```java
package Util;



import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;



public class ExUtil {
    static {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (ClassNotFoundException cne) {
            System.out.println("□□□□ □□ □□");
            cne.printStackTrace();
        }
    }

    public static Connection getConnection() {
        Connection con = null;
        try {
            con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "rok", "1111");
        } catch (SQLException se) {
            System.out.println("DB □□ □□");
            se.printStackTrace();
        }
        return con;
    }
}
```

File - D:\javaworkspace\miniProject\src\Util\BmiUtil.java

```java
package Util;

import ExDAO.ExSQL;
import ExDTO.EBMI;
import ExDTO.EMember;

public class BmiUtil {

    // □□ □□(1□) □□□□ □□□
    public static void del() {
        try {
            Thread.sleep(1000); // 1□ □□ (□□: □□□)
        } catch (InterruptedException ie) {
            Thread.currentThread().interrupt(); // □□□ □□□□ □ □□
        }
    }

    // BMI □□□ □□ □□□
    public void bmicul(String EId) {
        ExSQL sql = new ExSQL();
        EMember member = sql.emList(EId);  // □□ □□□ □□□□□□□□ □□□

        System.out.println("*- BMI □□ -*");

        // □□□ □□, □□, □□ □□□
        String Name = member.getEName();
        float weight = member.getEWeight();
        float height = member.getEHeight();

        // BMI □□ (□□ cm □□□ □□□□□ 10000□□ □□□ □)
        float bmicode = weight / ((height * height) / 10000); // BMI □□ □

        EBMI ebmi = new EBMI(); // BMI □□□ □□□ □□ □□

        ebmi.setEBmiName(Name); // □□ □□□ BMI □□□ □□
        ebmi.setEBmiCode((int) bmicode); // BMI □□ □□□ □□□□ □□ (□□□□ □□)

```

```
        // □□□ □□□ □□ □□□ □□
        System.out.println(
" □□□□□□□□    □□□□□□□□□□□□□□□□□");
        del();
        System.out.println(
"        □□□□□□□□□□□□□□□□□□□      □□□□□□□□ □□□□□□□□□□
□□□□□");
        del();
        System.out.println(
"        □□□□□□□□□□□□□□□□□□□      □□□□□ " + weight +
"□ □□□□□□□□□□□□");
        del();
        System.out.println(
"        □□□□□□□□□   □□□□□□□□      □□□□□□□□□□□□□□□□□□□
");
        del();
        System.out.println("            □□□□" + Name
 + "□□□□□□          □□□    " + height +
"□   □□□□□□□□□□□");
        del();
        System.out.println(
"        □□□□□□□□□□□□□□□□□□□      □□□□□□□□□□□□□□□□□□□
□□□□");
        del();
        System.out.println(
"        □□□□□ □□□□□□□□□□□□□      □□□□□□□" + (int)
bmicode + "□□□□□□□□□□□□□□");
        del();
        System.out.println(
"        □□□□ □□□□□□□□□□□□□□      □□□□□□□□□□□□□□□□□□□
□□□□");
        del();
        System.out.println("        □□□□□□□□□□");
        del();

        BMICategory(bmicode);  // BMI □□□□□ □□□□ □□□
□□
        System.out.println(
"*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*");

        // BMI □□ □□
```

```
62          System.out.println(
    "+----------------+----------------+");
63          System.out.println("|     Scale      |
    BMI        |");
64          System.out.println(
    "+----------------+----------------+");
65          System.out.println("|   □□□          |     ~
    18.5      |");
66          System.out.println("|   □□           |
    18.5~22    |");
67          System.out.println("|   □□□          |     23
    ~25       |");
68          System.out.println("|   □□           |
    26~30      |");
69          System.out.println("|   □□□□         |     30
    ~         |");
70          System.out.println(
    "+----------------+----------------+");
71      }
72
73      // BMI □□ □□ □□□□□ □□□□ □□□
74      public void BMICategory(float bmicode) {
75          if (bmicode < 18.5) {
76              System.out.println("□□□"); // BMI□ 18.5
    □□□ □□
77          } else if (bmicode >= 18.5 && bmicode < 23
    ) {
78              System.out.println("□□"); // BMI□ 18.5
    □□ 23 □□□ □□
79          } else if (bmicode >= 23 && bmicode < 25) {
80              System.out.println("□□□"); // BMI□ 23 □□
     25 □□□ □□
81          } else if (bmicode >= 25 && bmicode < 30) {
82              System.out.println("□□"); // BMI□ 25 □□
    30 □□□ □□
83          } else {
84              System.out.println("□□□□"); // BMI□ 30
    □□□ □□
85          }
86      }
87  }
```

```java
package Util;

import ExDAO.ExSQL;
import ExDTO.EFood;
import ExDTO.ERecords;
import ExDTO.E_Menu;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class MenuUtil {

    ExSQL sql = new ExSQL(); // □□□□□□ □□□ □□ ExSQL □□ □□
    Scanner sc = new Scanner(System.in); // □□□ □□□ □□ □□ Scanner □□ □□

    public void menu(String EId) {
        boolean run = true; // □□□□ □□ □□ □□
        List<EFood> savedFood = new ArrayList<>(); // □□□□ □□□ □□□ □□□□ □□□
        List<EFood> foodList = sql.getfoodlist(); // □□□□□□□□ □□ □□□□ □□□

        while (run) {
            System.out.println("□□ □□□ : □, □□, □□□");
            System.out.println("□□□□ □□□□□: ");
            String food = sc.next(); // □□□□ □□□ □□ □□

            foodList = sql.getfoodlist(); // □□ □□□□ □□ □□
            boolean found = false; // □□ □□ □□ □□

            // □□ □□□□□ □□□□ □□□□ □□ □□
            for (int i = 0; i < foodList.size(); i ++) {
                if (foodList.get(i).getFoName().contains(food)) {
```

```
33                      System.out.println("[" + (i + 1
   ) + "] " + foodList.get(i).getFoName() + " (" +
   foodList.get(i).getFoGrams() + ", " + foodList.get(i
   ).getFoCals() + " kcal)");
34                      found = true; // □□□ □□□□ true□
   □□
35                  }
36              }
37
38              // □□ □□□ □□ □□ □□□ □□
39              if (!found) {
40                  System.out.println("□□ □□□ □□□□.");
41              } else {
42                  System.out.println("□□ □□: ");
43                  int selectedIndex = sc.nextInt(); //
   □□□□ □□□ □□ □□ □□
44
45                  // □□ □□□ □□ □□ □□□ □□ □□□ □□
46                  if (food.equals("□□")) {
47                      if (selectedIndex >= 1 &&
   selectedIndex <= 3) {
48                          EFood selectedFood = foodList
   .get(selectedIndex - 1); // □□□ □□□ □□□□□ □□□
49                          savedFood.add(selectedFood);
   // □□□ □□□ □□ □□□□ □□
50                          System.out.println(
   selectedFood.getFoName() + " □□ □□");
51                      } else {
52                          System.out.println("□□□□ □□□
   □ □□ □□□ 1, 2, 3□□□.");
53                      }
54                  } else if (food.equals("□")) {
55                      if (selectedIndex >= 4 &&
   selectedIndex <= 6) {
56                          EFood selectedFood = foodList
   .get(selectedIndex - 1); // □□□ □□□ □□□□□ □□□
57                          savedFood.add(selectedFood);
   // □□□ □□□ □□ □□□□ □□
58                          System.out.println(
   selectedFood.getFoName() + " □□ □□");
59                      } else {
```

```
60                          System.out.println("□□□ □□□
   □ □□ □□□ 4, 5, 6□□□.");
61                      }
62                  } else if (food.equals("□□□")) {
63                      if (selectedIndex >= 7 &&
   selectedIndex <= 9) {
64                          EFood selectedFood =
   foodList.get(selectedIndex - 1); // □□□ □□□ □□□□□
   □□□
65                          savedFood.add(selectedFood
   ); // □□□ □□□ □□ □□□□ □□
66                          System.out.println(
   selectedFood.getFoName() + " □□ □□");
67                      } else {
68                          System.out.println("□□□□□
   □□□ □ □□ □□□ 7, 8, 9□□□.");
69                      }
70                  } else {
71                      System.out.println("□□□□ □□
   □□□□□.");
72                  }
73              }
74
75              // □□ □□ □□□ □□ □□
76              System.out.println("□□□ □ □□□□□□? (y/n)"
   );
77              String search = sc.next();
78              run = search.equalsIgnoreCase("y"); // '
   y' □□ □ □□, 'n' □□ □ □□
79          }
80
81          // □□□ □□□ □□□□ □□
82          if (savedFood.isEmpty()) {
83              System.out.println("□□□ □□□ □□□□.");
84          } else {
85              System.out.println("□□□ □□");
86              int totalCals = 0; // □ □□□ □□
87
88              System.out.println(
   "*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*");
89              for (EFood food : savedFood) {
```

```
 90                 System.out.println(food.getFoName
    () + "\t" + food.getFoGrams() + "\t" + food.
