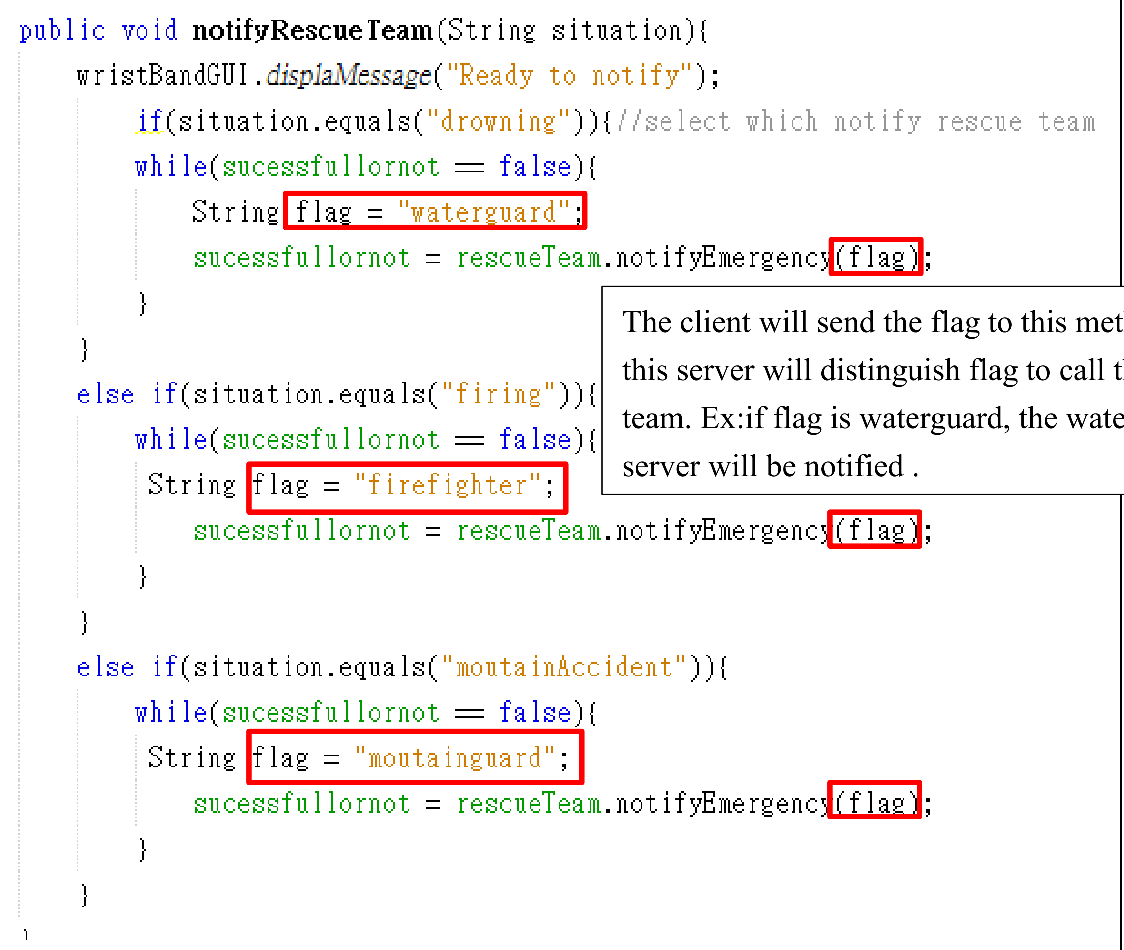
This 'record' method just send primitive type data to medical history class, so is the lowest coupling.

data type – identify



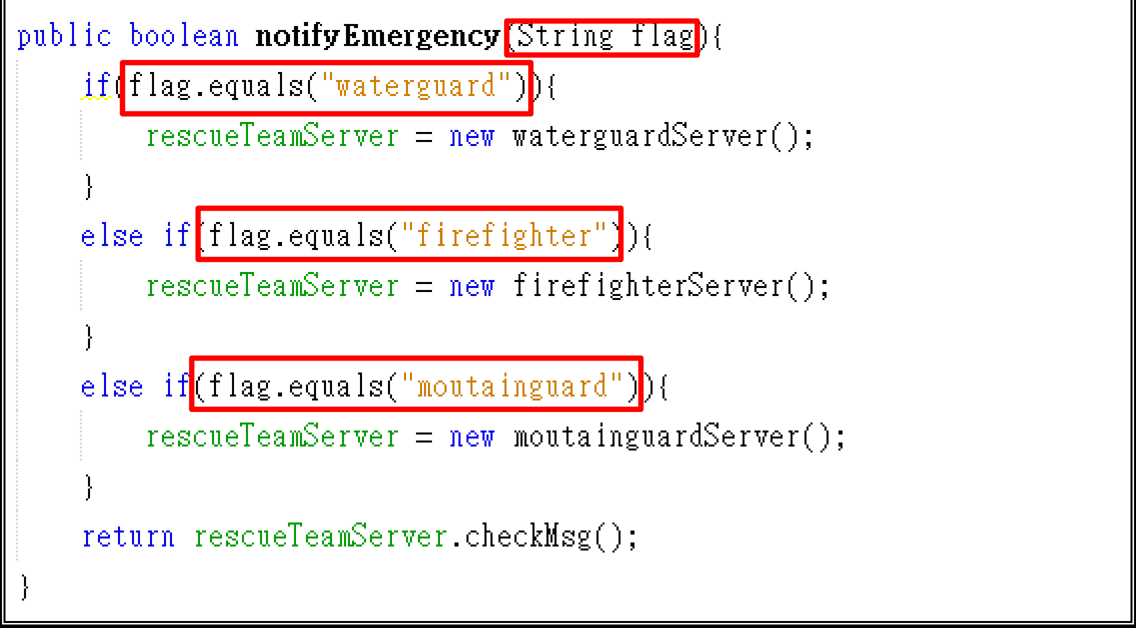
```
String situation = dangerDetermin.identify(mh.getBodyTemperature(),mh.getIdleTime(),
                                            mh.getShakingCount(),mh.getRoomtemperature());//identify situat

public class dangerDetermin {
    public String identify(double bodytempature,double idleTime,double shakeCount,double roomteamerature){
        String situation = new String();
        if(((bodytempature <= 45&&bodytempature >= 30) && shakeCount >= 3)||bodytempature <= 30){
            situation =  "drowning";
        }
        else if(roomteamerature >= 150){
            situation = "firing";
        }
        else if(idleTime >= 100){//>=100hr
            situation = "moutainAccident";
        }
        return situation;
    }
}
```

This method send primitive data type to dangerDermin class to calculate the danger control type



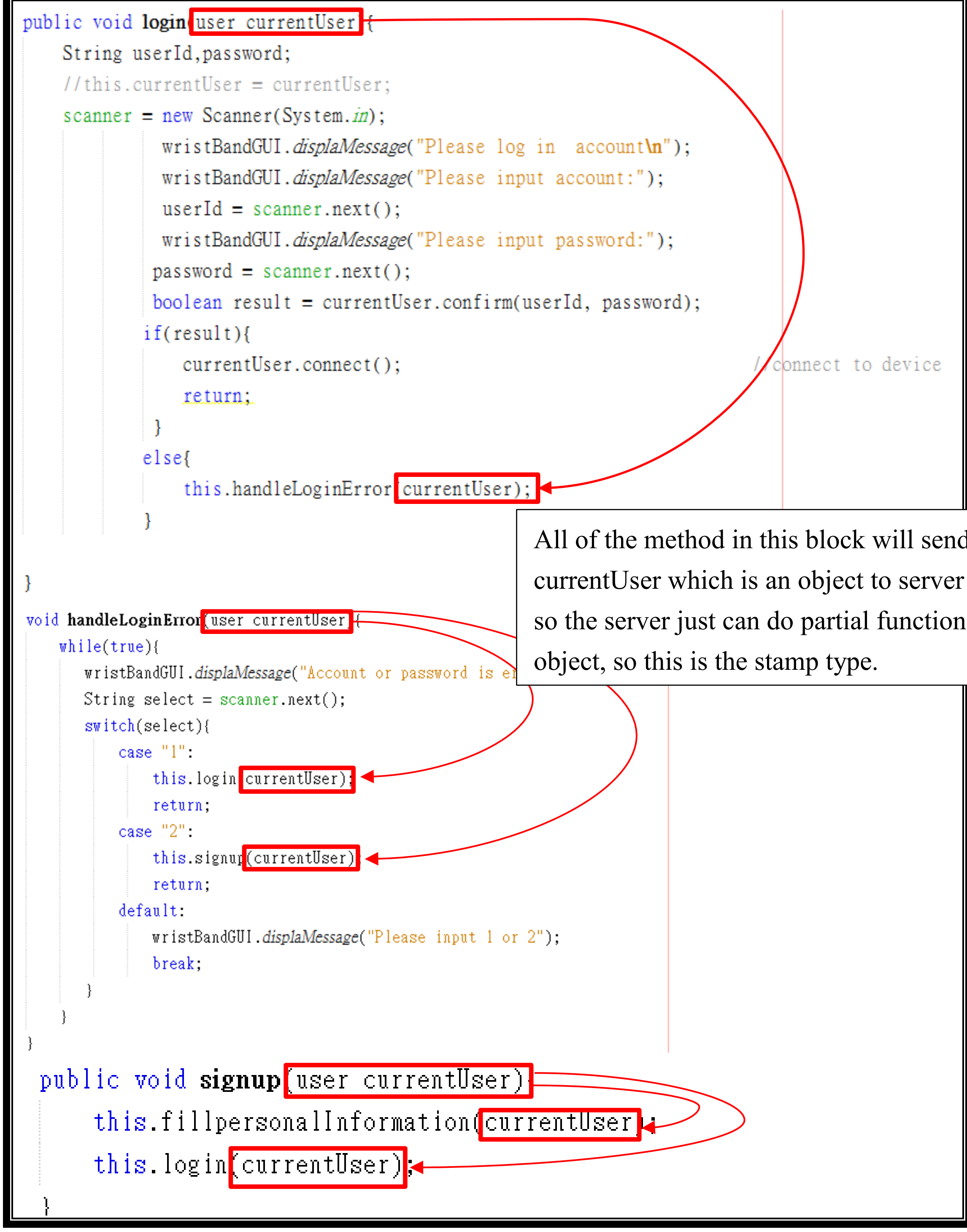
```
public void notifyRescueTeam(String situation){
    wristBandGUI.displaMessage("Ready to notify");
        if(situation.equals("drowning")){//select which notify rescue team
        while(sucessfullornot == false){
            String flag = "waterguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("firing")){
        while(sucessfullornot == false){
          String flag = "firefighter";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("moutainAccident")){
        while(sucessfullornot == false){
          String flag = "moutainguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
}
```

The client will send the flag to this method, and this server will distinguish flag to call the rescue team. Ex:if flag is waterguard, the waterguard server will be notified .



```
public boolean notifyEmergency(String flag){
    if(flag.equals("waterguard")){
        rescueTeamServer = new waterguardServer();
    }
    else if(flag.equals("firefighter")){
        rescueTeamServer = new firefighterServer();
    }
    else if(flag.equals("moutainguard")){
        rescueTeamServer = new moutainguardServer();
    }
    return rescueTeamServer.checkMsg();
}
```

stamp type



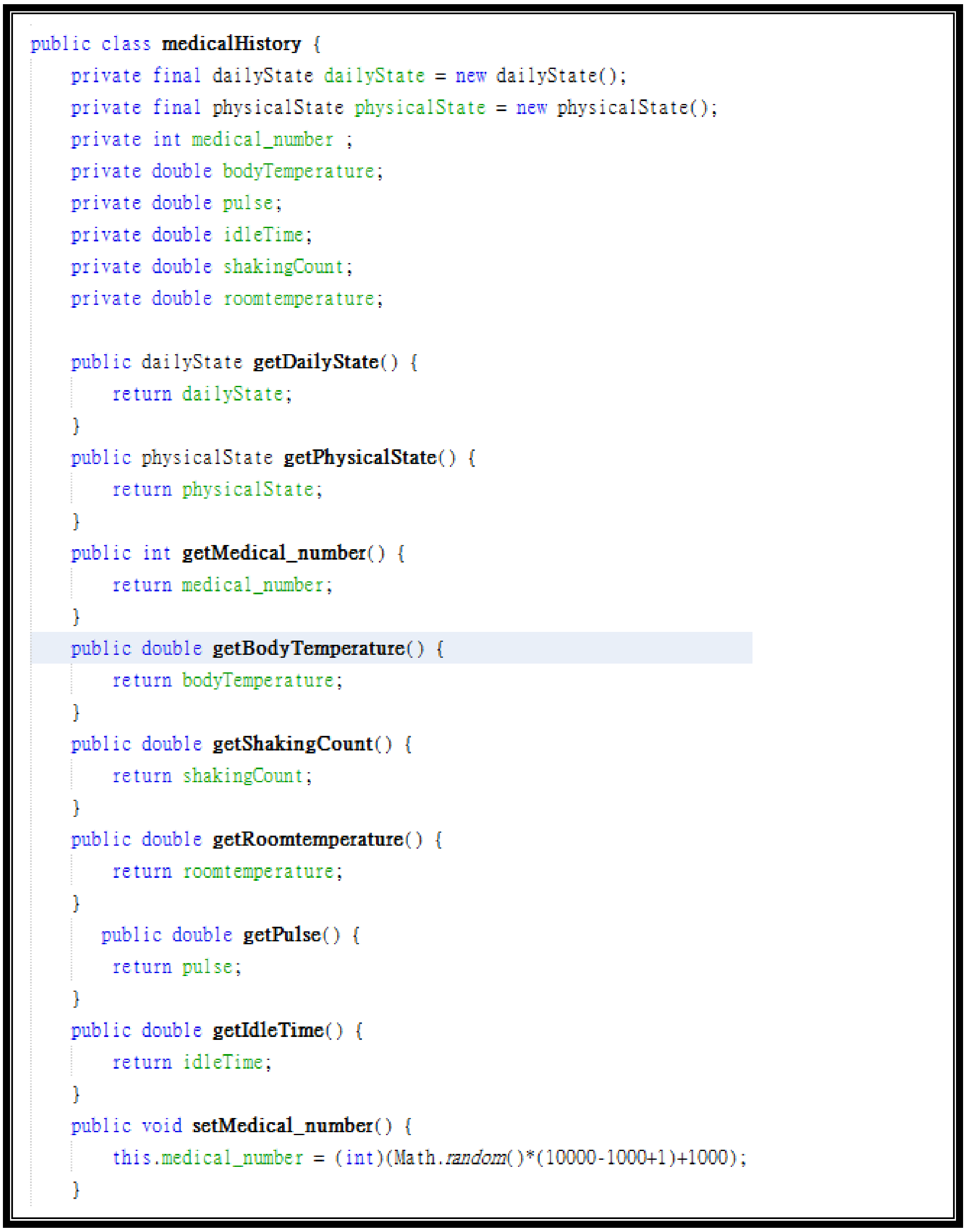
```
public void login(user currentUser){
    String userId,password;
    //this.currentUser = currentUser;
    scanner = new Scanner(System.in);
            wristBandGUI.displaMessage("Please log in  account\n");
            wristBandGUI.displaMessage("Please input account:");
            userId = scanner.next();
            wristBandGUI.displaMessage("Please input password:");
           password = scanner.next();
           boolean result = currentUser.confirm(userId, password);
          if(result){
              currentUser.connect();                                //connect to device
              return;
          }
          else{
              this.handleLoginError(currentUser);
          }

}
void handleLoginError(user currentUser){
    while(true){
        wristBandGUI.displaMessage("Account or password is e
        String select = scanner.next();
        switch(select){
            case "1":
                this.login(currentUser);
                return;
            case "2":
                this.signup(currentUser);
                return;
            default:
                wristBandGUI.displaMessage("Please input 1 or 2");
                break;
        }
    }
}
public void signup(user currentUser){
    this.fillpersonalInformation(currentUser);
    this.login(currentUser);
}
```

All of the method in this block will send the currentUser which is an object to server method, so the server just can do partial function in this object, so this is the stamp type.

3) There are seven types of method cohesion, choose three pieces of your project to describe what types of the cohesion they belong to.

①functional cohesion

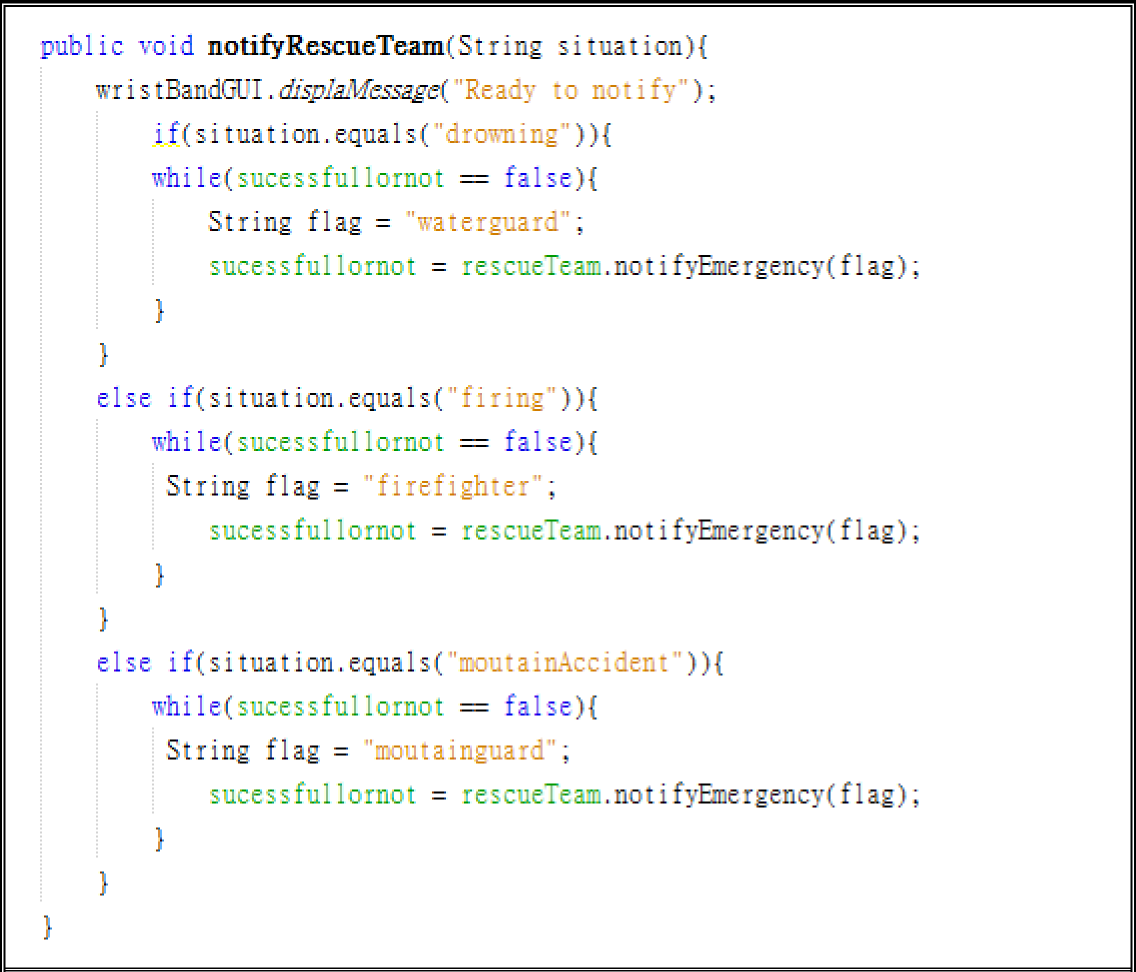
There are some methods: get Body Temperature, get Shaking Count, get Room temperature, get Pulse, get Idle Time. These methods are focus on one thing what they have to do.



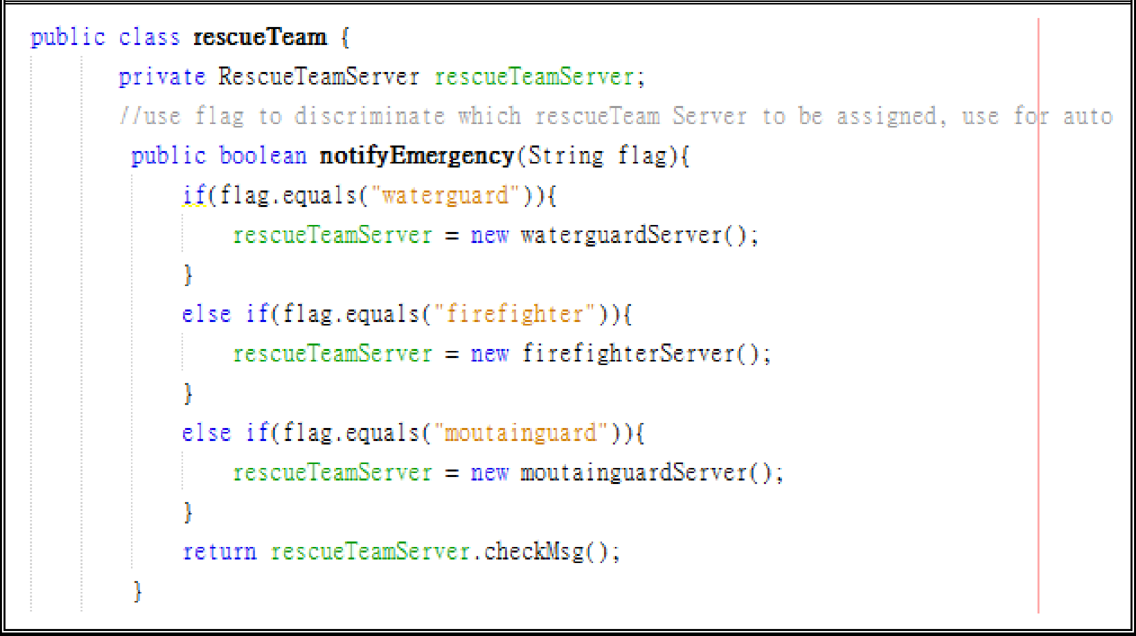
```
public class medicalHistory {
    private final dailyState dailyState = new dailyState();
    private final physicalState physicalState = new physicalState();
    private int medical_number ;
    private double bodyTemperature;
    private double pulse;
    private double idleTime;
    private double shakingCount;
    private double roomtemperature;

    public dailyState getDailyState() {
        return dailyState;
    }
    public physicalState getPhysicalState() {
        return physicalState;
    }
    public int getMedical_number() {
        return medical_number;
    }
    public double getBodyTemperature() {
        return bodyTemperature;
    }
    public double getShakingCount() {
        return shakingCount;
    }
    public double getRoomtemperature() {
        return roomtemperature;
    }
       public double getPulse() {
        return pulse;
    }
    public double getIdleTime() {
        return idleTime;
    }
    public void setMedical_number() {
        this.medical_number = (int)(Math.random()*(10000-1000+1)+1000);
    }
```

②Logical cohesion

In notifyRescueTeam, the control variable is flag, with different value in flag it control which rescueTeamServer will be executed.



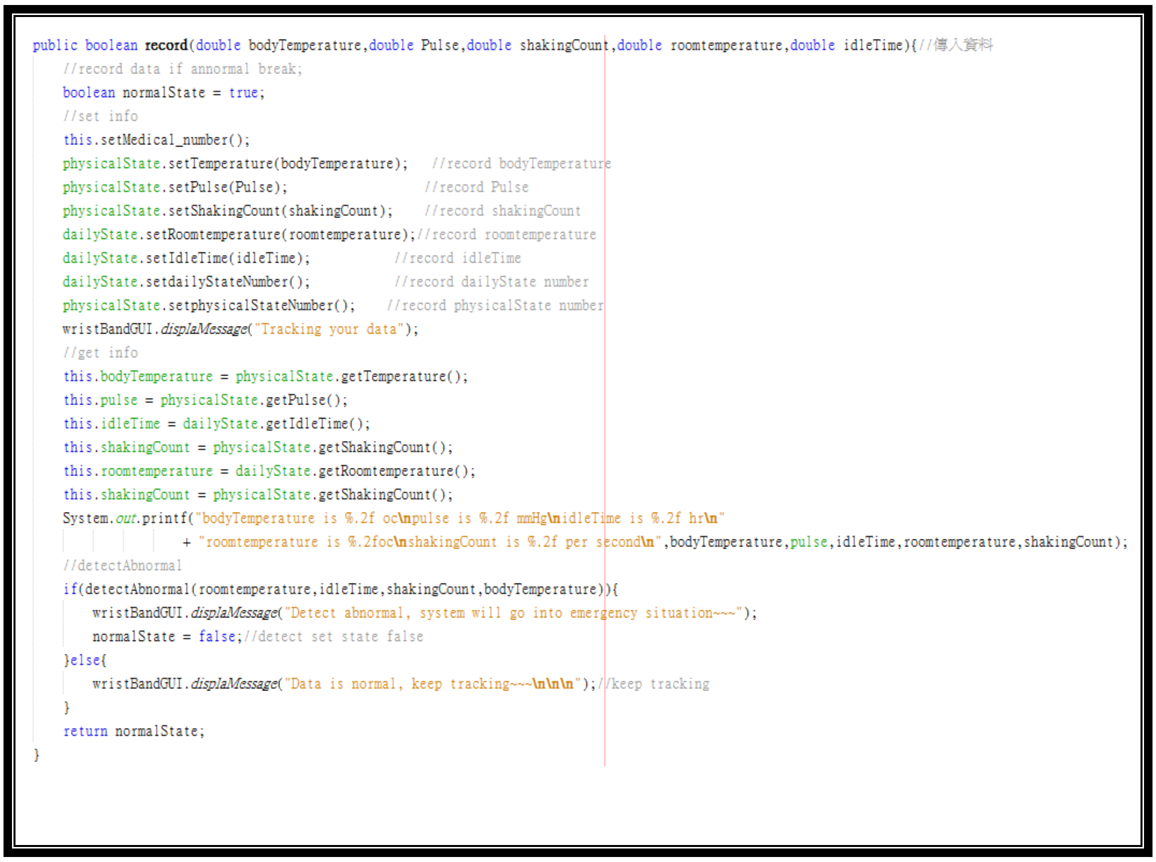
```
public void notifyRescueTeam(String situation){
    wristBandGUI.displaMessage("Ready to notify");
        if(situation.equals("drowning")){
        while(sucessfullornot == false){
            String flag = "waterguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("firing")){
        while(sucessfullornot == false){
         String flag = "firefighter";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("moutainAccident")){
        while(sucessfullornot == false){
         String flag = "moutainguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
}
```



```
public class rescueTeam {
        private RescueTeamServer rescueTeamServer;
        //use flag to discriminate which rescueTeam Server to be assigned, use for auto
         public boolean notifyEmergency(String flag){
            if(flag.equals("waterguard")){
                rescueTeamServer = new waterguardServer();
            }
            else if(flag.equals("firefighter")){
                rescueTeamServer = new firefighterServer();
            }
            else if(flag.equals("moutainguard")){
                rescueTeamServer = new moutainguardServer();
            }
            return rescueTeamServer.checkMsg();
         }
```

③Temporary & Classical cohesion

In record, all the methods keep executing form the user sign in until wristband system detect that medical history is abnormal.



```java
public boolean record(double bodyTemperature,double Pulse,double shakingCount,double roomtemperature,double idleTime){//傳入資料
    //record data if annormal break;
    boolean normalState = true;
    //set info
    this.setMedical_number();
    physicalState.setTemperature(bodyTemperature);   //record bodyTemperature
    physicalState.setPulse(Pulse);                   //record Pulse
    physicalState.setShakingCount(shakingCount);     //record shakingCount
    dailyState.setRoomtemperature(roomtemperature);//record roomtemperature
    dailyState.setIdleTime(idleTime);            //record idleTime
    dailyState.setdailyStateNumber();            //record dailyState number
    physicalState.setphysicalStateNumber();    //record physicalState number
    wristBandGUI.displaMessage("Tracking your data");
    //get info
    this.bodyTemperature = physicalState.getTemperature();
    this.pulse = physicalState.getPulse();
    this.idleTime = dailyState.getIdleTime();
    this.shakingCount = physicalState.getShakingCount();
    this.roomtemperature = dailyState.getRoomtemperature();
    this.shakingCount = physicalState.getShakingCount();
    System.out.printf("bodyTemperature is %.2f oc\npulse is %.2f mmHg\nidleTime is %.2f hr\n"
                + "roomtemperature is %.2foc\nshakingCount is %.2f per second\n",bodyTemperature,pulse,idleTime,roomtemperature,shakingCount);
    //detectAbnormal
    if(detectAbnormal(roomtemperature,idleTime,shakingCount,bodyTemperature)){
        wristBandGUI.displaMessage("Detect abnormal, system will go into emergency situation~~~");
        normalState = false;//detect set state false
    }else{
        wristBandGUI.displaMessage("Data is normal, keep tracking~~~\n\n\n");//keep tracking
    }
    return normalState;
}
```

4) Connascence generalized the ideas of cohesion and coupling, use three pieces of your project to describe what types of the connascence they belong to.

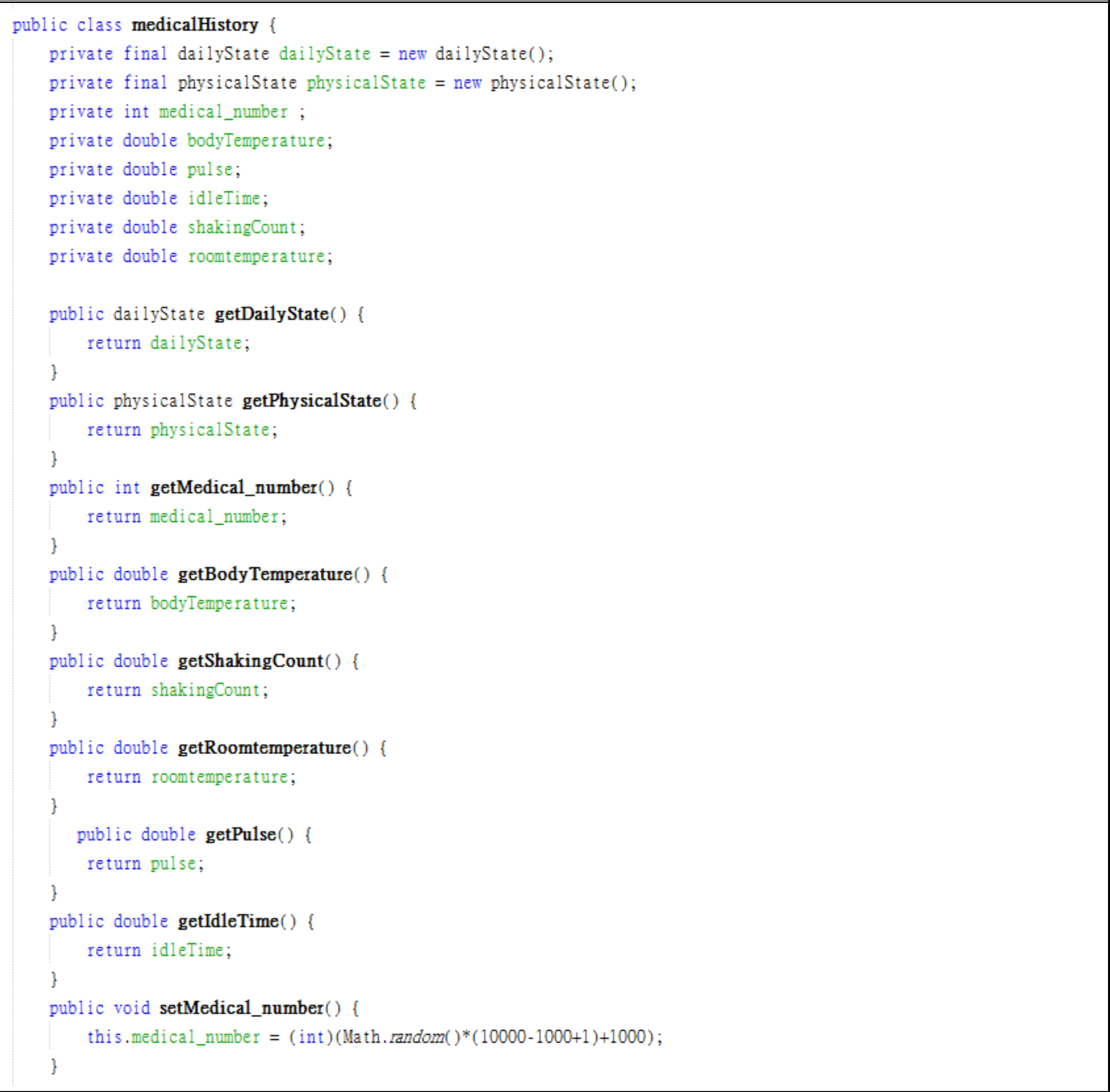
①Name Connascence

If any name of method in medicalHistory changed, then the method in Recording won’t be able to execute successfully.