    getFoCals() + " kcal"); // □□□ □□ □□
 91                 totalCals += food.getFoCals();  // □
     □□□ □□
 92             }
 93             System.out.println("\n□ " + totalCals +
    " kcal"); // □ □□□ □□
 94             System.out.println(
    "*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*");
 95         }
 96
 97         // □□□ □□ □□ □ □□ □□
 98         E_Menu menu = new E_Menu();
 99         double totalProtein = 0;
100         double totalCarbohydrates = 0;
101         double totalFats = 0;
102         int totalCals = 0;
103
104         for (EFood foods : foodList) {
105             totalProtein += foods.getFoProtein();
    // □□□ □□
106             totalCarbohydrates += foods.
    getFoCarbohydrates(); // □□□□ □□
107             totalFats += foods.getFoFats(); // □□ □□
108             totalCals += foods.getFoCals(); // □□□
    □□
109         }
110
111         // □□ □□□ □□ SQL □□□ □□
112         int FoNum = 1;
113         int efnum = sql.getFoodCals(FoNum); // □□□
    □□ □ □□□ □□□□
114         menu.setEMNUM(sql.menuNum()); // □□ □□ □□
115         menu.setEMID(EId); // □□□ ID □□
116         menu.setECalories(totalCals); // □ □□□ □□
117         menu.setEProtein((int) totalProtein); // □
    □□□ □□
118         menu.setECarbohydrates((int)
    totalCarbohydrates); // □ □□□□ □□
119         menu.setEFats((int) totalFats); // □ □□ □□
```

```
120             menu.setEfNum(efnum); // □□□ □□ □□ □□ □□
121             sql.insertMenu(menu); // □□□ □□□□□□□ □□
122 
123             // □□ □□□ □□□ □□□□ □□□□ □□
124             List<ERecords> recordList = sql.recordList(
    EId); // □□ □□ □□□□
125             if (recordList == null || recordList.isEmpty
    ()) {
126                 System.out.println("□□ □□□ □□□□.");
127                 return; // □□ □□□ □□□ □□
128             }
129 
130             System.out.println("□□ □□□ □□□□□: ");
131             int exChoose = sc.nextInt() - 1; // □□□□ □□□
     □□ □□ □□
132 
133             // □□□ □□ □□□ □□
134             if (exChoose >= 0 && exChoose < recordList.
    size()) {
135                 ERecords selectedExercise = recordList.
    get(exChoose); // □□□ □□ □□
136                 double caloriesBurned = selectedExercise
    .getECalolies(); // □□ □□□ □□□□
137                 String exerciseDate = selectedExercise.
    getEDate(); // □□ □□ □□ □□□□
138                 menu.setEMDate(exerciseDate); // □□□ □□
    □□
139 
140                 // □□□ □□ □□□□□ □ □□□ □□□□ □□□□ □□
141                 List<EFood> foodCalsList = savedFood;
    // □□□ □□ □□□
142                 if (foodCalsList != null) {
143                     foodCalsList.sort((a, b) -> b.
    getFoCals() - a.getFoCals()); // □□ □□□□ □□□ □□□□□□
    □□
144                     System.out.println("□ □□ □□□: " +
    caloriesBurned + " kcal");
145 
146                     // □□□ □□□□□ □□□ □ □ □□ □ □□□ □□ □
    □□
147                     for (EFood foods : foodCalsList) {
```

```
148                     int foodCalories = foods.
    getFoCals(); // □□□ □□□
149                     if (foodCalories > 0) {
150                         int foodCount = (int) (
    caloriesBurned / foodCalories); // □□□ □□ □□
151                         System.out.println(foods.
    getFoName() + " " + foodCount + "□");
152                     }
153                 }
154             } else {
155                 System.out.println("□□ □□□ □□□□ □
    □□□□□□.");
156                 }
157         } else {
158             System.out.println("□□□ □□ □□□ □□□□
    □□□□□.");
159         }
160     }
161 }
```

File - D:\javaworkspace\miniProject\src\Util\GoalsUtil.java

```java
package Util;

import ExDAO.Calender;
import ExDAO.ExSQL;
import ExDTO.EGoals;
import ExDTO.ExTypes;

import java.util.List;
import java.util.Scanner;

public class GoalsUtil {
    ExSQL sql = new ExSQL(); // □□□□□□ □□□ □□ ExSQL □□ □□
    Scanner sc = new Scanner(System.in); // □□□ □□□ □□ Scanner □□ □□

    // □□ □□ □□□
    public void goal(EGoals goals, String EId) {

        // □□ □□ □□ (□□□□□□□□ □□ □□□ □□□ □□)
        goals.setEGNum(sql.goalsNum());
        goals.setEGId(EId); // □□ □□□ □□

        // □□ □□ □□
        System.out.println("□□ □□□ □□□□□");
        // □□ □□ □□□ □□□□□□□□ □□□□
        List<ExTypes> exNameList = sql.getExName();
        // □□□□□□ □□ □□□ □□□□
        ExTypes selectedExercise = selectExerciseName(exNameList);

        // □□ □□□ □□□□□□ □□
        if (selectedExercise != null) {
            // □□□ □□ □□ □ □□□ □□□ □□
            goals.setECNUM(selectedExercise.getEXNUM());
            goals.setETEx(selectedExercise.getExName());
        } else if (exNameList == null || exNameList.isEmpty()) {
            // □□ □□□ □□ □□ □□ □□
```

```
36              System.out.println("□□ □□□ □□□ □ □□□□.
   □□□ □□ □□□□.");
37              return; // □□
38          }
39
40          // □□ □□ □□ □□
41          System.out.println("□□ □□ □□□ □□□□□ (□) >>");
42          int goalTime = sc.nextInt(); // □□□□□□ □□ □□□
     □□□□
43          goals.setETExTime(goalTime); // □□ □□ □□□ □□
44
45          // □□ □□ (□□ □□□ □□□ □□ □□ □□)
46          Calender.calendar(0);
47          System.out.print("□□ □□□□ □□□□□  >> ");
48          String startDate = sc.next();  // □□ □□□ □□ (
   □□□ □□□□)
49
50          // □□ □□□ □□
51          System.out.print("□□ □□□□ □□□□□ >> ");
52          String endDate = sc.next(); // □□ □□□ □□
53
54          // □□ □□ □□ (9□□ □□)
55          String day = "2024-09-"; // □□ □□□ □□ 9□□ □□
56          // □□□□ □□□ □□□ □□ □□□□ □□
57          String sdayStart = day + String.format("%02d"
   , Integer.parseInt(startDate)); // □□ □□□
58          String sdayEnd = day + String.format("%02d",
   Integer.parseInt(endDate)); // □□ □□□
59          goals.setEEDate(sdayEnd);  // □□ □□□ □□
60          goals.setESDate(sdayStart); // □□ □□□ □□
61
62          // □□ □□ □□ □□
63          if (sql.isGoalExists(EId, startDate, endDate
   )) {
64              // □□ □□□□ □□□ □□□ □□ □□□ □□□ □□□ □□
65              System.out.println("□□ □□□ □□ □□□ □□□□
   □□□□.");
66              return; // □□ □□□ □□□ □□ □□□ □□
67          }
68
69          // □□ □□
```

```
70          sql.saveGoal(goals); // □□□ □□□ □□□□□□□ □□
71      }
72
73      // □□ □□ □□ □□□
74      static ExTypes selectExerciseName(List<ExTypes>
    exList) {
75          Scanner sc = new Scanner(System.in); // □□□
    □□□ □□ Scanner □□ □□
76          System.out.println("□□ □□ □□");
77
78          // □□ □□ □□□ □□
79          for (int i = 0; i < exList.size(); i++) {
80              System.out.println("[" + (i + 1) + "]
    □□□□: " + exList.get(i).getExName());
81          }
82          System.out.println("□□□□□ □□□□□ >>");
83
84          // □□□□□□ □□□ □□ □□□ □□□□
85          int selectedNum = sc.nextInt() - 1; // □□□□
    □□□ □□□□ 1□ □□ □□□□ □□
86
87          // □□□ □□□ □□□□ □□
88          if (selectedNum >= 0 && selectedNum < exList
    .size()) {
89              return exList.get(selectedNum); // □□□□
    □□□ □□ □□
90          } else {
91              // □□□ □□□ □□□□ □□ □□
92              System.out.println("□□□ □□□□□.");
93              return null; // □□□ □□□ □□ null □□
94          }
95      }
96  }
```

```java
package Util;

import ExDAO.ExSQL;
import ExDTO.EMember;

import java.util.Scanner;

public class MemberUtil {

    public void Info(String EId) {
        Scanner sc = new Scanner(System.in);
        ExSQL sql = new ExSQL();

        EMember member;
        boolean run = true;

        while (run) {
            System.out.println("==========================================");
            System.out.println("[1]□□□□□\t\t[2]□□□□□\t\t[3]□□□□□□\t\t");
            System.out.println("[4]□□□□□□    [5]□□□□\t\t[6]□□□□");
            System.out.println("==========================================");
            System.out.println("□□ >>");
            int menu = sc.nextInt();
            switch (menu) {
                case 1:
                    member = sql.memberDetail(EId);
                    System.out.println("==========================================");
                    System.out.println("[1]□□□□\t\t[2]□   □\t\t[3]□   □");
                    System.out.println("[4]□   □\t\t[5]□□□\t\t[6]□□□□");
                    System.out.println("[7]□   \t\t[8]□□□\t\t");
                    System.out.println("==========================================");
                    System.out.print("□□□ □□ □□ >> "
```

```
32 );
33                         int menu2 = sc.nextInt();
34                         switch (menu2) {
35                             case 1:
36                                 System.out.print("□□□□
    >> ");
37                                 member.setEPw(sc.next());
38                                 break;
39                             case 2:
40                                 System.out.print("□□ >> "
   );
41                                 member.setEName(sc.next
   ());
42                                 break;
43                             case 3:
44                                 System.out.print("□□ >> "
   );
45                                 member.setEAge(sc.nextInt
   ());
46                                 break;
47                             case 4:
48                                 System.out.print("□□ >> "
   );
49                                 member.setEGender(sc.next
   ());
50                                 break;
51                             case 5:
52                                 System.out.print("□□□
    >> ");
53                                 member.setEEmail(sc.next
   ());
54                                 break;
55                             case 6:
56                                 System.out.print("□□□□
    >> ");
57                                 member.setEPhone(sc.next
   ());
58                                 break;
59                             case 7:
60                                 System.out.print("□ >> "
   );
```

```
61                                  member.setEHeight(sc.
   nextInt());
62                                  break;
63                              case 8:
64                                  System.out.print("□□□
    >> ");
65                                  member.setEWeight(sc.
   nextInt());
66                                  break;
67                              default:
68                                  System.out.println("□□
   □□□□□□");
69                                  break;
70                          }
71                          sql.ememberUpdate(member);
72                          break;
73                      case 2:
74                          sql.memberList(EId);
75                          break;
76                      case 3:
77                          GraphUtility.
   displayCaloriesGraph(EId);
78                          break;
79                      case 4:
80                          sql.gList(EId);
81                          break;
82                      case 5:
83                          System.out.print("□□ □□□□□□□□? (
   Y/N)>> ");
84                          String checkDelete = sc.next();
85                          if (checkDelete.equals("y")) {
86                              sql.ememberDelete(EId);
87                              EId = "";
88                          } else {
89                              System.out.println("□□□
   □□□□□.");
90                          }
91                          break;
92                      case 6:
93                          run = false; // □□□□
94                          break;
```

```
 95
 96                 default:
 97                     System.out.println("□□ □□□□□□□."
    );
 98                     break;
 99             }
100         }
101     }
102     public void exjoin(EMember member) {
103         Scanner sc = new Scanner(System.in);
104         System.out.println("□□□ >>");
105         member.setEId(sc.next());
106         System.out.println("□□□□ >>");
107         member.setEPw(sc.next());
108         System.out.println("□□ >>");
109         member.setEName(sc.next());
110         System.out.println("□□ >>");
111         member.setEAge(sc.nextInt());
112         System.out.println("□□ >>");
113         member.setEGender(sc.next());
114         System.out.println("□□□ >>");
115         member.setEEmail(sc.next());
116         System.out.println("□□□□ >>");
117         member.setEPhone(sc.next());
118         System.out.println("□ >>");
119         member.setEHeight(sc.nextInt());
120         System.out.println("□□□ >>");
121         member.setEWeight(sc.nextInt());
122     }
123 }
124
```

```java
package Util;

import ExDAO.Calender;
import ExDAO.ExSQL;
import ExDTO.EMember;
import ExDTO.ERecords;
import ExDTO.ExTypes;

import java.util.List;
import java.util.Scanner;

public class RecordUtil {
    ExSQL sql = new ExSQL(); // □□□□□□ □□□ □□ ExSQL □□ □□
    Scanner sc = new Scanner(System.in); // □□□ □□□ □□ □□ Scanner □□ □□

    public void record(ERecords records, String EId) {
        // □□ □□□ □□ ID □□
        records.setERNum(sql.recordNum()); // □ □□ □□ □□
        records.setERId(EId); // □□ ID □□

        int ExNum = 1; // □□ □□ □□ □□

        // □□ □□ □□□□
        EMember member = sql.emList(EId); // □□ □□ □□□□

        // □□ □□ □□ □□□□
        ExTypes extype = sql.tList(ExNum); // □□ □□ □□ □□□□

        System.out.println("□□ □□□ □□□□□");
        List<ExTypes> exNameList = sql.getExName(); // □□ □□ □□ □□□□

        // □□ □□□ □□□□ □□ □□□ □□
        if (exNameList == null || exNameList.isEmpty()) {
```