②Position Connascence

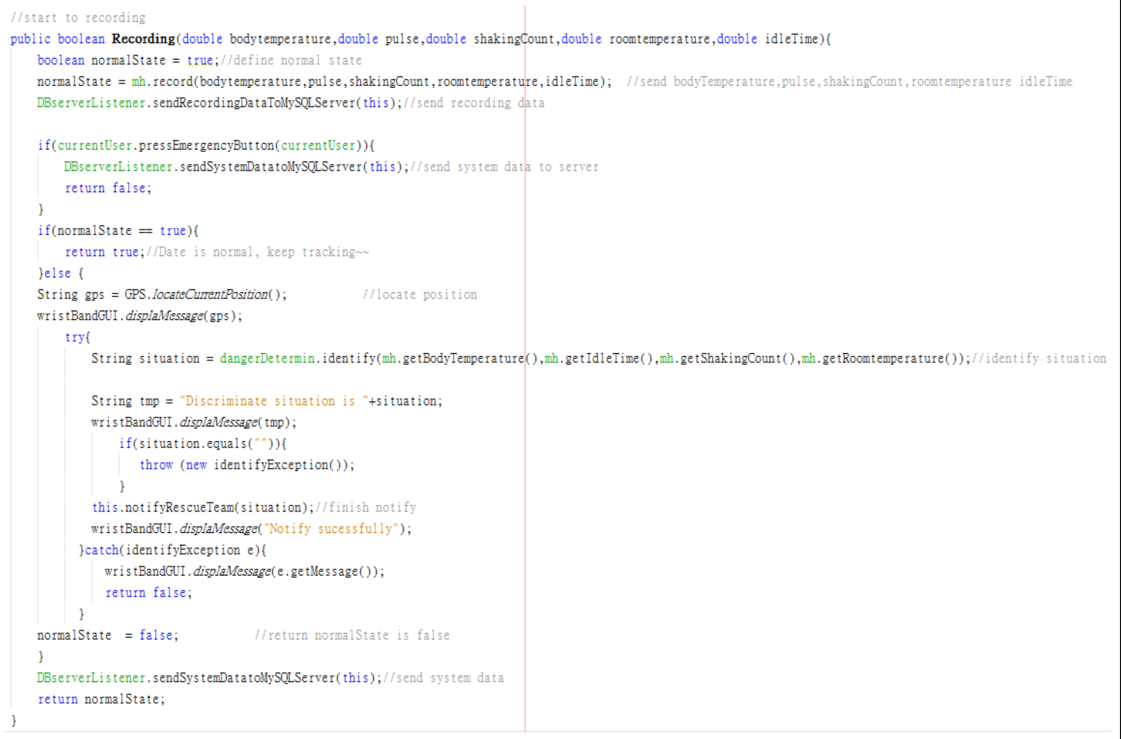
If wristband system sent wrong sequence of argument to dangerDetermin, then the value of argument in medical history will get wrong number, it maybe will detect abnormal situation and notify rescue team, in fact, this is a mistake.

class medicalHistory - 1



```java
public class medicalHistory {
    private final dailyState dailyState = new dailyState();
    private final physicalState physicalState = new physicalState();
    private int medical_number ;
    private double bodyTemperature;
    private double pulse;
    private double idleTime;
    private double shakingCount;
    private double roomtemperature;

    public dailyState getDailyState() {
        return dailyState;
    }
    public physicalState getPhysicalState() {
        return physicalState;
    }
    public int getMedical_number() {
        return medical_number;
    }
    public double getBodyTemperature() {
        return bodyTemperature;
    }
    public double getShakingCount() {
        return shakingCount;
    }
    public double getRoomtemperature() {
        return roomtemperature;
    }
      public double getPulse() {
        return pulse;
    }
    public double getIdleTime() {
        return idleTime;
    }
    public void setMedical_number() {
        this.medical_number = (int)(Math.random()*(10000-1000+1)+1000);
    }
```

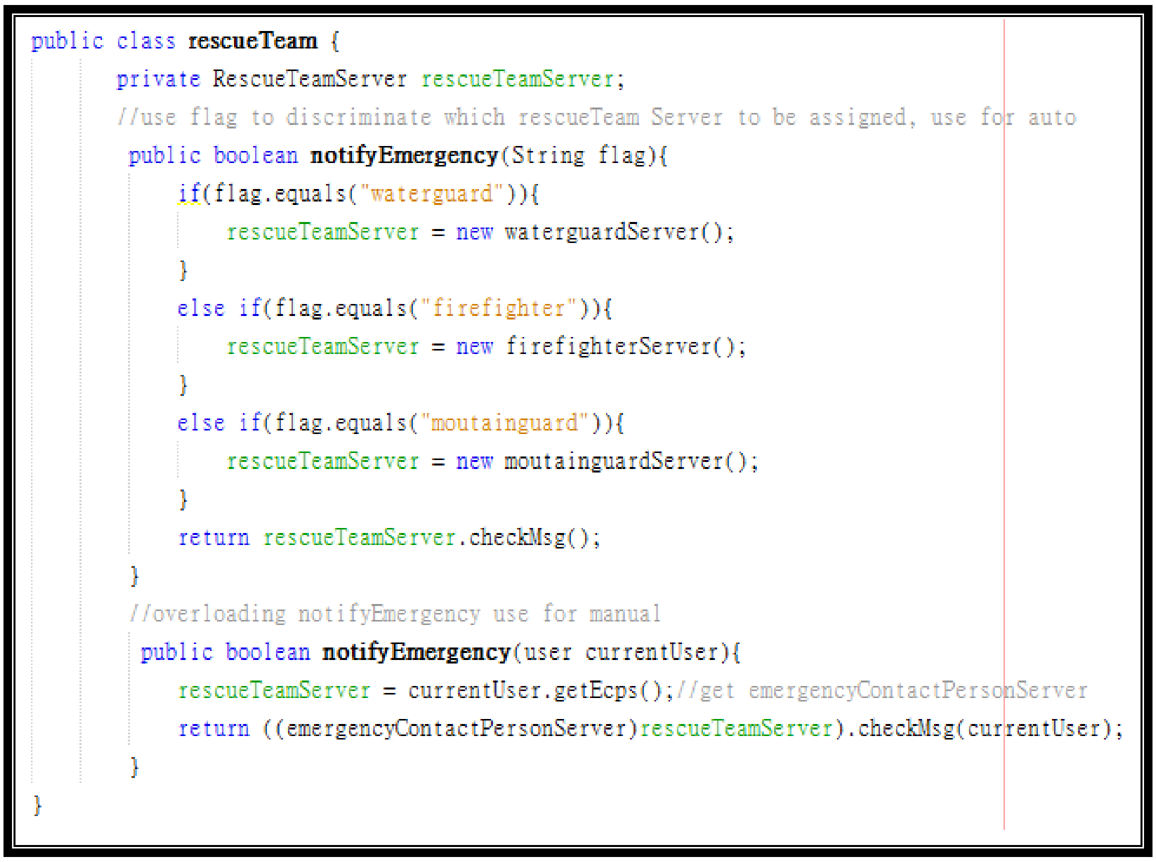
class medicalHistory - 2



```
//start to recording
public boolean Recording(double bodytemperature,double pulse,double shakingCount,double roomtemperature,double idleTime){
    boolean normalState = true;//define normal state
    normalState = mh.record(bodytemperature,pulse,shakingCount,roomtemperature,idleTime);  //send bodyTemperature,pulse,shakingCount,roomtemperature idleTime
    DBserverListener.sendRecordingDataToMySQLServer(this);//send recording data

    if(currentUser.pressEmergencyButton(currentUser)){
        DBserverListener.sendSystemDatatoMySQLServer(this);//send system data to server
        return false;
    }
    if(normalState == true){
        return true;//Date is normal, keep tracking~~
    }else {
    String gps = GPS.locateCurrentPosition();           //locate position
    wristBandGUI.displaMessage(gps);
        try{
            String situation = dangerDetermin.identify(mh.getBodyTemperature(),mh.getIdleTime(),mh.getShakingCount(),mh.getRoomtemperature());//identify situation

            String tmp = "Discriminate situation is "+situation;
            wristBandGUI.displaMessage(tmp);
                if(situation.equals("")){
                   throw (new identifyException());
                }
            this.notifyRescueTeam(situation);//finish notify
            wristBandGUI.displaMessage("Notify sucessfully");
          }catch(identifyException e){
             wristBandGUI.displaMessage(e.getMessage());
             return false;
          }
    normalState  = false;           //return normalState is false
    }
    DBserverListener.sendSystemDatatoMySQLServer(this);//send system data
    return normalState;
}
```

③Algorithm Connascence

Both of notifyEmergency(String flag) and notifyEmergency(user currentUser) are rely on notifyRescueTeam method.



```
public class rescueTeam {
        private RescueTeamServer rescueTeamServer;
        //use flag to discriminate which rescueTeam Server to be assigned, use for auto
         public boolean notifyEmergency(String flag){
             if(flag.equals("waterguard")){
                 rescueTeamServer = new waterguardServer();
             }
             else if(flag.equals("firefighter")){
                 rescueTeamServer = new firefighterServer();
             }
             else if(flag.equals("moutainguard")){
                 rescueTeamServer = new moutainguardServer();
             }
             return rescueTeamServer.checkMsg();
         }
         //overloading notifyEmergency use for manual
          public boolean notifyEmergency(user currentUser){
             rescueTeamServer = currentUser.getEcps();//get emergencyContactPersonServer
             return ((emergencyContactPersonServer)rescueTeamServer).checkMsg(currentUser);
         }
}
```

5) Use one class from your project that can create a set of invariants and add them to the CRC card or the class diagram.

CRC card

Front

Class name: WristBandSystem	**ID:**1	**Type:** concrete ,Domain

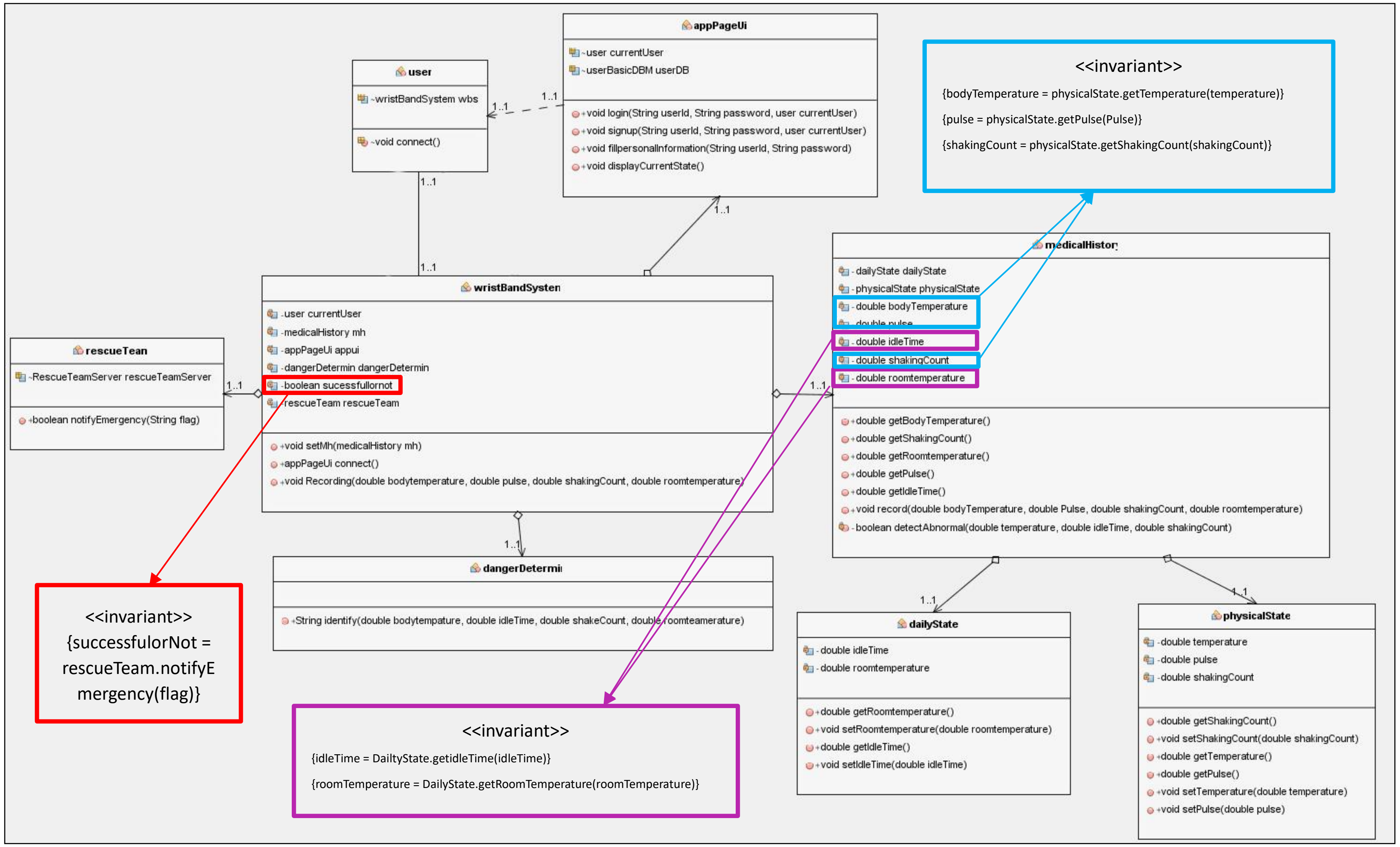
Description:	**Association Use Case:**
Record user's physical states and Daily States in real-time. If detect abnormal states, then identify which situation could be happen. Eventually notify rescue team.	Record Daily State Record Physical State Auto Notify Emergency Situation
Responsibilities:	**Collaborators:**
connect	appPageUi
setMH	medicalHistory
recording	medicalHistory
	GPS
	dangerousDetermin
	rescueTeam

Back

Attributes:			
currentUser	(1..1)	(user)	
mh	(1..1)	(medicalHistory)	
appUi	(1..1)	(appPageUi)	
DangerousDetermin	(1..1)	(dangerousDetermin)	
successfulorNot	(1..1)	(Boolean)	{successfulorNot = rescueTeam.notifyEmergency(flag)}
rescueTeam	(1..1)	(rescueTeam)	
Relationships:			
Generalization(a-kind-of):			
Aggregation(has-parts):			

appPageU{1..1}
user{1..1}
medicalHistory{1..1}
dangerDetermine{1..1}
rescueTeam{1..1}			

Other Associations:			
User{1..1}			


appPageUi
~user currentUser
~userBasicDBM userDB
+void login(String userId, String password, user currentUser)
+void signup(String userId, String password, user currentUser)
+void fillpersonalInformation(String userId, String password)
+void displayCurrentState()
user
~wristBandSystem wbs
~void connect()
1..1
1..1
1..1
1..1
1..1
wristBandSysten
-user currentUser
-medicalHistory mh
-appPageUi appui
-dangerDetermin dangerDetermin
-boolean sucessfullornot
-rescueTeam rescueTeam
+void setMh(medicalHistory mh)
+appPageUi connect()
+void Recording(double bodytemperature, double pulse, double shakingCount, double roomtemperature)
rescueTean
-RescueTeamServer rescueTeamServer
+boolean notifyEmergency(String flag)
1..1
<<invariant>>
{successfulorNot = rescueTeam.notifyEmergency(flag)}
dangerDetermi
+String identify(double bodytempature, double idleTime, double shakeCount, double roomteamerature)
1..1
<<invariant>>
{idleTime = DailtyState.getidleTime(idleTime)}
{roomTemperature = DailyState.getRoomTemperature(roomTemperature)}
<<invariant>>
{bodyTemperature = physicalState.getTemperature(temperature)}
{pulse = physicalState.getPulse(Pulse)}
{shakingCount = physicalState.getShakingCount(shakingCount)}
medicalHistor
-dailyState dailyState
-physicalState physicalState
-double bodyTemperature
-double pulse
-double idleTime
-double shakingCount
-double roomtemperature
1..1
+double getBodyTemperature()
+double getShakingCount()
+double getRoomtemperature()
+double getPulse()
+double getIdleTime()
+void record(double bodyTemperature, double Pulse, double shakingCount, double roomtemperature)
-boolean detectAbnormal(double temperature, double idleTime, double shakingCount)
1..1
1..1
dailyState
-double idleTime
-double roomtemperature
+double getRoomtemperature()
+void setRoomtemperature(double roomtemperature)
+double getIdleTime()
+void setIdleTime(double idleTime)
physicalState
-double temperature
-double pulse
-double shakingCount
+double getShakingCount()
+void setShakingCount(double shakingCount)
+double getTemperature()
+double getPulse()
+void setTemperature(double temperature)
+void setPulse(double pulse)

Class Diagram

Text File

```
WristBandSystem Class invariants:
   successfulorNot = rescueTeam.notifyEmergency(flag)
```

```
MedicalHistory Class invariants:
   bodyTemperature = physicalState.getTemperature(temperature)
   pulse = physicalState.getPulse(Pulse)
   shakingCount = physicalState.getShakingCount(shakingCount)
   idleTime = DailtyState.getidleTime(idleTime)
   roomTemperature = DailyState.getRoomTemperature(roomTemperature)
```

6) Use a method of a class from your project that can create a contract and describe its algorithm specification. Specify the pre- or post- condition and use both Structured English and an activity diagram to specify the algorithm.