```
34              System.out.println("□□ □□□ □□□ □ □□□□.
   □□□ □□ □□□□.");
35              return; // □□□ □□ □□□ □□
36          }
37
38          // □□ □□ □□ □□
39          System.out.println("□□ □□ □□");
40          for (int i = 0; i < exNameList.size(); i++) {
41              System.out.println("[" + (i + 1) + "]
   □□□□: " + exNameList.get(i).getExName()); // □□ □□ □□
42          }
43
44          System.out.print("□□□□□ □□□□□ >> ");
45          int selectedIndex = sc.nextInt() - 1; // □□□
   □□ □□
46
47          // □□□□ □□□ □□ □□□ □□ □□□ □□
48          if (selectedIndex >= 0 && selectedIndex <
   exNameList.size()) {
49              ExTypes selectedExercise = exNameList.get
   (selectedIndex); // □□□ □□ □□ □□□□
50              records.setERHNum(selectedExercise.
   getEXNUM()); // □□□ □□□ □□ □□
51              records.setEExType(selectedExercise.
   getExName()); // □□□ □□□ □□ □□
52          } else {
53              System.out.println("□□□ □□ □□□ □□□□ □□□□□
   ."); // □□□ □□ □ □□□ □□
54              return; // □□ □□ □ □□
55          }
56
57          // □□ □□ □□
58          Calender.calendar(0);  // □□ □□ □□
59          System.out.print("□□ □□ (□) >> ");
60          String EDate = sc.next();  // □□□□ □□□ □□
61          String day = "2024-09-";  // □□ □□ □□
62          String sday = day + String.format("%02d",
   Integer.parseInt(EDate));  // '2024-09-01' □□□□ □□
63          records.setEDate(sday); // □□□ □□ □□
64
65          // □□ □□ □□ □□
```

```
66          System.out.print("□□ (□) >> ");
67          int time = sc.nextInt(); // □□□□ □□□ □□ □□
68          records.setETime(time); // □□ □□ □□
69
70          // □□ □ □□□ □□
71          float weight = member.getEWeight(); // □□□
    □□ □□□□
72          float exCalories = extype.getExCalories();
    // □□ □□□ □□□ □□□ □□□□
73
74          System.out.println("□□: " + weight); // □□
    □□
75          System.out.println("□□ □□□ □□□ □□□: " +
    exCalories); // □□□ □□□ □□
76
77          // □□ □□□ □□ □ □□
78          double caloriesBurned = exCalories * weight
     * (time / 60.0); // □ □□ □□□ □□
79          records.setECalolies((int) caloriesBurned);
    // □□ □□□ □□
80          System.out.printf("□ □□ □□□: %.2f kcal\n",
    caloriesBurned); // □□ □□□ □□
81
82          // □□ □□ □□ □ □□ □□
83          if (sql.checkExistingRecords(EId, EDate)) {
    // □□ □□□ □□□ □□
84              Calender.calendar(Integer.parseInt(EDate
    )); // □□□ □□ □□
85          }
86
87          // □□ □□□ □□□□□□□ □□
88          sql.ExRecord(records); // □□ □□ □□
89      }
90  }
```

File - D:\javaworkspace\miniProject\src\Util\GraphUtility.java

```java
package Util;

import ExDAO.ExSQL;
import ExDTO.ERecords;

import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.util.List;

public class GraphUtility {
    private static JFrame frame; // □□□□ □□□ JFrame □□

    // □□ □□ □□□ □□□□ □□□□ □□□□ □□□
    public static void displayCaloriesGraph(String EId) {
        ExSQL sql = new ExSQL(); // □□□□□□ □□□ □□ ExSQL □□ □□
        // SQL□□ EId□ □□□□ □□ □□□ □□□
        List<ERecords> recordsList = sql.recordList(EId); // EId□ □□ □□ □□ □□□□

        // □□ JFrame□ □□□□ □□ □□ □□□□ □□□□
        if (frame == null) {
            frame = new JFrame("□□ □□ □□□"); // □ JFrame □□
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // □ □□ □□ □□
            frame.setSize(1000, 1000); // □□□ □□ □□
        } else {
            frame.getContentPane().removeAll(); // □□ □□ □□
        }

        // □□□ □□□ □□□ □□□ □ (TreeMap□ □□□□ □□□ □□)
        Map<String, Integer> caloriesByDate = new TreeMap<>();

        // □ □□ □□□ □□□□ □□□ □□□ □□
        for (ERecords record : recordsList) {
```

```
34              // □□□□ □□ □□□ □□ (□: "2024-09-26 10:00:
   00" -> "2024-09-26")
35              String exerciseDate = record.getEDate().
   split(" ")[0];
36              int calories = record.getECalolies();
37
38              // □□ □□□ □□□ □□ □□ □□□□ □□□□ □□
39              caloriesByDate.merge(exerciseDate,
   calories, Integer::sum);
40          }
41
42          // □□□□ □□□ □□□ JFrame□ □□□□ □□□□
43          frame.add(new GraphPanel(caloriesByDate));
   // Map □□
44          frame.revalidate(); // □□□ □□ □□□
45          frame.repaint(); // □□□ □□ □□□
46          frame.setVisible(true); // □□□□ □□□ □□
47      }
48  }
49
50  // □□□□ □□□ JPanel □□□
51  class GraphPanel extends JPanel {
52      private Map<String, Integer> caloriesByDate; //
   □□□ □□□ □□□ □□□□ □
53
54      // □□□□□ □□□ □□□ □□□ □□□□
55      public GraphPanel(Map<String, Integer>
   caloriesByDate) {
56          this.caloriesByDate = caloriesByDate;
57      }
58
59      // □□□□ □□□ □□□ (JPanel□ paintComponent □□□ □□□□□
   )
60      @Override
61      protected void paintComponent(Graphics g) {
62          super.paintComponent(g); // □□ □□□ □□ □□
63
64          // □□□ □□□ □□□ □□□
65          int width = getWidth();
66          int height = getHeight();
67          // □□ □□□ □□□ □□ □□□ □□□□ □□□□ □□
```

```
68          int barWidth = width / (caloriesByDate.size
    () + 2);
69          // □□ □□□ □□ □□□□ □□□□ □□ □□ □□
70          int maxCalories = caloriesByDate.values().
    stream().max(Integer::compareTo).orElse(1);
71
72          int index = 0; // □□□□ □□ □□□
73          // □□□ □□□ □□□□ □□□□ □□ □□ □□
74          List<Color> colors = Arrays.asList(Color.
    BLUE, Color.GREEN, Color.RED, Color.YELLOW, Color.