Contract

Method Name: notifyRescueTeam	**Class Name:** wristBandSystem	**ID:** 1
Client(consumers): wristBandSystem		
Associated Use Case: Automatically notify emergency situation		
Description of Responsibilities: Wristband system gets accident situation, send flag to rescue team and rescue return successfully receive SOS message, if received print "Notify sucessfully", or notify rescue team again.		
Arguments Received: situation: String		
Pre-Conditions: sucessfullornot==false		
Post-Conditions: assert(Sucessfullornot)		

Method Specification

Method Name: notifyRescueTeam	**Class Name:** wristBandSystem	**ID:**
Contract ID:	**Programmer:** Kevin	Data Due: 12/22/17

Programming Language:

Java

Triggers/Events:

System wants to notify emergency message to rescue team

Arguments Received: **Data Type:**	**Notes:**
String	Emergency situation

Messages Sent & Argument Passed: **ClassName.MethodName:**	**Data Type:**	**Notes:**
rescueTeam.notifyEmergency(flag)	boolean	

Arguments Returned: **Data Type:**	**Notes:**
void	

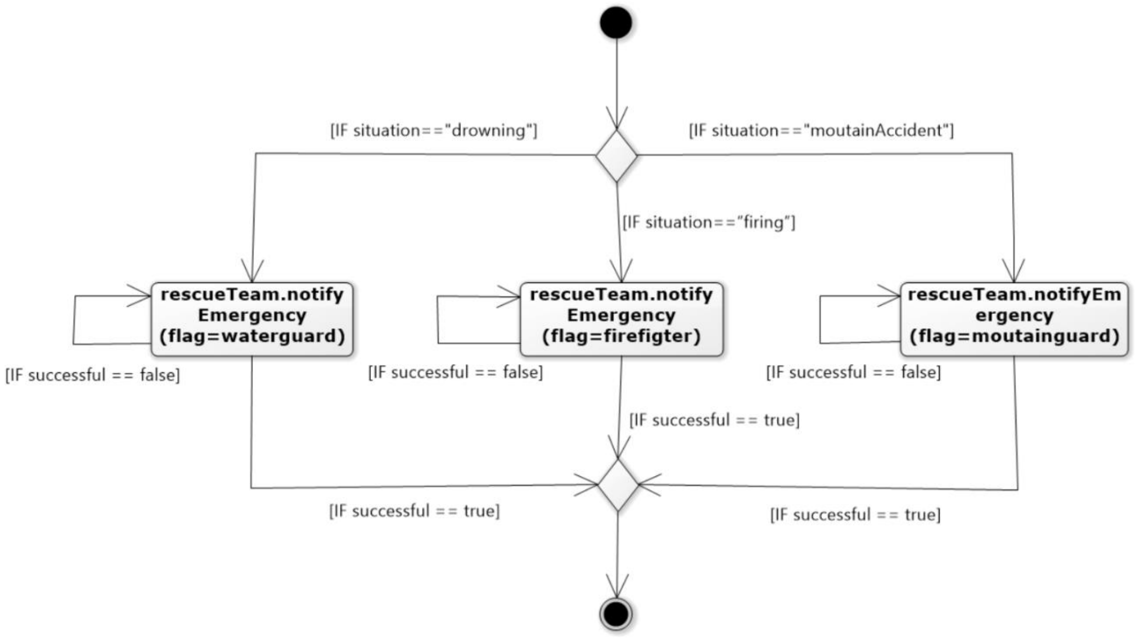
Algorithm Specification:

```
IF situation==”drowning”
    WHILE not sucessfullornot
        flag=”waterguard”
        sucessfullornot = rescueTeam.notifyEmergency(flag)
ElSE
    IF situation=="firing"
        WHILE not sucessfullornot
            flag="firefighter"
            sucessfullornot = rescueTeam.notifyEmergency(flag)
    ELSE
        IF situation=="moutainAccident"
            WHILE not sucessfullornot
                flag="moutainguard"
                sucessfullornot = rescueTeam.notifyEmergency(flag)
```

Misc.Notes:

None

Activity Diagram


[IF situation=="drowning"]
[IF situation=="moutainAccident"]
[IF situation=="firing"]
rescueTeam.notify Emergency (flag=waterguard)
rescueTeam.notify Emergency (flag=firefigter)
rescueTeam.notifyEmergency (flag=moutainguard)
[IF successful == false]
[IF successful == false]
[IF successful == false]
[IF successful == true]
[IF successful == true]
[IF successful == true]

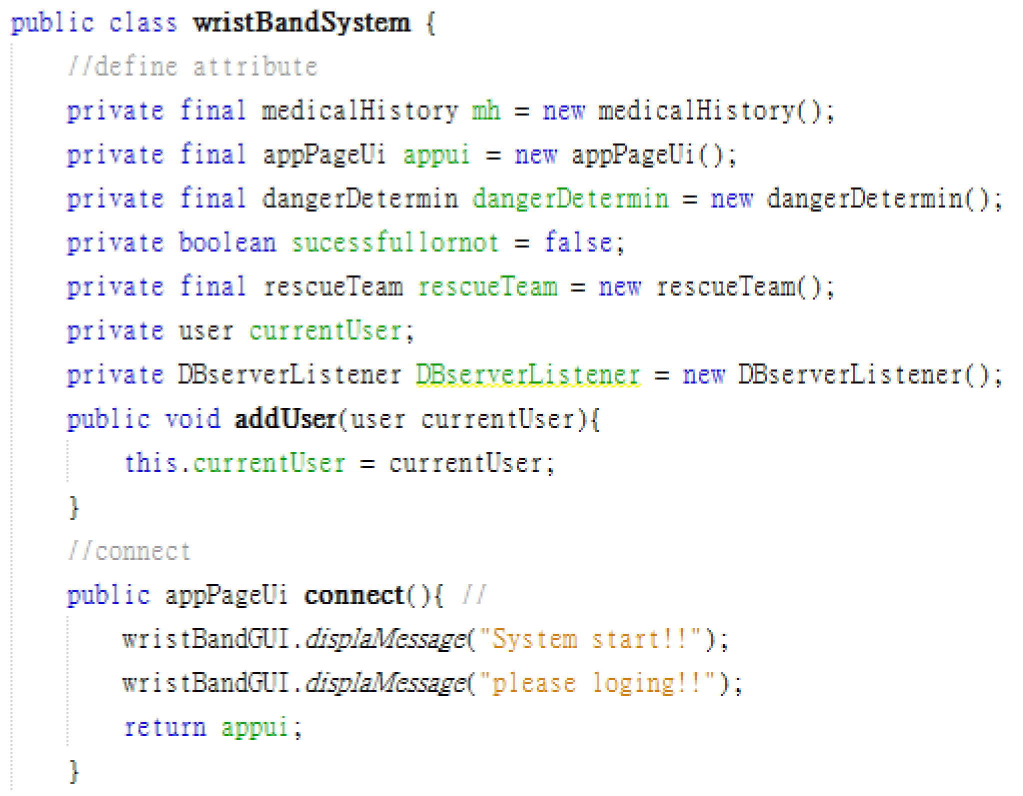
7) Please evaluate any piece of your project in terms of cohesion, coupling, and connascence perspective.

Coupling (Interaction, data coupling) (①)
Medical history keeps recording user’s body temperature, pulse, shaking count, room temperature, idle time, and wristband system call the method record.

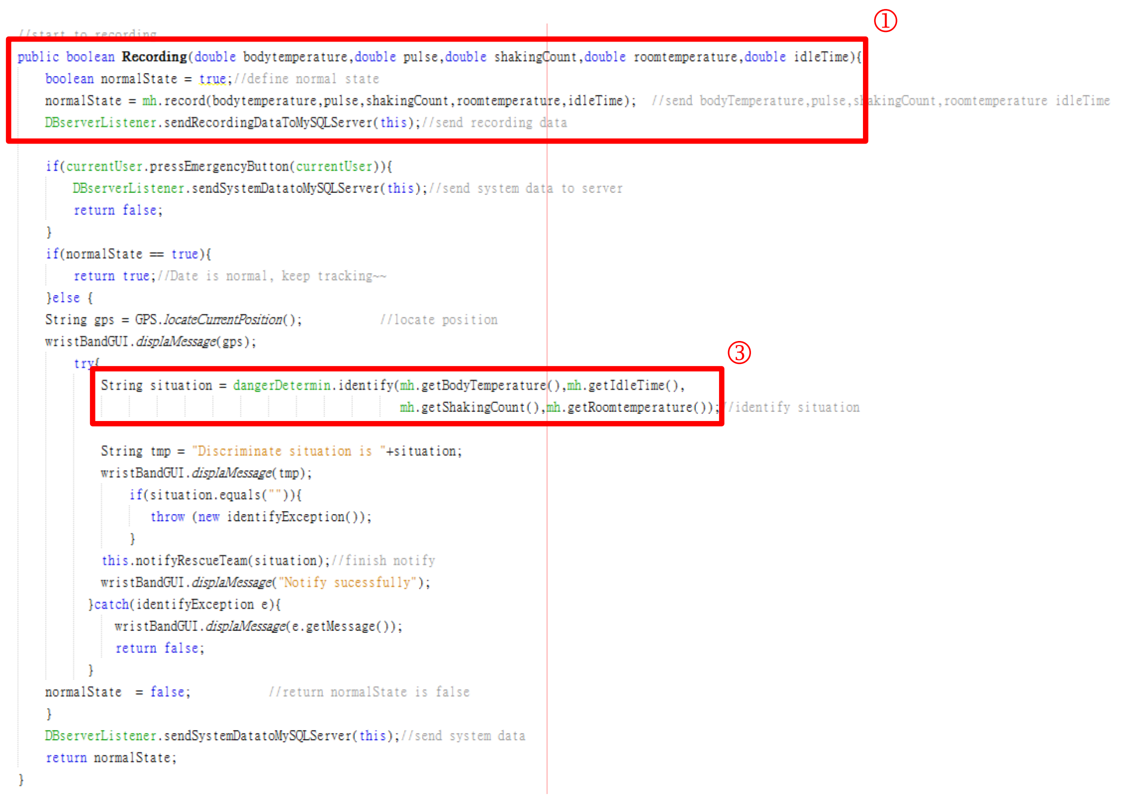
Cohesion (Method, functional cohesion) (②)
In medical history, there are some methods: get Body Temperature, get Shaking Count, get Room temperature, get Pulse, get Idle Time. These methods are focus on one thing what they have to do.

Connascence (Position connascence) (③)
If wristband system sent wrong sequence of argument to dangerDetermin, then the value of argument in medical history will get wrong number, it maybe will detect abnormal situation and notify rescue team, in fact, this is a mistake.

class wristBandSystem - 1



```
public class wristBandSystem {
    //define attribute
    private final medicalHistory mh = new medicalHistory();
    private final appPageUi appui = new appPageUi();
    private final dangerDetermin dangerDetermin = new dangerDetermin();
    private boolean sucessfullornot = false;
    private final rescueTeam rescueTeam = new rescueTeam();
    private user currentUser;
    private DBserverListener DBserverListener = new DBserverListener();
    public void addUser(user currentUser){
        this.currentUser = currentUser;
    }
    //connect
    public appPageUi connect(){ //
        wristBandGUI.displaMessage("System start!!");
        wristBandGUI.displaMessage("please loging!!");
        return appui;
    }
```



①

```
//start to recording
public boolean Recording(double bodytemperature,double pulse,double shakingCount,double roomtemperature,double idleTime){
    boolean normalState = true;//define normal state
    normalState = mh.record(bodytemperature,pulse,shakingCount,roomtemperature,idleTime);  //send bodyTemperature,pulse,shakingCount,roomtemperature idleTime
    DBserverListener.sendRecordingDataToMySQLServer(this);//send recording data

    if(currentUser.pressEmergencyButton(currentUser)){
        DBserverListener.sendSystemDatatoMySQLServer(this);//send system data to server
        return false;
    }
    if(normalState == true){
        return true;//Date is normal, keep tracking~~
    }else {
    String gps = GPS.locateCurrentPosition();           //locate position
    wristBandGUI.displaMessage(gps);
        try{
```

③

```
            String situation = dangerDetermin.identify(mh.getBodyTemperature(),mh.getIdleTime(),
                                                    mh.getShakingCount(),mh.getRoomtemperature());//identify situation

            String tmp = "Discriminate situation is "+situation;
            wristBandGUI.displaMessage(tmp);
                if(situation.equals("")){
                   throw (new identifyException());
                }
            this.notifyRescueTeam(situation);//finish notify
            wristBandGUI.displaMessage("Notify sucessfully");
        }catch(identifyException e){
            wristBandGUI.displaMessage(e.getMessage());
            return false;
        }
    normalState  = false;           //return normalState is false
    }
    DBserverListener.sendSystemDatatoMySQLServer(this);//send system data
    return normalState;
}
```

class wristBandSystem – 2

```
//use for auto
public void notifyRescueTeam(String situation){
    wristBandGUI.displaMessage("Ready to notify");
        if(situation.equals("drowning")){//select which notify rescue team
        while(sucessfullornot == false){
            String flag = "waterguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("firing")){
        while(sucessfullornot == false){
         String flag = "firefighter";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
    else if(situation.equals("moutainAccident")){
        while(sucessfullornot == false){
         String flag = "moutainguard";
            sucessfullornot = rescueTeam.notifyEmergency(flag);
        }
    }
}
//overolading notifyRescueTeam use for manual
public void notifyRescueTeam(user currentUser){
    wristBandGUI.displaMessage("Ready to notify "+
                        currentUser.getEmergencyContactPerson()+" person");
        while(sucessfullornot == false){
            sucessfullornot = rescueTeam.notifyEmergency(currentUser);
        }
    wristBandGUI.displaMessage("Notify sucessfully");
}

public medicalHistory getMh() {
    return mh;
}

public int isSucessfullornot() {
    if(sucessfullornot == true){
        return 1;
    }
    return 0;
}

public user getCurrentUser() {
    return currentUser;
}
```