    CYAN, Color.MAGENTA, Color.ORANGE, Color.PINK);
75
76          // □□□□ □□□□ □□ □□□
77          for (Map.Entry<String, Integer> entry :
    caloriesByDate.entrySet()) {
78              String date = entry.getKey(); // □□
79              int calories = entry.getValue(); // □□
    □□□ □□□
80
81              // □□□ □□□□ □□□□ □□ □□
82              Color color = colors.get(Math.abs(date.
    hashCode()) % colors.size());
83              g.setColor(color); // □□□ □□ □□
84
85              // □ □□□ □□ □□□□ □□□□ □□
86              int barHeight = (int) ((calories / (
    double) maxCalories) * (height - 100)); // 100□ □□
87
88              // □□ □□□
89              g.fillRect(50 + (index * barWidth),
    height - barHeight - 50, barWidth - 30, barHeight);
    // □□ □□ □ □□ □□
90
91              // □ □□ □□ □□□ □ □□
92              g.setColor(Color.BLACK);
93              g.drawString(calories + " kcal", 50 + (
    index * barWidth) + (barWidth / 4), height -
    barHeight - 60);
94
95              // X□□ □□ □□ (□□ □□)
96              g.setColor(Color.BLACK);
```

```
 97              int textWidth = g.getFontMetrics().
    stringWidth(date);
 98              g.drawString(date, 50 + (index *
    barWidth) + (barWidth / 2) - (textWidth / 2) - 10,
    height - 5); // □□ □□□□ □□ □□
 99
100              index++; // □□ □□□ □□ □□□ □□
101          }
102
103          // Y□□ □□□ □□ □□
104          g.setColor(Color.BLACK);
105          // □□ □□□ □□ □□□□ 10 □□□ □□□ □□
106          for (int i = 0; i <= maxCalories; i += (
    maxCalories / 10)) {
107              int y = height - 50 - (int) ((i / (
    double) maxCalories) * (height - 100));
108              g.drawString(String.valueOf(i), 10, y);
    // Y□□ □□□ □ □□
109              g.drawLine(45, y, 50, y); // Y□□ □□□ □□
110          }
111
112          // X□□ Y□ □ □□□ □□ □□
113          g.drawString("□□ □□ □□□", width / 2 - 50, 30
    ); // □□□ □□ □□
114          g.drawLine(50, height - 50, width, height -
    50); // X□ □□□
115          g.drawLine(50, 50, 50, height - 50); // Y□
    □□□
116      }
117 }
```

File - D:\javaworkspace\miniProject\src\ExDAO\Color.java

```java
package ExDAO;

public class Color {
    // Text colors
    public static final String RESET = "\u001B[0m"; // Reset color
    public static final String BLACK = "\u001B[30m"; // Black
    public static final String RED = "\u001B[31m"; // Red
    public static final String GREEN = "\u001B[32m"; // Green
    public static final String YELLOW = "\u001B[33m"; // Yellow
    public static final String BLUE = "\u001B[34m"; // Blue
    public static final String MAGENTA = "\u001B[35m"; // Magenta
    public static final String CYAN = "\u001B[36m"; // Cyan
    public static final String WHITE = "\u001B[37m"; // White
    // Bright text colors
    public static final String BRIGHT_BLACK = "\u001B[90m"; // Bright Black
    public static final String BRIGHT_RED = "\u001B[91m"; // Bright Red
    public static final String BRIGHT_GREEN = "\u001B[92m"; // Bright Green
    public static final String BRIGHT_YELLOW = "\u001B[93m"; // Bright Yellow
    public static final String BRIGHT_BLUE = "\u001B[94m"; // Bright Blue
    public static final String BRIGHT_MAGENTA = "\u001B[95m"; // Bright Magenta
    public static final String BRIGHT_CYAN = "\u001B[96m"; // Bright Cyan
    public static final String BRIGHT_WHITE = "\u001B[97m"; // Bright White
    public static final String BRIGHT_GRAY = "\u001B[37m";
```

```
24
25 }
```

```java
package ExDAO;

import ExDTO.*;
import Util.ExUtil;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ExSQL {

    Connection con; // □□□□□□ □□□ □□ Connection □□
    PreparedStatement pstmt; // SQL □□□ □□□□ □□ PreparedStatement □□
    ResultSet rs; // SQL □□ □□ □□□ □□ □□ ResultSet □□

    // □□□□□□ □□ □□ □□□
    public void conClose() {
        try {
            con.close(); // □□ □□
        } catch (SQLException e) {
            throw new RuntimeException(e); // □□ □□ □□□ □□□ □□
        }
    }

    // □□ □□ □□□
    public void join(EMember member) {
        con = ExUtil.getConnection(); // □□□□□□ □□
        try {
            String sql = "INSERT INTO EMEMBER VALUES (?,?,?,?,?,?,?,?,?)"; // SQL □□
            pstmt = con.prepareStatement(sql); // □□ □□
            // PreparedStatement□ □□□ □□
            pstmt.setString(1, member.getEId());
            pstmt.setString(2, member.getEPw());
            pstmt.setString(3, member.getEName());
            pstmt.setInt(4, member.getEAge());
            pstmt.setString(5, member.getEGender());
            pstmt.setString(6, member.getEEmail());
```

```
38              pstmt.setString(7, member.getEPhone());
39              pstmt.setInt(8, member.getEHeight());
40              pstmt.setInt(9, member.getEWeight());
41
42              int result = pstmt.executeUpdate(); // ▯▯
    ▯▯
43
44              // ▯▯▯ ▯▯ ▯▯
45              if (result > 0) {
46                  System.out.println(member.getEName
   () + "▯▯▯!! ▯▯▯▯ ▯▯ ▯▯▯▯▯▯.");
47              } else {
48                  System.out.println("▯▯▯▯ ▯▯");
49              }
50          } catch (Exception e) {
51              throw new RuntimeException(e); // ▯▯ ▯▯ ▯
    ▯▯▯ ▯▯▯ ▯▯
52          }
53      }
54
55      // ▯▯▯ ▯▯▯
56      public boolean login(String EId, String EPw) {
57          con = ExUtil.getConnection(); // ▯▯▯▯▯▯ ▯▯
58          try {
59              String sql = "SELECT * FROM EMEMBER WHERE
    EID = ? AND EPW =? "; // SQL ▯▯
60              pstmt = con.prepareStatement(sql); // ▯▯
   ▯▯
61
62              // PreparedStatement▯ ▯▯▯ ▯▯
63              pstmt.setString(1, EId);
64              pstmt.setString(2, EPw);
65
66              rs = pstmt.executeQuery(); // ▯▯ ▯▯
67
68              EMember member; // ▯▯ ▯▯▯ ▯▯ ▯▯
69
70              if (rs.next()) { // ▯▯▯ ▯▯▯▯ ▯▯
71                  member = new EMember();
72                  member.setEId(rs.getString("EId"));
   // ▯▯▯ ▯▯
```

```
 73                 member.setEPw(rs.getString("EPw"));
    // □□□□ □□
 74                 return true; // □□□ □□
 75             } else {
 76                 System.out.println("□□□□ □□□□□ □□□□
    □□□□!"); // □□□ □□ □□□
 77             }
 78         } catch (SQLException e) {
 79             throw new RuntimeException(e); // □□ □□
    □ □□□ □□□ □□
 80         }
 81         return false; // □□□ □□
 82     }
 83
 84     // □□ □□ □□ □□□
 85     public void ExRecord(ERecords eRecords) {
 86         con = ExUtil.getConnection(); // □□□□□□ □□
 87         try {
 88             String sql = "INSERT INTO ERECORDS
    VALUES (?, ?, ?, ?, ?, ?, ?)"; // SQL □□
 89             pstmt = con.prepareStatement(sql); // □□
     □□
 90
 91             // PreparedStatement□ □□□ □□
 92             pstmt.setInt(1, eRecords.getERNum());
 93             pstmt.setString(2, eRecords.getERId());
 94             pstmt.setInt(3, eRecords.getERHNum());
 95             pstmt.setString(4, eRecords.getEExType
    ());
 96             pstmt.setString(5, eRecords.getEDate());
 97             pstmt.setInt(6, eRecords.getETime());
 98             pstmt.setInt(7, eRecords.getECalolies
    ());
 99
100             int result = pstmt.executeUpdate(); //
    □□ □□
101
102             // □□□ □□ □□
103             if (result > 0) {
104                 System.out.println("□□□□□□□");
105             } else {
```