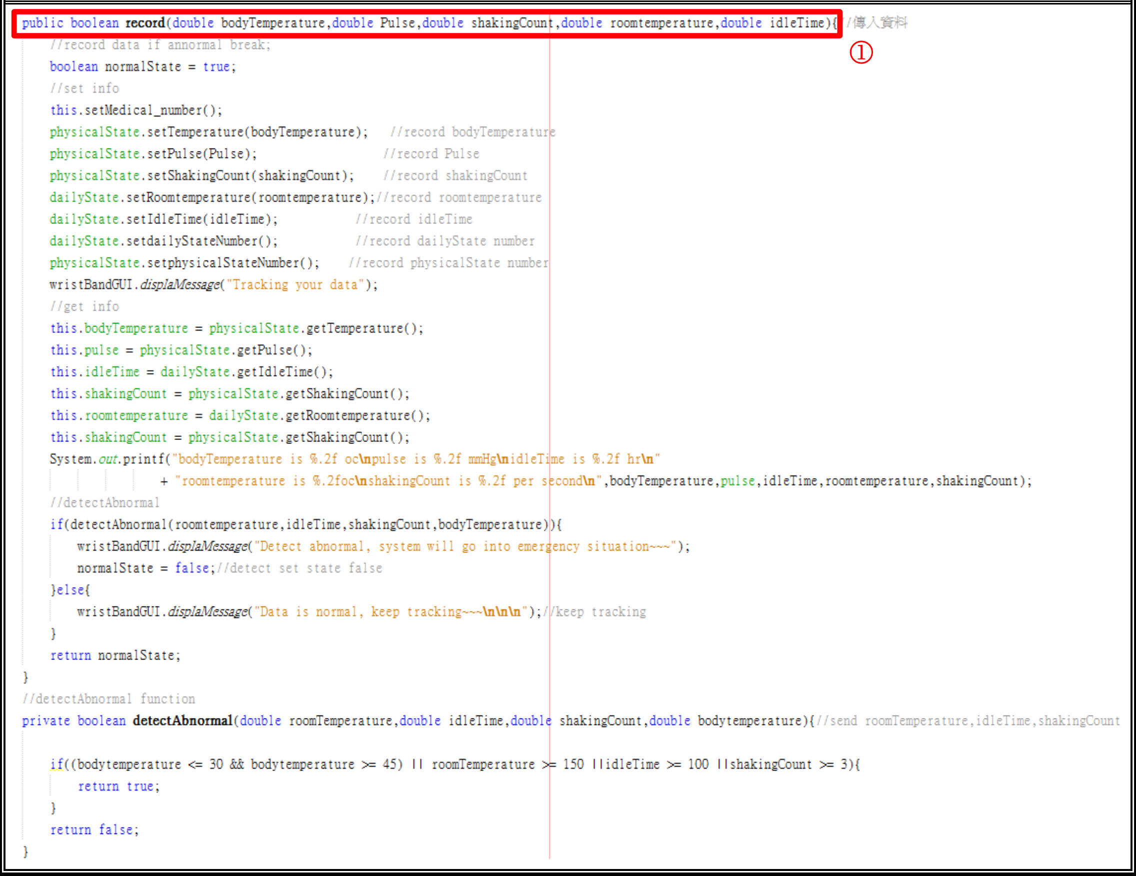
class medicalHistory - 1

```
public class medicalHistory {
    private final dailyState dailyState = new dailyState();
    private final physicalState physicalState = new physicalState();
    private int medical_number ;
    private double bodyTemperature;
    private double pulse;
    private double idleTime;
    private double shakingCount;
    private double roomtemperature;
```

②

```
    public dailyState getDailyState() {
        return dailyState;
    }
    public physicalState getPhysicalState() {
        return physicalState;
    }
    public int getMedical_number() {
        return medical_number;
    }
    public double getBodyTemperature() {
        return bodyTemperature;
    }
    public double getShakingCount() {
        return shakingCount;
    }
    public double getRoomtemperature() {
        return roomtemperature;
    }
       public double getPulse() {
        return pulse;
    }
    public double getIdleTime() {
        return idleTime;
    }
    public void setMedical_number() {
        this.medical_number = (int)(Math.random()*(10000-1000+1)+1000);
    }
```

class medicalHistory - 2



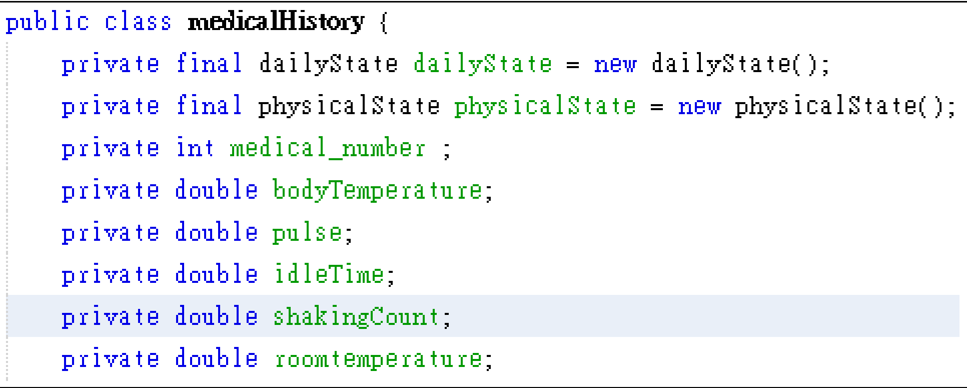
```
public boolean record(double bodyTemperature,double Pulse,double shakingCount,double roomtemperature,double idleTime){//傳入資料
                                                                                                                 ①
    //record data if annormal break;
    boolean normalState = true;
    //set info
    this.setMedical_number();
    physicalState.setTemperature(bodyTemperature);   //record bodyTemperature
    physicalState.setPulse(Pulse);                   //record Pulse
    physicalState.setShakingCount(shakingCount);     //record shakingCount
    dailyState.setRoomtemperature(roomtemperature);//record roomtemperature
    dailyState.setIdleTime(idleTime);              //record idleTime
    dailyState.setdailyStateNumber();              //record dailyState number
    physicalState.setphysicalStateNumber();      //record physicalState number
    wristBandGUI.displaMessage("Tracking your data");
    //get info
    this.bodyTemperature = physicalState.getTemperature();
    this.pulse = physicalState.getPulse();
    this.idleTime = dailyState.getIdleTime();
    this.shakingCount = physicalState.getShakingCount();
    this.roomtemperature = dailyState.getRoomtemperature();
    this.shakingCount = physicalState.getShakingCount();
    System.out.printf("bodyTemperature is %.2f oc\npulse is %.2f mmHg\nidleTime is %.2f hr\n"
                + "roomtemperature is %.2foc\nshakingCount is %.2f per second\n",bodyTemperature,pulse,idleTime,roomtemperature,shakingCount);
    //detectAbnormal
    if(detectAbnormal(roomtemperature,idleTime,shakingCount,bodyTemperature)){
        wristBandGUI.displaMessage("Detect abnormal, system will go into emergency situation~~~");
        normalState = false;//detect set state false
    }else{
        wristBandGUI.displaMessage("Data is normal, keep tracking~~~\n\n\n");//keep tracking
    }
    return normalState;
}
//detectAbnormal function
private boolean detectAbnormal(double roomTemperature,double idleTime,double shakingCount,double bodytemperature){//send roomTemperature,idleTime,shakingCount

    if((bodytemperature <= 30 && bodytemperature >= 45) || roomTemperature >= 150 ||idleTime >= 100 ||shakingCount >= 3){
        return true;
    }
    return false;
}
```

8) What are the factors in determining the type of object persistence format that will be adopted in your project?

We choose RDBMS for our Wristband system. Please see the following reasons.

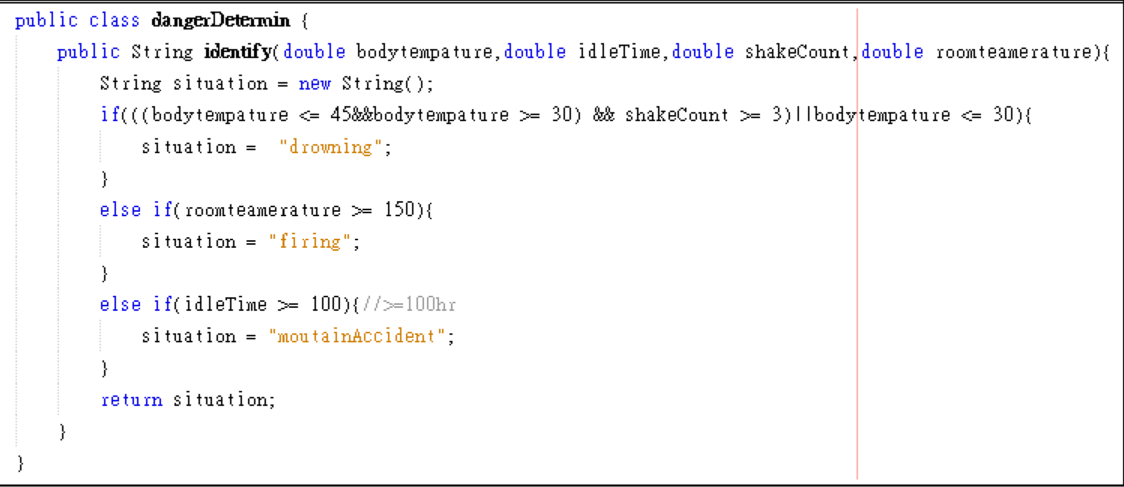
1. All of our data are structured. For examples in our medical history data like pulse, body temperature, hand shaking count and these data are all numeric.
2. Our system does not require to process over complicated data. Our data are only use to basic calculation or determining the value is out of range or not.
3. Since the data or table of our system are simple, the relations of table are simple too. It seldom occurs impendence mismatch.
4. Additionally, in the security issue, something of our data are very confidential, like user name, password, address, emergency contact person name and emergency contact person number. etc. After observing the market, the RDBMS implements nearly perfect 'ACID' concept, which can ensure data being consistent, and can be durable, and it supply confidentiality, integrity and encryption to secure our user data; we think this is another the very important reason concerning our project.

1. All of our data are structured. For examples in our medical history data like pulse, body temperature, hand shaking count and these data are all numeric.



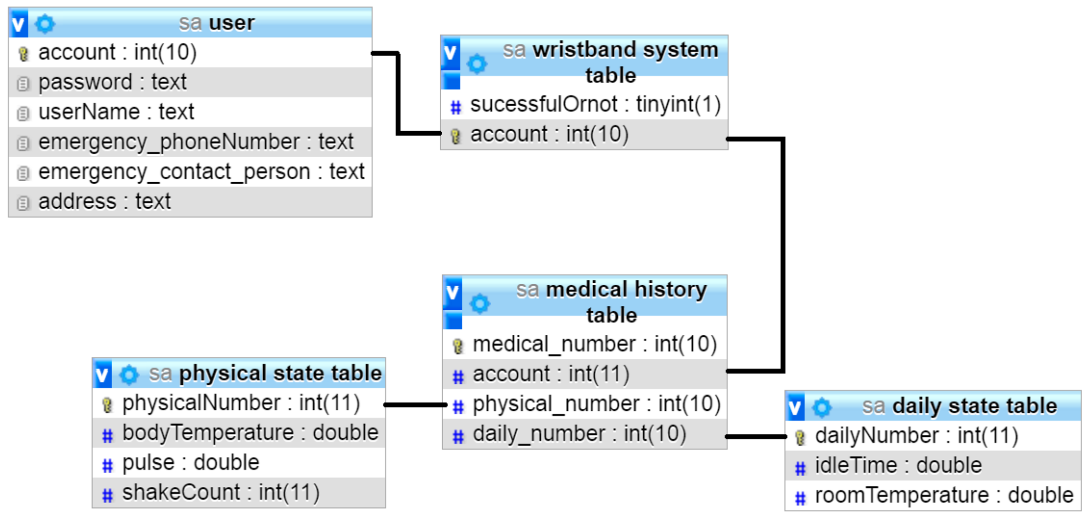
```
public class medicalHistory {
    private final dailyState dailyState = new dailyState();
    private final physicalState physicalState = new physicalState();
    private int medical_number ;
    private double bodyTemperature;
    private double pulse;
    private double idleTime;
    private double shakingCount;
    private double roomtemperature;
```

2. Our system does not require to process over complicated data. Our data are only use to basic calculation or determining the value is out of range or not.



```
public class dangerDetermin {
    public String identify( double bodytempature,double idleTime,double shakeCount,double roomteamerature){
        String situation = new String();
        if(((bodytempature <= 45&&bodytempature >= 30) && shakeCount >= 3)||bodytempature <= 30){
            situation =  "drowning";
        }
        else if( roomteamerature >= 150){
            situation = "firing";
        }
        else if(idleTime >= 100){//>=100hr
            situation = "moutainAccident";
        }
        return situation;
    }
}
```

3. Since the data or table of our system are simple, the relations of table are simple too. It seldom occurs impendence mismatch.



sa user
account : int(10)
password : text
userName : text
emergency_phoneNumber : text
emergency_contact_person : text
address : text

sa wristband system table
sucessfulOrnot : tinyint(1)
account : int(10)

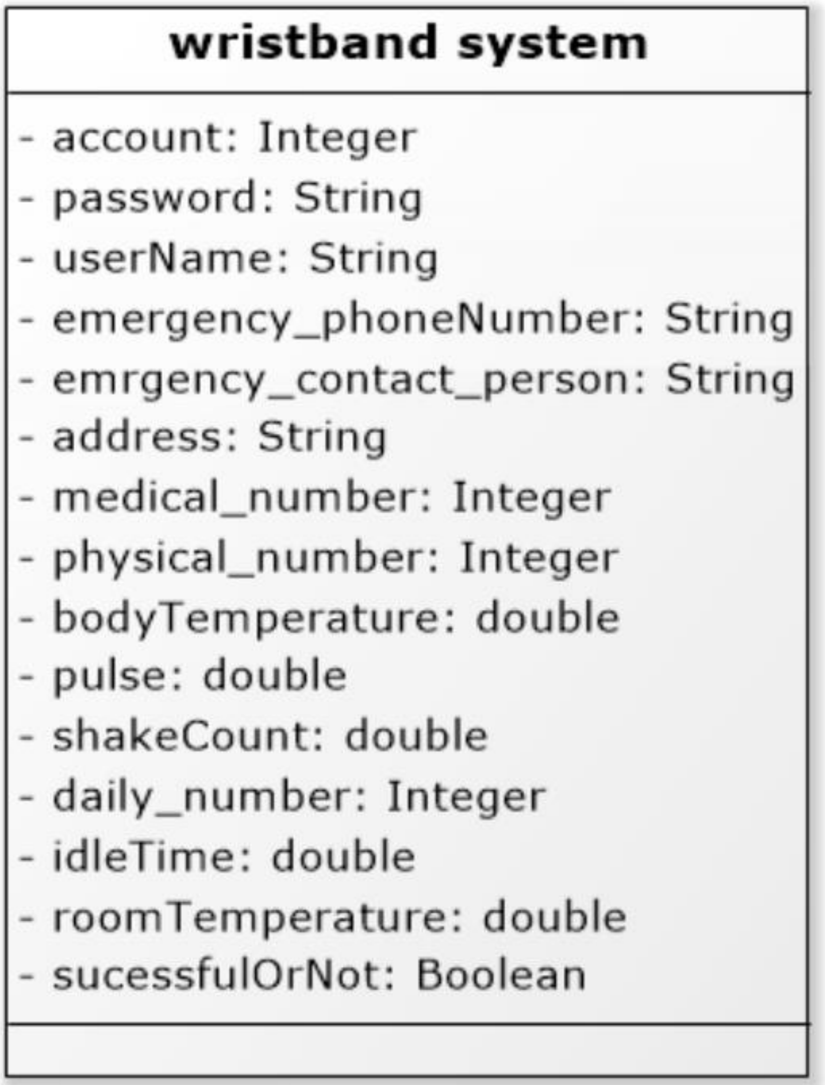
sa medical history table
medical_number : int(10)
account : int(11)
physical_number : int(10)
daily_number : int(10)

sa physical state table
physicalNumber : int(11)
bodyTemperature : double
pulse : double
shakeCount : int(11)

sa daily state table
dailyNumber : int(11)
idleTime : double
roomTemperature : double

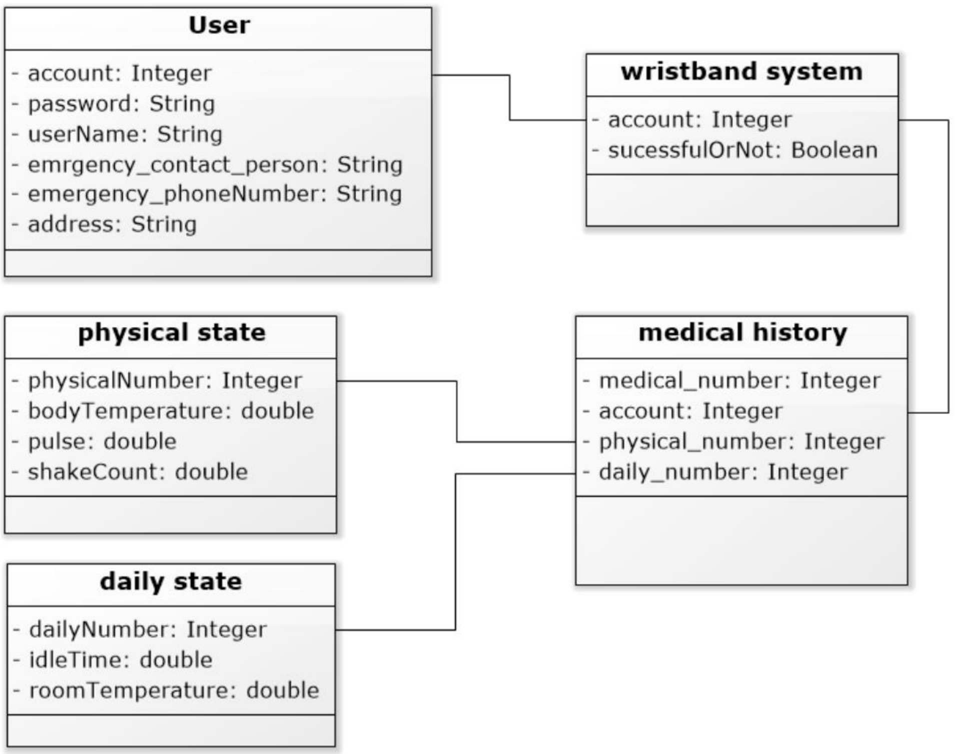
9) Map problem domain objects of your project to a RDBMS format, and use an example to describe the steps of normalization and apply it to the class diagram in third normal form.

Class Diagram

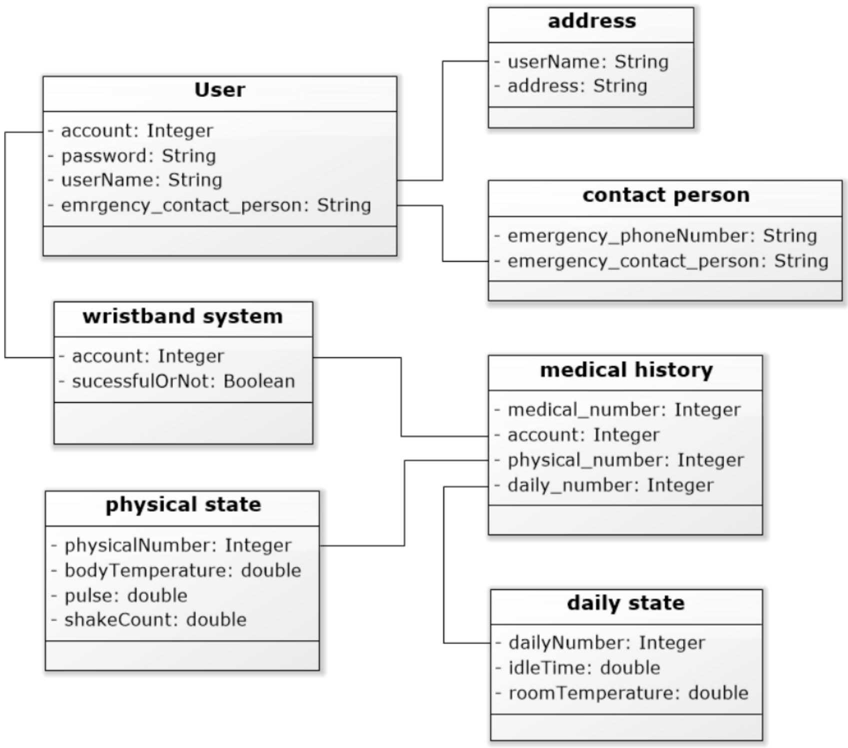
First Normal Form Class Diagram


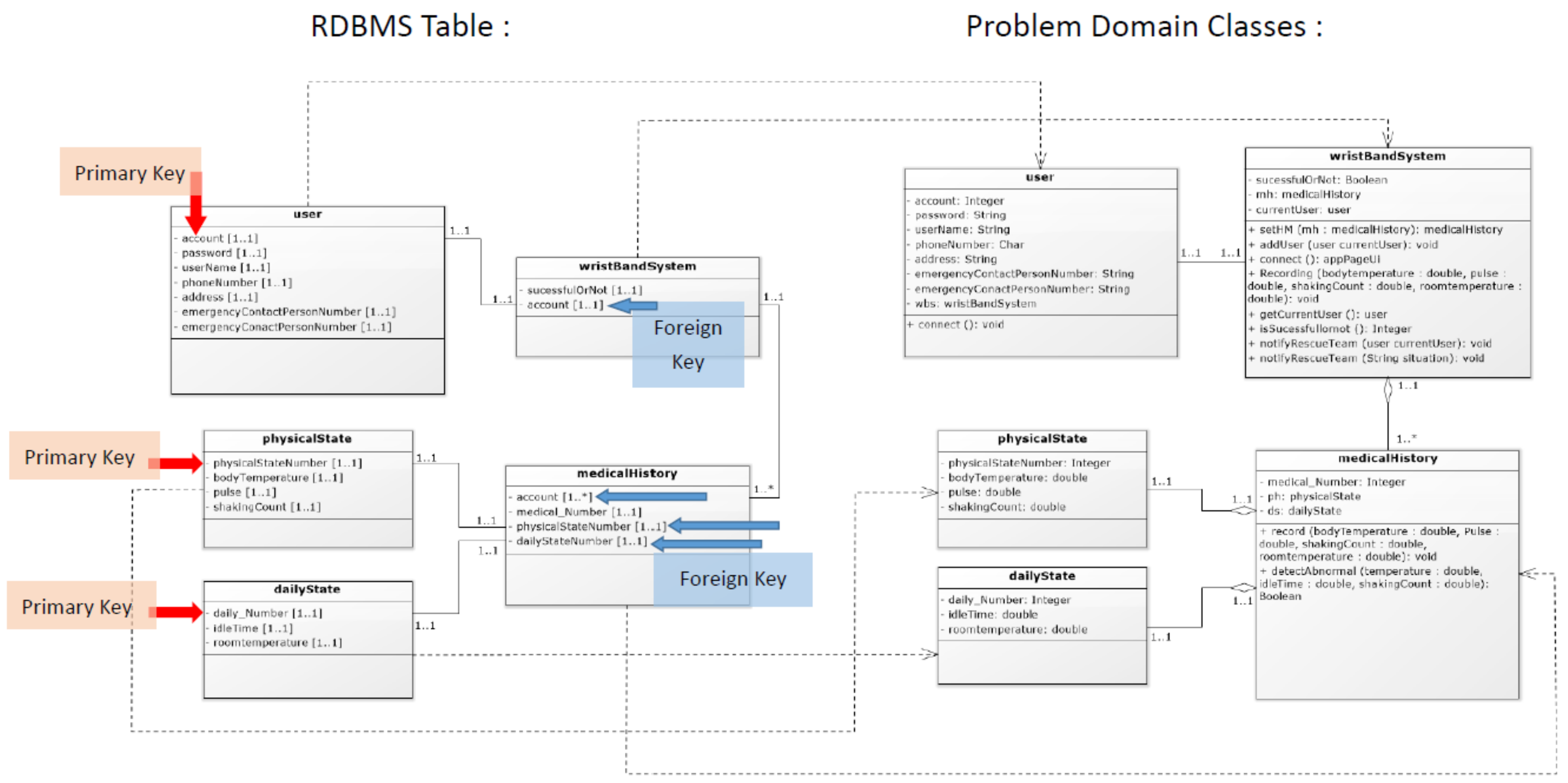
wristband system
- account: Integer
- password: String
- userName: String
- emergency_phoneNumber: String
- emrgency_contact_person: String
- address: String
- medical_number: Integer
- physical_number: Integer
- bodyTemperature: double
- pulse: double
- shakeCount: double
- daily_number: Integer
- idleTime: double
- roomTemperature: double
- sucessfulOrNot: Boolean

Second Normal Form Class Diagram



Third Normal Form Class Diagram




RDBMS Table :
Problem Domain Classes :
Primary Key
Primary Key
Primary Key
Foreign Key
Foreign Key
user
- account [1..1]
- password [1..1]
- userName [1..1]
- phoneNumber [1..1]
- address [1..1]
- emergencyContactPersonNumber [1..1]
- emergencyConactPersonNumber [1..1]
wristBandSystem
- sucessfulOrNot [1..1]
- account [1..1]
physicalState
- physicalStateNumber [1..1]
- bodyTemperature [1..1]
- pulse [1..1]
- shakingCount [1..1]
dailyState
- daily_Number [1..1]
- idleTime [1..1]
- roomtemperature [1..1]
medicalHistory
- account [1..*]
- medical_Number [1..1]
- physicalStateNumber [1..1]
- dailyStateNumber [1..1]
1..1
1..1
1..1
1..*
1..1
1..1
1..1
1..1
user
- account: Integer
- password: String
- userName: String
- phoneNumber: Char
- address: String
- emergencyContactPersonNumber: String
- emergencyConactPersonNumber: String
- wbs: wristBandSystem
+ connect (): void
wristBandSystem
- sucessfulOrNot: Boolean
- mh: medicalHistory
- currentUser: user
+ setHM (mh : medicalHistory): medicalHistory
+ addUser (user currentUser): void
+ connect (): appPageUI
+ Recording (bodytemperature : double, pulse : double, shakingCount : double, roomtemperature : double): void
+ getCurrentUser (): user
+ isSucessfullornot (): Integer
+ notifyRescueTeam (user currentUser): void
+ notifyRescueTeam (String situation): void
physicalState
- physicalStateNumber: Integer
- bodyTemperature: double
- pulse: double
- shakingCount: double
dailyState
- daily_Number: Integer
- idleTime: double
- roomtemperature: double
medicalHistory
- medical_Number: Integer
- ph: physicalState
- ds: dailyState
+ record (bodyTemperature : double, Pulse : double, shakingCount : double, roomtemperature : double): void
+ detectAbnormal (temperature : double, idleTime : double, shakingCount : double): Boolean
1..1
1..1
1..1
1..*
1..1
1..1
1..1
1..1

Mapping

Zero Normal Form

user						medical history			physical state			daily state		wristband system
account	password	userName	emergency_phoneNumber	emergency_contact_person	address	medical_number	daily_number	physical_number	bodyTemperature	pulse	shakeCount	idleTime	roomTemperature	sucessfulOrnot
477	678	Dan	0984154384	Brown	addr009	8397	956	396	35.9	69	7	53	34	1
477	678	Dan	0984154385	Brown	addr010	9160	690	450	35.1	63	8	76	18	1
477	678	Dan	0984154386	Brown	addr011	9620	524	567	35.5	88	2	55	20	1
477	678	Dan	0984154382	Brown	addr007	7128	439	716	35.8	120	8	89	28	1
477	678	Dan	0984154383	Brown	addr008	7397	665	779	36.9	78	2	83	25	1
477	678	Dan	0984154381	Brown	addr006	1361	757	839	36.2	91	7	71	18	1
498	234	Kurumi	095116316	Luke	addr004	7841	346	250	36.4	70	9	8	25	1
498	234	Kurumi	095116314	Luke	addr002	3245	250	645	36.2	78	2	47	19	1
498	234	Kurumi	095116315	Luke	addr003	7271	485	812	35.8	104	3	57	30	1
540	345	Jerry	0951516514847	Leon	addr003	2018	702	235	36.5	73	5	96	31	1
645	567	Kevin	092471157	Kurn	addr010	6525	355	103	35.9	85	8	6	16	1
645	567	Kevin	092471158	Kurn	addr011	8223	911	246	35.3	71	1	47	33	1
645	567	Kevin	092471155	Kurn	addr008	4772	934	462	37	64	5	13	33	1
645	567	Kevin	092471156	Kurn	addr009	6097	955	475	36.9	109	2	45	19	1
645	567	Kevin	092471154	Kurn	addr007	3217	354	499	36.2	106	3	29	20	1
645	567	Kevin	092471152	Kurn	addr005	1029	422	533	35.9	84	2	66	15	1
645	567	Kevin	092471153	Kurn	addr006	1849	943	705	36.3	111	4	76	18	1
722	123	Leon	09515131	Oscar	addr001	5935	139	822	35.8	108	3	30	31	1
761	789	Ben	098713158	Berry	addr008	4070	268	376	36	68	9	53	35	1
761	789	Ben	098713157	Berry	addr007	1079	419	782	36.1	68	4	15	34	1
761	789	Ben	098713159	Berry	addr009	6306	105	846	35.7	97	5	20	29	1
782	890	Ian	09851496	Lai	addr009	4455	411	630	35.4	117	7	97	28	1
782	890	Ian	09851495	Lai	addr008	1819	150	939	35.6	74	2	82	20	1
833	456	Justin	09515876	Bieber	addr004	1253	235	599	36	115	3	36	25	1
833	456	Justin	09515878	Bieber	addr006	5493	286	628	37	78	10	42	17	1
833	456	Justin	09515877	Bieber	addr005	3273	183	897	37	83	6	32	31	1

First Normal Form

wrisbandsystem table

account	password	userName	emergency_phoneNumber	emergency_contact_person	address	medical_number	daily_number	physical_number	idleTime	roomTemperature	yTempera	pulse	shakeCount	cessfulOrr
477	678	Dan	0984154384	Brown	addr009	8397	956	396	53	34	35.9	69	7	1
477	678	Dan	0984154385	Brown	addr010	9160	690	450	76	18	35.1	63	8	1
477	678	Dan	0984154386	Brown	addr011	9620	524	567	55	20	35.5	88	2	1
477	678	Dan	0984154382	Brown	addr007	7128	439	716	89	28	35.8	120	8	1
477	678	Dan	0984154383	Brown	addr008	7397	665	779	83	25	36.9	78	2	1
477	678	Dan	0984154381	Brown	addr006	1361	757	839	71	18	36.2	91	7	1
498	234	Kurumi	095116316	Luke	addr004	7841	346	250	8	25	36.4	70	9	1
498	234	Kurumi	095116314	Luke	addr002	3245	250	645	47	19	36.2	78	2	1
498	234	Kurumi	095116315	Luke	addr003	7271	485	812	57	30	35.8	104	3	1
540	345	Jerry	0951516514847	Leon	addr003	2018	702	235	96	31	36.5	73	5	1
645	567	Kevin	092471157	Kurn	addr010	6525	355	103	6	16	35.9	85	8	1
645	567	Kevin	092471158	Kurn	addr011	8223	911	246	47	33	35.3	71	1	1
645	567	Kevin	092471155	Kurn	addr008	4772	934	462	13	33	37	64	5	1
645	567	Kevin	092471156	Kurn	addr009	6097	955	475	45	19	36.9	109	2	1
645	567	Kevin	092471154	Kurn	addr007	3217	354	499	29	20	36.2	106	3	1
645	567	Kevin	092471152	Kurn	addr005	1029	422	533	66	15	35.9	84	2	1
645	567	Kevin	092471153	Kurn	addr006	1849	943	705	76	18	36.3	111	4	1
722	123	Leon	09515131	Oscar	addr001	5935	139	822	30	31	35.8	108	3	1
761	789	Ben	098713158	Berry	addr008	4070	268	376	53	35	36	68	9	1
761	789	Ben	098713157	Berry	addr007	1079	419	782	15	34	36.1	68	4	1
761	789	Ben	098713159	Berry	addr009	6306	105	846	20	29	35.7	97	5	1
782	890	Ian	09851496	Lai	addr009	4455	411	630	97	28	35.4	117	7	1
782	890	Ian	09851495	Lai	addr008	1819	150	939	82	20	35.6	74	2	1
833	456	Justin	09515876	Bieber	addr004	1253	235	599	36	25	36	115	3	1
833	456	Justin	09515878	Bieber	addr006	5493	286	628	42	17	37	78	10	1
833	456	Justin	09515877	Bieber	addr005	3273	183	897	32	31	37	83	6	1