```
106                 System.out.println("□□□□");
107             }
108         } catch (SQLException e) {
109             System.out.println("SQL □□ □□: " + e.
    getMessage());
110             e.printStackTrace(); // □□ □□
111         }
112     }
113 
114     // □□ □□ □□ □□ □□□
115     public int recordNum() {
116         int ErNum = 0; // □□ □□ □□□
117         con = ExUtil.getConnection(); // □□□□□□ □□
118         try {
119             String sql = "SELECT MAX(ERNUM) FROM
    ERECORDS"; // □□ □□ □□ □□ SQL
120             pstmt = con.prepareStatement(sql); // □□
     □□
121 
122             rs = pstmt.executeQuery(); // □□ □□
123 
124             if (rs.next()) { // □□□ □□□□ □□
125                 ErNum = rs.getInt(1); // □□ □□ □□ □□
126             }
127         } catch (SQLException e) {
128             throw new RuntimeException(e); // □□ □□
    □ □□□ □□□ □□
129         }
130         return ErNum + 1; // □□ □□ □□ □□
131     }
132 
133     // □□ □□ □□ □□□
134     public EMember emList(String EId) {
135         con = ExUtil.getConnection(); // □□□□□□ □□
136         EMember member = null; // □□ □□ □□□
137 
138         try {
139             String sql = "SELECT ENAME, EHEIGHT,
    EWEIGHT FROM EMEMBER WHERE EId = ?"; // SQL □□
140             pstmt = con.prepareStatement(sql); // □□
     □□
```

```
141             pstmt.setString(1, EId); // □□□ □□
142             rs = pstmt.executeQuery(); // □□ □□
143 
144             if (rs.next()) { // □□□ □□□□ □□
145                 member = new EMember();
146                 member.setEName(rs.getString("EName"
    )); // □□ □□
147                 member.setEHeight((int) rs.getFloat(
    "EHeight")); // □ □□
148                 member.setEWeight((int) rs.getFloat(
    "EWeight")); // □□□ □□
149             }
150 
151         } catch (SQLException e) {
152             throw new RuntimeException(e); // □□ □□
    □ □□□ □□□ □□
153         }
154         return member;  // EMember □□ □□
155     }
156 
157     // □□ □□□: □□□ □□□□ □□□□ □□□□ □□
158     private List<ExTypes> getExTypes(String column
    ) {
159         con = ExUtil.getConnection(); // □□□□□□ □□
160         List<ExTypes> exList = new ArrayList<>();
    // □□ □□ □□□ □□□
161         try {
162             String sql = "SELECT EXNUM, " + column
     + " FROM EXTYPES"; // SQL □□
163             pstmt = con.prepareStatement(sql); // □□
     □□
164 
165             rs = pstmt.executeQuery(); // □□ □□
166 
167             while (rs.next()) { // □□□ □□□□ □□ □□
168                 ExTypes extypes = new ExTypes(); //
    □□ □□ □□ □□
169                 extypes.setEXNUM(rs.getInt("EXNUM"
    )); // □□ □□ □□
170                 if ("EXNAME".equals(column)) { //
    □□□ □□
```

```
171                     extypes.setExName(rs.getString("
    EXNAME")); // □□ □□
172                 }
173                 exList.add(extypes); // □□□□ □□
174             }
175         } catch (Exception e) {
176             throw new RuntimeException(e); // □□ □□
    □ □□□ □□□ □□
177         }
178         return exList; // □□ □□ □□□ □□
179     }
180 
181 
182     // □□ □□□ □□□□ □□□
183     public List<ExTypes> getExName() {
184         return getExTypes("EXNAME"); // □□ □□□□
185     }
186 
187     // □□ □□ □□ □□ □□□
188     public EMember memberDetail(String eId) {
189         con = ExUtil.getConnection(); // □□□□□□ □□
190         EMember member = new EMember(); // □□ □□ □□□
191 
192         try {
193             // (1) sql□ □□
194             String sql = "SELECT * FROM EMEMBER
    WHERE EID = ?"; // SQL □□
195 
196             // (2) DB□□
197             pstmt = con.prepareStatement(sql); // □□
     □□
198 
199             // (3) '?' □□□ □□ ('?' □□□ □□)
200             pstmt.setString(1, eId); // □□□ □□
201 
202             // (4) □□ : select => rs
203             rs = pstmt.executeQuery(); // □□ □□
204 
205             // (5) □□
206             if (rs.next()) { // □□□ □□□□ □□
207                 member.setEId(rs.getString(1)); //
```

```
207 □□□ □□
208                 member.setEPw(rs.getString(2)); //
    □□□□ □□
209                 member.setEName(rs.getString(3));
    // □□ □□
210                 member.setEAge(rs.getInt(4)); // □□
    □□
211                 member.setEGender(rs.getString(5));
    // □□ □□
212                 member.setEEmail(rs.getString(6));
    // □□□ □□
213                 member.setEPhone(rs.getString(7));
    // □□□□ □□
214                 member.setEHeight(rs.getInt(8)); //
    □ □□
215                 member.setEWeight(rs.getInt(9)); //
    □□□ □□
216             }
217         } catch (SQLException e) {
218             e.printStackTrace(); // □□ □□
219         }
220         return member; // □□ □□ □□
221     }
222
223     public void ememberUpdate(EMember member) {
224         con = ExUtil.getConnection(); // □□□□□□ □□
    □□
225         try {
226             // (1) SQL□ □□
227             String sql = "UPDATE EMEMBER SET EPW
     = ?, ENAME = ?, EAGE = ?, EGENDER = ?, EEMAIL = ?,
    EPHONE = ?, EHEIGHT=?, EWEIGHT=? WHERE EID = ?";
228
229             // (2) PreparedStatement □□
230             pstmt = con.prepareStatement(sql);
231
232             // (3) '?'□ □□□□ □□
233             pstmt.setString(1, member.getEPw());
234             pstmt.setString(2, member.getEName());
235             pstmt.setInt(3, member.getEAge());
236             pstmt.setString(4, member.getEGender());
```

```
237             pstmt.setString(5, member.getEEmail());
238             pstmt.setString(6, member.getEPhone());
239             pstmt.setInt(7, member.getEHeight());
240             pstmt.setInt(8, member.getEWeight());
241             pstmt.setString(9, member.getEId());
242 
243             // (4) □□
244             int result = pstmt.executeUpdate();
245 
246             // (5) □□ □□
247             if (result > 0) {
248                 System.out.println("□□□□ □□□ □□□□□ 
    □□□□□□□□□□!");
249             } else {
250                 System.out.println("□□ □□");
251             }
252         } catch (SQLException e) {
253             throw new RuntimeException(e); // □□ □□ 
    □ □□□ □□□ □□□
254         }
255     }
256 
257     // □□ □□□
258     public void ememberDelete (String EId) {
259         Connection con = null;
260         PreparedStatement pstmt = null;
261 
262         try {
263             con = ExUtil.getConnection(); // □□□□□□ 
    □□ □□
264 
265             String deleteGoalsRecords = "DELETE FROM
     EGOALS WHERE EGID = ?"; // □□ □□□ □□ SQL
266             pstmt = con.prepareStatement(
    deleteGoalsRecords);
267             pstmt.setString(1, EId); // EId □□□
268             pstmt.executeUpdate(); // □□ □□□ □□ □□
269 
270             String deleteMENURecords = "DELETE FROM
    MENU WHERE EMID = ?"; // □□ □□□ □□ SQL
271             pstmt = con.prepareStatement(
```

```
271 deleteMENURecords);
272             pstmt.setString(1, EId); // EId □□□
273             pstmt.executeUpdate(); // □□ □□□ □□ □□
274 
275             // □□□ □□□ □□ □□ □□□ □□
276             String deleteRecords = "DELETE FROM
    ERECORDS WHERE ERID = ?"; // □□ □□□ □□ SQL
277             pstmt = con.prepareStatement(
    deleteRecords);
278             pstmt.setString(1, EId); // EId □□□
279             pstmt.executeUpdate(); // □□ □□□ □□ □□
280 
281             // □□ □□ □□
282             String sql = "DELETE FROM EMEMBER WHERE
    EID = ?"; // □□ □□ SQL
283             pstmt = con.prepareStatement(sql);