Second Normal Form

user table

account	password	userName	emergency_phoneNumber	emergency_contact_person	address
477	678	Dan	0984154381	Brown	addr006
498	234	Kurumi	095116314	Luke	addr002
540	345	Jerry	0951516514847	Leon	addr003
645	567	Kevin	092471152	Kurn	addr005
722	123	Leon	09515131	Oscar	addr001
761	789	Ben	098713157	Berry	addr007
782	890	Ian	09851495	Lai	addr008
833	456	Justin	09515876	Bieber	addr004

wristband system table

account	sucessfulOrnot
477	1
498	1
540	1
645	1
722	1
761	1
782	1
833	1

medical history table

medical_number	account	physical_number	daily_number
1361	477	839	757
7128	477	716	439
7397	477	779	665
8397	477	396	956
9160	477	450	690
9620	477	567	524
3245	498	645	250
7271	498	812	485
7841	498	250	346
2018	540	235	702
1029	645	533	422
1849	645	705	943
3217	645	499	354
4772	645	462	934
6097	645	475	955
6525	645	103	355
8223	645	246	911
5935	722	822	139
1079	761	782	419
4070	761	376	268
6306	761	846	105
1819	782	939	150
4455	782	630	411
1253	833	599	235
3273	833	897	183
5493	833	628	286

physical number table

physicalNumber	bodyTemperature	pulse	shakeCount
103	35.9	69	7
235	35.1	63	8
246	35.5	88	2
250	35.8	120	8
376	36.9	78	2
396	36.2	91	7
450	36.4	70	9
462	36.2	78	2
475	35.8	104	3
499	36.5	73	5
533	35.9	85	8
567	35.3	71	1
599	37	64	5
628	36.9	109	2
630	36.2	106	3
645	35.9	84	2
705	36.3	111	4
716	35.8	108	3
779	36	68	9
782	36.1	68	4
812	35.7	97	5
822	35.4	117	7
839	35.6	74	2
846	36	115	3
897	37	78	10
939	37	83	6

daily number table

dailyNumber	idleTime	roomTemperature
105	53	34
139	76	18
150	55	20
183	89	28
235	83	25
250	71	18
268	8	25
286	47	19
346	57	30
354	96	31
355	6	16
411	47	33
419	13	33
422	45	19
439	29	20
485	66	15
524	76	18
665	30	31
690	53	35
702	15	34
757	20	29
911	97	28
934	82	20
943	36	25
955	42	17
956	32	31

Third Normal Form

contact person table

emergency_phoneNumber	emergency_contact_person
0984154381	Brown
095116314	Luke
0951516514847	Leon
092471152	Kurn
09515131	Oscar
098713157	Berry
09851495	Lai
09515876	Bieber

user table

account	password	userName	emergency_contact_person
477	678	Dan	Brown
498	234	Kurumi	Luke
540	345	Jerry	Leon
645	567	Kevin	Kurn
722	123	Leon	Oscar
761	789	Ben	Berry
782	890	Ian	Lai
833	456	Justin	Bieber

wristband system table

account	sucessfulOrnot
477	1
498	1
540	1
645	1
722	1
761	1
782	1
833	1

daily state table

dailyNumber	idleTime	roomTemperature
105	53	34
139	76	18
150	55	20
183	89	28
235	83	25
250	71	18
268	8	25
286	47	19
346	57	30
354	96	31
355	6	16
411	47	33
419	13	33
422	45	19
439	29	20
485	66	15
524	76	18
665	30	31
690	53	35
702	15	34
757	20	29
911	97	28
934	82	20
943	36	25
955	42	17
956	32	31

medical history table

medical_number	account	physical_number	daily_number
1361	477	839	757
7128	477	716	439
7397	477	779	665
8397	477	396	956
9160	477	450	690
9620	477	567	524
3245	498	645	250
7271	498	812	485
7841	498	250	346
2018	540	235	702
1029	645	533	422
1849	645	705	943
3217	645	499	354
4772	645	462	934
6097	645	475	955
6525	645	103	355
8223	645	246	911
5935	722	822	139
1079	761	782	419
4070	761	376	268
6306	761	846	105
1819	782	939	150
4455	782	630	411
1253	833	599	235
3273	833	897	183
5493	833	628	286

physical state table

physicalNumber	bodyTemperature	pulse	shakeCount
103	35.9	69	7
235	35.1	63	8
246	35.5	88	2
250	35.8	120	8
376	36.9	78	2
396	36.2	91	7
450	36.4	70	9
462	36.2	78	2
475	35.8	104	3
499	36.5	73	5
533	35.9	85	8
567	35.3	71	1
599	37	64	5
628	36.9	109	2
630	36.2	106	3
645	35.9	84	2
705	36.3	111	4
716	35.8	108	3
779	36	68	9
782	36.1	68	4
812	35.7	97	5
822	35.4	117	7
839	35.6	74	2
846	36	115	3
897	37	78	10
939	37	83	6

address table

userName	address
Dan	addr006
Kurumi	addr002
Jerry	addr003
Kevin	addr005
Leon	addr001
Ben	addr007
Ian	addr008
Justin	addr004

Participate In Assignments

ID	Name	Participate	Responsibility
B10423002	Leon	100%	3) 4) 7) 8) check file
B10423003	Kurumi	100%	1) word check file
B10423009	Jerry	100%	Java Code 5) 9) check file
B10423015	Justin	100%	5) 6) 9) check file
B10423032	Kevin	100%	Java Code SQL Code 2) 3) 4) 6) check file
B10423041	Dan	100%	2) word check file
B10423045	Rong	100%	3) 4) 5) Activity Diagram 7) 8) check file
W10423301	Ben	0%	
A10523050	Ian	0%	

Java code

class DBserverListener

```
//use for communicating with MySQL

import com.mysql.jdbc.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;



public class DBserverListener {
    private Connection conn;
    private Statement stmt = null;
    private ResultSet rs = null;

     //insert user data
    public void    sendDataToMySQLServer(user currentUser){
        String driver = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/sa";
        String user = "root";
        String password = "12345";
        try {
            Class.forName(driver);
            conn = (Connection) DriverManager.getConnection(url,user, password);
        }
        catch(ClassNotFoundException e) {
            System.out.println("can't find driver");
            e.printStackTrace();
        }
        catch(SQLException e) {
            e.printStackTrace();
        }

        int account = currentUser.getAccount();
        String userpassword = currentUser.getPassword();
        String username = currentUser.getUserName();
        String contact_phone = currentUser.getemergencyContactPersonNumber();
        String contact_person = currentUser.getEmergencyContactPerson();
        String address = currentUser.getAddress();

this.insertuserData(account,userpassword,username,contact_phone,contact_person,ad
dress);

         try {
            conn.close();    //close SQL
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }



    //insert recording data
    public   void sendRecordingDataToMySQLServer(wristBandSystem wrist){
        String driver = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/sa";
        String user = "root";
        String password = "12345";
        try {
            Class.forName(driver);
            conn = (Connection) DriverManager.getConnection(url,user, password);
        }
        catch(ClassNotFoundException e) {
            System.out.println("can't find driver");
            e.printStackTrace();
        }
        catch(SQLException e) {
            e.printStackTrace();
        }
        user currentUser = wrist.getCurrentUser();
        medicalHistory medicalHistory = wrist.getMh();
        dailyState DailyState = medicalHistory.getDailyState();
        physicalState PhysicalState = medicalHistory.getPhysicalState();
        //insert daily data
        int dailyNumber = DailyState.getdailyStateNumber();
        double idleTime =DailyState.getIdleTime();
        double roomTemperature = DailyState.getRoomtemperature();
        this.insertDailyStateData(dailyNumber,idleTime,roomTemperature);
        //insert physical data
        int physicalNumber =PhysicalState.getphysicalStateNumber();
        double bodyTemperature = PhysicalState.getTemperature();
        double pulse = PhysicalState.getPulse();
        double shakeCount = PhysicalState.getShakingCount();
        this.insertPhysicalStateData(physicalNumber, bodyTemperature, pulse,
shakeCount);

        //insert medical data
        int medicalNumber = medicalHistory.getMedical_number();
        int account = currentUser.getAccount();

this.insertMedicalStateData(medicalNumber,account,physicalNumber,dailyNumber);
        try {
            conn.close();    //close SQL
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    //send system data
    public void sendSystemDatatoMySQLServer(wristBandSystem wrist){

        String driver = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/sa";
        String user = "root";
        String password = "12345";
        try {
            Class.forName(driver);
            conn = (Connection) DriverManager.getConnection(url,user, password);
        }
        catch(ClassNotFoundException e) {
            System.out.println("can't find driver");
            e.printStackTrace();
        }
        catch(SQLException e) {
            e.printStackTrace();
        }
        user currentUser = wrist.getCurrentUser();

        //insert system data
        int sucessfulornot = wrist.isSucessfullornot();
        int account = currentUser.getAccount();
        this.insertSystemStateData(sucessfulornot,account);

        try {
            conn.close();    //close SQL
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }



    //insert data in DailyState
    public void insertDailyStateData(int Number,double Time,double Temperature){
        try {
        PreparedStatement preparedStatement = null;
        String insertTableSQL = "INSERT INTO `daily state table` "
                    + "(dailyNumber, idleTime, roomTemperature) VALUES"
                    + "(?,?,?)";

                preparedStatement = conn.prepareStatement(insertTableSQL);

                preparedStatement.setInt(1, Number);
                preparedStatement.setDouble(2, Time);
                preparedStatement.setDouble(3, Temperature);

                // execute insert SQL stetement
                preparedStatement.executeUpdate();

            //conn.close();
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        }

        //insert data in PhysicalState
        public void insertPhysicalStateData(int physicalNumber,double
bodyTemperature,double pulse,double shakeCount){
            try{
            PreparedStatement preparedStatement = null;
            String insertTableSQL = "INSERT INTO `physical state table` "
                        + "(physicalNumber, bodyTemperature, pulse,shakeCount)
VALUES"
                        + "(?,?,?,?)";

                    preparedStatement = conn.prepareStatement(insertTableSQL);

                    preparedStatement.setInt(1, physicalNumber);
                    preparedStatement.setDouble(2, bodyTemperature);
                    preparedStatement.setDouble(3, pulse);
                            preparedStatement.setDouble(4, shakeCount);
                    // execute insert SQL stetement
                    preparedStatement.executeUpdate();

                //conn.close();
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        }

        public void insertuserData(int account,String userpassword,String username,String
emergency_phoneNumber,String emergency_contact_person,String address){
            try{
            PreparedStatement preparedStatement = null;
            String insertTableSQL = "INSERT INTO `user` "
                        + "(account, password,
userName,emergency_phoneNumber,emergency_contact_person,address) VALUES"
                        + "(?,?,?,?,?,?)";
            preparedStatement = conn.prepareStatement(insertTableSQL);

            preparedStatement.setInt(1, account);
            preparedStatement.setString(2, userpassword);
            preparedStatement.setString(3, username);
                    preparedStatement.setString(4, emergency_phoneNumber);
                    preparedStatement.setString(5, emergency_contact_person);
                    preparedStatement.setString(6, address);
            // execute insert SQL stetement
            preparedStatement.executeUpdate();

        //conn.close();
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
}

private void insertSystemStateData(int sucessfulornot, int account) {
    try{
    PreparedStatement preparedStatement = null;
    String insertTableSQL = "INSERT INTO `wristband system table` "
            + "(sucessfulornot, account) VALUES"
            + "(?,?)";

            preparedStatement = conn.prepareStatement(insertTableSQL);

            preparedStatement.setInt(1, sucessfulornot);
            preparedStatement.setInt(2, account);

            // execute insert SQL stetement
            preparedStatement.executeUpdate();

        //conn.close();
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
}
```

```
    private void insertMedicalStateData(int medicalNumber,int account ,int
physicalNumber, int dailyNumber) {
        try{
        PreparedStatement preparedStatement = null;
        String insertTableSQL = "INSERT INTO `medical history table` "
                    + "(medical_number,account,physical_number,daily_number)
VALUES"
                    + "(?,?,?,?)";

                preparedStatement = conn.prepareStatement(insertTableSQL);

                preparedStatement.setInt(1, medicalNumber);
                        preparedStatement.setInt(2, account);
                        preparedStatement.setInt(3, physicalNumber);
                        preparedStatement.setInt(4, dailyNumber);
                // execute insert SQL stetement
                preparedStatement.executeUpdate();

            conn.close();
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

class GPS

```
public class GPS {
    public static String locateCurrentPosition(){
        return "Position is in Yuntech";
    }
}
```

interface RescueTeamServer

```
public interface RescueTeamServer {
    boolean checkMsg();
}
```

class TeamWork2

```
import java.util.Scanner;

/**
 *
 * @author User
 */
public class TeamWork2 {
    //DB test
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        wristBandSystem wbs = new wristBandSystem();
        user currentUser = new user(wbs);
        wbs.addUser(currentUser);
       // String userId = "yuntech",   password = "12345";
        appPageUi appui = wbs.connect();
        wristBandGUI.displaMessage("Please select login(1) or sign up(2)");
        String selection = scanner.next();

        Loop:outer:
        while(true){
        switch(selection){
            case"1":
                appui.login(currentUser);
                break outer;
            case"2":
                appui.signup(currentUser);
                break outer;
            default:
                wristBandGUI.displaMessage("Input error, please input login(1) or
sign up(2) again");
                selection = scanner.next();

               break;
        }
```

```
            }

            // start recording
            boolean normalState = true;
            //currentUser.pressEmergencyButton();//press button
            while(normalState == true){
                normalState = wbs.Recording(37, 120, 1, 30, 50);//test manually
                //normalState = wbs.Recording(Math.random()*(45-30+1)+30,
Math.random()*(120-80+1)+80, Math.random()*3+1,
Math.random()*(200-25+1)+25, Math.random()*(150-0+1)+0);//send
bodyTemperature,pulse,shakingCount,roomtemperature idleTime
            }
        }
}
```

class appPageUi

```
import java.util.Scanner;

/**
 *
 * @author User
 */
public class appPageUi {
   DBserverListener DBserverListener = new DBserverListener();
    Scanner scanner = new Scanner(System.in);
    //login
    public void login(user currentUser){
        String userId,password;
        //this.currentUser = currentUser;
        scanner = new Scanner(System.in);
                  wristBandGUI.displaMessage("Please log in   account\n");
                  wristBandGUI.displaMessage("Please input account:");
                  userId = scanner.next();
                  wristBandGUI.displaMessage("Please input password:");
                 password = scanner.next();
                 boolean result = currentUser.confirm(userId, password);
                if(result){
                    currentUser.connect();
//connect to device
                    return;
                }
                else{
                    this.handleLoginError(currentUser);
                }

    }
    //handleLoginError
    void handleLoginError(user currentUser){
        while(true){
            wristBandGUI.displaMessage("Account or password is error, please input
account again(1) or sign up(2)");
            String select = scanner.next();
            switch(select){
                case "1":
                    this.login(currentUser);
                    return;
                case "2":
                    this.signup(currentUser);
                    return;
                default:
                    wristBandGUI.displaMessage("Please input 1 or 2");
                    break;
            }
        }
    }
    //sign up
    public void signup(user currentUser){
        this.fillpersonalInformation(currentUser);
        this.login(currentUser);
    }
    //fill personal info
    public void fillpersonalInformation(user currentUser){
        String userId,password,emergencyContactPerson,addressNumber,address;

        wristBandGUI.displaMessage("First use, please signup account\n");
        wristBandGUI.displaMessage("Please input account:");
        userId = scanner.next();
        wristBandGUI.displaMessage("Please input password:");
        password = scanner.next();
         wristBandGUI.displaMessage("Please input emergencyContactPerson:");
        emergencyContactPerson = scanner.next();
        wristBandGUI.displaMessage("Please input emergencyContactPerson phone
Number:");
        addressNumber = scanner.next();
        wristBandGUI.displaMessage("Please input address:");
        address = scanner.next();
        currentUser.record(userId,
password,emergencyContactPerson,addressNumber,address);      //record new data
in userDB
        wristBandGUI.displaMessage("Signup sucessfully\n") ;
        DBserverListener.sendDataToMySQLServer(currentUser);
    }
}
```

class dailyState

```
public class dailyState {
    private int dailyStateNumber ;//primary key initial = 0
    private double IdleTime = 0;//idleTime store
    private double Roomtemperature;//roomtemperature store




    //get last
    public int getdailyStateNumber(){
        return dailyStateNumber;
    }

    public double getRoomtemperature() {
        return Roomtemperature;
    }

    public double getIdleTime() {
        return this.IdleTime;
    }
    public void setRoomtemperature(double roomtemperature) {
        this.Roomtemperature = roomtemperature;
    }

    public void setIdleTime(double idleTime) {
        this.IdleTime = idleTime;
    }

    public void setdailyStateNumber(){
        this.dailyStateNumber=   (int)(Math.random()*(1000-100+1)+100);
    }

}
```

class dangerDetermin

```
public class dangerDetermin {
    public String identify(double bodytempature,double idleTime,double
shakeCount,double roomteamerature){
        String situation = new String();
        if(((bodytempature <= 45&&bodytempature >= 30) && shakeCount >=
3)||bodytempature <= 30){
            situation =   "drowning";
        }
        else if(roomteamerature >= 150){
            situation = "firing";
        }
        else if(idleTime >= 100){//>=100hr
            situation = "moutainAccident";
        }
        return situation;
    }
}
```

class emergencyContactPersonServer

```
public class emergencyContactPersonServer implements RescueTeamServer{
    private   String Name = "default";

    public void setName(String Name) {
        this.Name = Name;
    }
    //get a new emergencyContactPersonServer
    public static emergencyContactPersonServer
getemergencyContactPersonServer(){
        emergencyContactPersonServer ecps = new
emergencyContactPersonServer();
        return ecps;
    }
    //overloading checkMsg
    public boolean checkMsg(user currentUser){
        boolean confirm = true;             //select by server
        if(confirm){
            String msg = currentUser.getEmergencyContactPerson() + " confirm";
            wristBandGUI.displaMessage(msg);
            return true;
        }
        else{
            String msg = currentUser.getEmergencyContactPerson() + " doesn't
confirm";
            wristBandGUI.displaMessage(msg);
            return false;
        }
    }
    @Override
    public boolean checkMsg(){
            return false;
    }
}
```

class firefighterServer

```
public class firefighterServer implements RescueTeamServer{
   private final String name = "firefighter";
    @Override//implements checkMsg
    public boolean checkMsg(){
        boolean confirm = true; //select by server
        if(confirm){
            wristBandGUI.displaMessage(this.name+" confirm");
            return true;
        }
         else{
            wristBandGUI.displaMessage(this.name+" doesn't confirm");
            return false;
        }
    }
}
```

class identifyException

```
public class identifyException extends Exception{
    @Override
    public String getMessage(){
       return "Sorry, the system can't identify situation, please use emergency button
to contact";
    }
}
```

class medicalHistory

```
public class medicalHistory {
    private final dailyState dailyState = new dailyState();
    private final physicalState physicalState = new physicalState();
    private int medical_number ;
    private double bodyTemperature;
    private double pulse;
    private double idleTime;
    private double shakingCount;
    private double roomtemperature;

    public dailyState getDailyState() {
        return dailyState;
    }

    public physicalState getPhysicalState() {
        return physicalState;
    }

    public int getMedical_number() {
        return medical_number;
    }

    public double getBodyTemperature() {
        return bodyTemperature;
    }

    public double getShakingCount() {
        return shakingCount;
    }

    public double getRoomtemperature() {
        return roomtemperature;
    }



    public double getPulse() {
        return pulse;
    }

    public double getIdleTime() {
        return idleTime;
    }

    public void setMedical_number() {
        this.medical_number = (int)(Math.random()*(10000-1000+1)+1000);
    }



    public boolean record(double bodyTemperature,double Pulse,double
shakingCount,double roomtemperature,double idleTime){//傳入資料
        //record data if annormal break;
        boolean normalState = true;
        //set info
        this.setMedical_number();
        physicalState.setTemperature(bodyTemperature);    //record
bodyTemperature
        physicalState.setPulse(Pulse);                    //record Pulse
        physicalState.setShakingCount(shakingCount);     //record shakingCount
        dailyState.setRoomtemperature(roomtemperature);//record roomtemperature
        dailyState.setIdleTime(idleTime);                //record idleTime
        dailyState.setdailyStateNumber();                //record dailyState number
        physicalState.setphysicalStateNumber();      //record physicalState number
        wristBandGUI.displaMessage("Tracking your data");
        //get info
        this.bodyTemperature = physicalState.getTemperature();
        this.pulse = physicalState.getPulse();
        this.idleTime = dailyState.getIdleTime();
        this.shakingCount = physicalState.getShakingCount();
        this.roomtemperature = dailyState.getRoomtemperature();
        this.shakingCount = physicalState.getShakingCount();
        System.out.printf("bodyTemperature is %.2f oc\npulse is %.2f
mmHg\nidleTime is %.2f hr\nroomtemperature is %.2foc\nshakingCount is %.2f per
second\n",bodyTemperature,pulse,idleTime,roomtemperature,shakingCount);
        //detectAbnormal

if(detectAbnormal(roomtemperature,idleTime,shakingCount,bodyTemperature)){
            wristBandGUI.displaMessage("Detect abnormal, system will go into
emergency situation~~~");
            normalState = false;//detect set state false
        }else{
            wristBandGUI.displaMessage("Data is normal, keep
tracking~~~\n\n\n");//keep tracking
        }
        return normalState;
    }
    //detectAbnormal function
    private boolean detectAbnormal(double roomTemperature,double
```

```
idleTime,double shakingCount,double bodytemperature){//send
roomTemperature,idleTime,shakingCount

        if((bodytemperature <= 30 && bodytemperature >= 45) || roomTemperature
>= 150 ||idleTime >= 100 ||shakingCount >= 3){
            return true;
        }
        return false;
    }
}
```

class moutainguardServer

```
public class moutainguardServer implements RescueTeamServer{
    private final String name = "moutainguard";
    @Override//implements checkMsg
    public boolean checkMsg(){
        boolean confirm = true;//select by server
        if(confirm){
            wristBandGUI.displaMessage(this.name+" confirm");
            return true;
        }
         else{
            wristBandGUI.displaMessage(this.name+" doesn't confirm");
            return false;
        }
    }
}
```

class physicalState

```
public class physicalState {
    private int physicalStateNumber;//initial key
    private double Temperature ;//Temperature store
    private double Pulse ;        //Pulse store
    private double ShakingCount ;//ShakingCount store

    //get last data
    public int getphysicalStateNumber(){
        return physicalStateNumber;
    }

    public double getShakingCount() {
        return ShakingCount;
    }

    public double getTemperature() {
        return Temperature;
    }

    public double getPulse() {
        return Pulse;
    }
    //add new data
    public void setShakingCount(double shakingCount) {
        this.ShakingCount = shakingCount;
    }

    public void setTemperature(double temperature) {
        this.Temperature = temperature;
    }

    public void setPulse(double pulse) {
        this.Pulse = pulse;
    }

    public void setphysicalStateNumber(){
        this.physicalStateNumber=   (int)(Math.random()*(1000-100+1)+100);
    }
}
```

class rescueTeam

```
public class rescueTeam {
        private RescueTeamServer rescueTeamServer;
        //use flag to discriminate which rescueTeam Server to be assigned, use for auto
         public boolean notifyEmergency(String flag){
              if(flag.equals("waterguard")){
                   rescueTeamServer = new waterguardServer();
              }
              else if(flag.equals("firefighter")){
                   rescueTeamServer = new firefighterServer();
              }
              else if(flag.equals("moutainguard")){
                   rescueTeamServer = new moutainguardServer();
              }
              return rescueTeamServer.checkMsg();
         }
         //overloading notifyEmergency use for manual
          public boolean notifyEmergency(user currentUser){
              rescueTeamServer = currentUser.getEcps();//get
emergencyContactPersonServer
              return
((emergencyContactPersonServer)rescueTeamServer).checkMsg(currentUser);
         }
}
```

class user

```
public class user {
    private wristBandSystem wbs ;
    public   int account = (int)(Math.random()*(1000-100+1)+100);//primary key
    private String userName = "Kevin";
    private String password   = "12345";
    private String emergencyContactPersonNumber = "default";
    private String address = "Dream Mall";
    private String emergencyContactPerson = "default";
    //get last
    public int getAccount() {
        return account;
    }
    public   String getemergencyContactPersonNumber() {
        return emergencyContactPersonNumber;
    }

    public   String getAddress() {
        return address;
    }

    public   void setemergencyContactPersonNumber(String
emergencyContactPersonNumber) {
        this.emergencyContactPersonNumber = emergencyContactPersonNumber;
    }

    public void setAccount(int account) {
        this.account = account;
    }

    public   void setAddress(String address) {
        this.address = address;
    }

    private emergencyContactPersonServer ecps;
    public user(wristBandSystem wbs){
        this.wbs = wbs;
         ecps =
emergencyContactPersonServer.getemergencyContactPersonServer();
    }

    public emergencyContactPersonServer getEcps() {
        return ecps;
    }

    public void setEcps(emergencyContactPersonServer ecps) {
        this.ecps = ecps;
    }

     //connect to this device
    public void connect(){//none
        //cellpone.connect();
        wristBandGUI.displaMessage("Connect system sucessfully");
    }
    //press EmergencyButton over 5 times
    public boolean pressEmergencyButton(user currentUser){
        double count = Math.random()*(5-0+1)+1; //define count
        //if count >= 5 active notify function
        if(count >= 5){
         wristBandGUI.displaMessage("You press emergency button over 5 
times\nThe system will notify your emergency contact person");
         wbs.notifyRescueTeam(currentUser);
         return true;
        }
        else{
            return false;
        }
    }

    public void updateInformation(){ //sync user info

    }



    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }

    public String getEmergencyContactPerson() {
        return emergencyContactPerson;
    }

    public void setEmergencyContactPerson(String emergencyContactPerson) {
        this.emergencyContactPerson = emergencyContactPerson;
        ecps.setName(emergencyContactPerson);
    }

    public void record(String userName,String password,String
emergencyContactPerson,String emergencyContactPersonNumber,String address){
        this.userName = userName;
        this.password = password;
        this.emergencyContactPerson = emergencyContactPerson;
        this.emergencyContactPersonNumber = emergencyContactPersonNumber;
        this.address = address;
    }
    public boolean confirm(String userName,String password){
        if(userName.equals(this.userName) && password.equals(this.password)){
            return true;
        }
        else{
            return false;
        }
    }
}
```

class waterguardServer

```
public class waterguardServer implements RescueTeamServer{
    private final String name = "waterguard";
    @Override//implements checkMsg
    public boolean checkMsg() {
        boolean confirm = true; //select by server
        if(confirm){
            wristBandGUI.displaMessage(this.name+" confirm");
            return true;
        }
        else{
            wristBandGUI.displaMessage(this.name+" doesn't confirm");
            return false;
        }
    }
}
```

class wristBandGUI

```
public class wristBandGUI {
    public static void displaMessage(String msg){
        System.out.println(msg);
    }
}
```

class wristBandSystem

```
public class wristBandSystem {
    //define attribute
    private final medicalHistory mh = new medicalHistory();
    private final appPageUi appui = new appPageUi();
    private final dangerDetermin dangerDetermin = new dangerDetermin();
    private boolean sucessfullornot = false;
    private final rescueTeam rescueTeam = new rescueTeam();
    private user currentUser;
    private DBserverListener DBserverListener = new DBserverListener();
    public void addUser(user currentUser){
        this.currentUser = currentUser;
    }
    //connect
    public appPageUi connect(){ //
        wristBandGUI.displaMessage("System start!!");
        wristBandGUI.displaMessage("please loging!!");
        return appui;
    }

    //start to recording
    public boolean Recording(double bodytemperature,double pulse,double
shakingCount,double roomtemperature,double idleTime){
        boolean normalState = true;//define normal state
        normalState =
mh.record(bodytemperature,pulse,shakingCount,roomtemperature,idleTime);   //send
bodyTemperature,pulse,shakingCount,roomtemperature idleTime
        DBserverListener.sendRecordingDataToMySQLServer(this);//send recording
data

        if(currentUser.pressEmergencyButton(currentUser)){
            DBserverListener.sendSystemDatatoMySQLServer(this);//send system
data to server
            return false;
        }
        if(normalState == true){
            return true;//Date is normal, keep tracking~~
        }else {
        String gps = GPS.locateCurrentPosition();            //locate position
        wristBandGUI.displaMessage(gps);
            try{
                String situation =
dangerDetermin.identify(mh.getBodyTemperature(),mh.getIdleTime(),mh.getShaking
Count(),mh.getRoomtemperature());//identify situation

                String tmp = "Discriminate situation is "+situation;
                wristBandGUI.displaMessage(tmp);
                    if(situation.equals("")){
                        throw (new identifyException());
                    }
                this.notifyRescueTeam(situation);//finish notify
                wristBandGUI.displaMessage("Notify sucessfully");
              }catch(identifyException e){
                  wristBandGUI.displaMessage(e.getMessage());
                  return false;
              }
        normalState   = false;            //return normalState is false
        }
        DBserverListener.sendSystemDatatoMySQLServer(this);//send system data
        return normalState;
    }
    //use for auto
    public void notifyRescueTeam(String situation){
        wristBandGUI.displaMessage("Ready to notify");
            if(situation.equals("drowning")){                          //select which
notify rescue team
            while(sucessfullornot == false){
                String flag = "waterguard";
                sucessfullornot = rescueTeam.notifyEmergency(flag);
            }
        }
        else if(situation.equals("firing")){
            while(sucessfullornot == false){
             String flag = "firefighter";
                sucessfullornot = rescueTeam.notifyEmergency(flag);
            }
        }
        else if(situation.equals("moutainAccident")){
            while(sucessfullornot == false){
             String flag = "moutainguard";
                sucessfullornot = rescueTeam.notifyEmergency(flag);
            }
        }
    }
    //overolading notifyRescueTeam use for manual
    public void notifyRescueTeam(user currentUser){
        wristBandGUI.displaMessage("Ready to notify
"+currentUser.getEmergencyContactPerson()+" person");
            while(sucessfullornot == false){
                sucessfullornot = rescueTeam.notifyEmergency(currentUser);
            }
        wristBandGUI.displaMessage("Notify sucessfully");
    }
    public medicalHistory getMh() {
        return mh;
    }
    public int isSucessfullornot() {
        if(sucessfullornot == true){
            return 1;
        }
        return 0;
    }
    public user getCurrentUser() {
        return currentUser;
    }
}
```

SQL code

```
-- phpMyAdmin SQL Dump
-- version 4.7.4
-- https://www.phpmyadmin.net/
--
-- 主機: 127.0.0.1:3306
-- 產生時間： 2017-12-24 08:43:19
-- 伺服器版本: 5.7.19-log
-- PHP 版本： 5.6.31

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- 資料庫： `sa`
--

-- --------------------------------------------------------

--
-- 資料表結構 `daily state table`
--

DROP TABLE IF EXISTS `daily state table`;
CREATE TABLE IF NOT EXISTS `daily state table` (
  `dailyNumber` int(11) NOT NULL,
  `idleTime` double NOT NULL,
  `roomTemperature` double NOT NULL,
  PRIMARY KEY (`dailyNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- 資料表結構 `medical history table`
--

DROP TABLE IF EXISTS `medical history table`;
CREATE TABLE IF NOT EXISTS `medical history table` (
  `medical_number` int(10) NOT NULL,
  `account` int(11) NOT NULL,
  `physical_number` int(10) NOT NULL,
  `daily_number` int(10) NOT NULL,
  PRIMARY KEY (`medical_number`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- 資料表結構 `physical state table`
--

DROP TABLE IF EXISTS `physical state table`;
CREATE TABLE IF NOT EXISTS `physical state table` (
  `physicalNumber` int(11) NOT NULL,
  `bodyTemperature` double NOT NULL,
  `pulse` double NOT NULL,
  `shakeCount` int(11) NOT NULL,
  PRIMARY KEY (`physicalNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- 資料表結構 `user`
```

```
--

DROP TABLE IF EXISTS `user`;
CREATE TABLE IF NOT EXISTS `user` (
  `account` int(10) NOT NULL,
  `password` text NOT NULL,
  `userName` text NOT NULL,
  `emergency_phoneNumber` text NOT NULL,
  `emergency_contact_person` text NOT NULL,
  `address` text NOT NULL,
  PRIMARY KEY (`account`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- 資料表結構 `wristband system table`
--

DROP TABLE IF EXISTS `wristband system table`;
CREATE TABLE IF NOT EXISTS `wristband system table` (
  `sucessfulOrnot` tinyint(1) NOT NULL,
  `account` int(10) NOT NULL,
  PRIMARY KEY (`account`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
COMMIT;

/*!40101 SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```