284             pstmt.setString(1, EId); // EId □□□
285 
286             int result = pstmt.executeUpdate(); //
    □□ □□ □□
287             if (result > 0) {
288                 System.out.println("□□ □□□ □□□□□□□!"
    );
289             } else {
290                 System.out.println("□□ □□");
291             }
292         } catch (SQLException e) {
293             // □□ □□ □ □□□ □□
294             System.out.println("□□ □ □□□ □□□□□□: "
     + e.getMessage());
295         } finally {
296             // □□ □□
297             try {
298                 if (pstmt != null) pstmt.close();
    // PreparedStatement □□
299                 if (con != null) con.close(); // □□
    □□
300             } catch (SQLException e) {
301                 // □□ □□ □ □□ □□ □ □□□ □□
302                 System.out.println("□□ □□ □ □□ □□: "
     + e.getMessage());
```

```
303             }
304         }
305     }
306     // □□ □□ □□ □□□
307     public void memberList(String EId) {
308         con = ExUtil.getConnection(); // □□□□□□ □□
    □□
309         ArrayList<EMember> memberList = new
    ArrayList<>(); // □□ □□□ □□□ □□□
310         try {
311             String sql = "SELECT * FROM EMEMBER
    WHERE EID = ?"; // □□ □□ □□ SQL □
312             pstmt = con.prepareStatement(sql);
313             pstmt.setString(1, EId); // EId □□□
314             rs = pstmt.executeQuery(); // □□ □□
315 
316             while (rs.next()) { // □□ □□ □□
317                 EMember member = new EMember();
318                 member.setEId(rs.getString(1));
319                 member.setEPw(rs.getString(2));
320                 member.setEName(rs.getString(3));
321                 member.setEAge(rs.getInt(4));
322                 member.setEGender(rs.getString(5));
323                 member.setEEmail(rs.getString(6));
324                 member.setEPhone(rs.getString(7));
325                 member.setEHeight(rs.getInt(8));
326                 member.setEWeight(rs.getInt(9));
327                 memberList.add(member); // □□□□ □□
328             }
329 
330             // □□□ □□ □□
331             if (memberList.isEmpty()) {
332                 System.out.println("□□□ □□□ □□□□.");
333             } else {
334                 for (EMember mlist : memberList) {
335                     System.out.println(mlist); // □
    □□ □□ □□
336                 }
337             }
338 
339         } catch (SQLException e) {
```

```
340             System.out.println("SQL ??: " + e.
    getMessage()); // ?? ?? ? ?? ??? ??
341         } finally {
342             conClose(); // ?? ?? ??? ??
343         }
344     }
345 
346     // ?? ?? ?? ???
347     public List<ERecords> recordList(String EId) {
348         con = ExUtil.getConnection(); // ?????? ??
    ??
349         List<ERecords> rList = new ArrayList<>();//
    ?? ?? ??? ???
350 
351         try {
352             String sql = "SELECT * FROM ERECORDS
    WHERE ERID = ? ORDER BY ERNUM"; // ?? ?? ?? SQL ?
353             pstmt = con.prepareStatement(sql);
354             pstmt.setString(1, EId); // EId ???
355             rs = pstmt.executeQuery(); // ?? ??
356 
357             while (rs.next()) { // ?? ?? ??
358                 ERecords records = new ERecords();
359                 records.setERNum(rs.getInt(1));
360                 records.setERId(rs.getString(2));
361                 records.setERHNum(rs.getInt(3));
362                 records.setEExType(rs.getString(4));
363                 records.setEDate(rs.getString(5));
364                 records.setETime(rs.getInt(6));
365                 records.setECalolies(rs.getInt(7));
366                 rList.add(records); // ???? ??
367             }
368 
369             // ?? ?? ??
370             for (ERecords records : rList) {
371                 System.out.println(records);
372             }
373         } catch (SQLException e) {
374             throw new RuntimeException(e); // ?? ??
    ? ??? ??? ???
375         }
```

```
376         return rList;
377     }
378 
379     // □□ □□□ □□ □□□
380     public List<EFood> getfoodlist() {
381         con = ExUtil.getConnection(); // □□□□□□ □□
    □□
382         List<EFood> foList = new ArrayList<>(); //
    □□ □□□ □□□ □□□
383         try {
384             String sql = "SELECT * FROM FOOD"; // □□
     □□ SQL □
385             pstmt = con.prepareStatement(sql);
386 
387             rs = pstmt.executeQuery(); // □□ □□
388 
389             while (rs.next()) { // □□ □□ □□
390                 EFood foods = new EFood();
391                 foods.setFoNum(rs.getInt(1));
392                 foods.setFoName(rs.getString(2));
393                 foods.setFoGrams(rs.getString(3));
394                 foods.setFoCals(rs.getInt(4));
395                 foList.add(foods); // □□□□ □□
396             }
397         } catch (SQLException e) {
398             throw new RuntimeException(e); // □□ □□
    □ □□□ □□□ □□□
399         }
400 
401         return foList; // □□ □□□ □□
402     }
403 
404     // □□ □□ □□ □□□
405     public void saveGoal(EGoals goals) {
406         con = ExUtil.getConnection(); // □□□□□□ □□
    □□
407         try {
408             String sql = "INSERT INTO EGOALS VALUES
     (?, ?, ?, ?, ?, ?, ?)"; // □□ □□ SQL □
409 
410             // PreparedStatement□ □□□□ SQL □□ □□
```

```
411             pstmt = con.prepareStatement(sql);
412 
413             // ????? ??
414             pstmt.setInt(1, goals.getEGNum
    ());        // ?? ??
415             pstmt.setString(2, goals.getEGId
    ());      // ?? ???
416             pstmt.setInt(3, goals.getECNUM
    ());        // ?? ??
417             pstmt.setString(4, goals.getETEx
    ());      // ?? ?? ??
418             pstmt.setInt(5, goals.getETExTime
    ());     // ?? ?? ??
419             pstmt.setString(6, goals.getESDate
    ());     // ?? ???
420             pstmt.setString(7, goals.getEEDate
    ());     // ?? ???
421 
422             // SQL ??
423             int result = pstmt.executeUpdate();  //
    ?? ??, ??? ?? ?? ??? ??
424             if (result > 0) {
425                 System.out.println("?? ??? ??????
    ????????!");
426             }
427         } catch (SQLException e) {
428             e.printStackTrace(); // ?? ?? ? ?? ????
    ??
429         }
430     }
431 
432     // ?? ?? ?? ???
433     public int goalsNum() {
434         con = ExUtil.getConnection(); // ??????? ??
    ??
435         int EgNum = 0; // ?? ??? ??? ??
436         try {
437             String sql = "SELECT MAX(EGNUM) FROM
    EGOALS"; // ?? ?? ?? ?? SQL ?
438             pstmt = con.prepareStatement(sql);
439             rs = pstmt.executeQuery(); // ?? ??
```

```
440
441             if (rs.next()) {
442                 EgNum = rs.getInt(1); // □□ □□ □□ □□□□
443             }
444         } catch (SQLException e) {
445             throw new RuntimeException(e); // □□ □□ □ □□□ □□□ □□□
446         }
447         return EgNum + 1; // □□ □□ □□ □□
448     }
449
450     // □□ □□□ □□ □□□
451     public void gList(String EId) {
452         con = ExUtil.getConnection(); // □□□□□□ □□ □□
453         ArrayList<EGoals> gList = new ArrayList<>(); // □□ □□□ □□□ □□□
454
455         try {
456             String sql = "SELECT * FROM EGOALS WHERE EGID = ?"; // □□ □□ SQL □
457             pstmt = con.prepareStatement(sql);
458             pstmt.setString(1, EId); // EId □□□
459
460             rs = pstmt.executeQuery(); // □□ □□
461
462             while (rs.next()) { // □□ □□ □□
463                 EGoals goal = new EGoals();
464                 goal.setEGNum(rs.getInt(1));
465                 goal.setEGId(rs.getString(2));
466                 goal.setECNUM(rs.getInt(3));
467                 goal.setETEx(rs.getString(4));
468                 goal.setETExTime(rs.getInt(5));
469                 goal.setESDate(rs.getString(6));
470                 goal.setEEDate(rs.getString(7));
471                 gList.add(goal); // □□□□ □□
472             }
473
474             // □□ □□□ □□
475             for (EGoals egoal : gList) {
```

```
476                 System.out.println(egoal); // □ □□ □□ □□
477             }
478 
479         } catch (SQLException e) {
480             e.printStackTrace(); // □□ □□ □ □□ □□□□ □□
481         } finally {
482             conClose(); // □□ □□ □□□ □□
483         }
484     }
485 
486     // □□ □□□ □□□□ □□□
487     public ExTypes tList(int Exnum) {
488         con = ExUtil.getConnection(); // □□□□□□ □□ □□
489         ExTypes types = null; // □□□ □□□ ExTypes □□
490 
491         try {
492             // (1) SQL□ □□
493             String sql = "SELECT EXNAME, EXCALORIES FROM EXTYPES WHERE EXNUM = ?";
494             pstmt = con.prepareStatement(sql); // PreparedStatement □□
495             pstmt.setInt(1, Exnum); // □□ □□ □□□
496             rs = pstmt.executeQuery(); // □□ □□
497 
498             // (2) □□ □□
499             if (rs.next()) {
500                 types = new ExTypes(); // □□□ ExTypes □□ □□
501                 types.setExName(rs.getString("EXNAME")); // □□ □□ □□
502                 types.setExCalories((int) rs.getFloat("EXCALORIES")); // □□□ □□
503             }
504 
505         } catch (SQLException e) {
506             throw new RuntimeException(e); // SQL □□ □□ □ □□□ □□□ □□□
507         }
```

```
508         return types; // ExTypes □□ □□
509     }
510 
511     // □□ □□□ □□ □□□ □□□□ □□□
512     public boolean checkExistingRecords(String eId,
    String edate) {
513         con = ExUtil.getConnection(); // □□□□□□ □□
    □□
514 
515         try {
516             // (1) SQL□ □□
517             String sql = "SELECT COUNT(*) FROM
    ERECORDS WHERE ERID = ? AND TO_CHAR(EDATE, 'DD
    ') = ?";
518             pstmt = con.prepareStatement(sql); //
    PreparedStatement □□
519             pstmt.setString(1, eId); // □□ □□□ □□□
520             pstmt.setString(2, edate); // □□ □□□
521 
522             ResultSet rs = pstmt.executeQuery(); //
    □□ □□
523             if (rs.next()) {
524                 // (2) □□ □□
525                 int count = rs.getInt(1); // □ □□
    □□□ □ □□□□
526                 if (count > 0) {
527                     System.out.println("□□ □□□ □□□
    □□□□□"); // □□□ □□□□ □□
528                     return true; // □□ □□
529                 }
530             } else {
531                 System.out.println("□□ □□□ □□□ □□□□
    □□□□."); // □□□ □□□□ □□ □□
532             }
533         } catch (SQLException e) {
534             e.printStackTrace(); // □□ □□ □ □□ □□□□
    □□
535         }
536         return false; // □□□ □□□□ □□ □□
537     }
538     // □□ □□□ □□ □□□ □□□□ □□□
```

```
539
540     public boolean isGoalExists(String EId, String
    EsDate, String EeDate) {
541         con = ExUtil.getConnection(); // □□□□□□ □□
    □□
542
543         try {
544             // (1) SQL□ □□
545             String sql = "SELECT COUNT(*) FROM
    EGOALS WHERE EGID = ? AND TO_CHAR(ESDATE,'DD') >= ?
    AND TO_CHAR(EEDATE,'DD')<= ?";
546             pstmt = con.prepareStatement(sql); //
    PreparedStatement □□
547             pstmt.setString(1, EId); // □□ □□□ □□□
548             pstmt.setString(2, EsDate); // □□ □□ □□□
549             pstmt.setString(3, EeDate); // □□ □□ □□□
550             rs = pstmt.executeQuery(); // □□ □□
551
552             // (2) □□ □□
553             if (rs.next()) {
554                 int count = rs.getInt(1); // □ □□
    □□□ □ □□□□
555                 if (count > 0) {
556                     System.out.println("□□ □□□ □□□
    □□□□□"); // □□□ □□□□ □□
557                     return true; // □□ □□
558                 }
559             } else {
560                 System.out.println("□□ □□□ □□□ □□□□
    □□□□."); // □□□ □□□□ □□ □□
561             }
562         } catch (SQLException e) {
563             e.printStackTrace(); // □□ □□ □ □□ □□□□
    □□
564         }
565         return false; // □□□ □□□□ □□ □□
566     }
567
568     public int getFoodCals(int FoNum) {
569         int foNum =1;
570         con = ExUtil.getConnection();
```

```
571             try {
572                 String sql = "SELECT * FROM FOOD WHERE
    FONUM = ? ";
573                 pstmt = con.prepareStatement(sql);
574                 pstmt.setInt(1,FoNum);
575                 rs = pstmt.executeQuery();
576                 if (rs.next()) {
577                     rs.getInt("FONUM");
578                     rs.getString("FONAME");
579                     rs.getString("FOGRAMS");
580                     rs.getInt("FOCALS");
581                     rs.getInt("FOPROTEIN");
582                     rs.getInt("FOCARBOHYDRATES");
583                     rs.getInt("FOFATS");
584 
585                 }
586 
587             } catch (SQLException e) {
588                 throw new RuntimeException(e);
589             }
590             return foNum ;
591         }
592 
593         public void insertMenu(E_Menu menu) {
594             String sql = "INSERT INTO MENU VALUES
    (?,?,?,?,?,?,?,?)"; // SQL □□
595             try {
596                 pstmt = con.prepareStatement(sql); // □□
    □□
597             // PreparedStatement□ □□□ □□
598             pstmt.setInt(1, menu.getEMNUM());
599             pstmt.setString(2, menu.getEMID());
600             pstmt.setString(3, menu.getEMDate());
601             pstmt.setInt(4, menu.getECalories());
602             pstmt.setInt(5, menu.getEProtein());
603             pstmt.setInt(6, menu.getECarbohydrates());
604             pstmt.setInt(7, menu.getEFats());
605             pstmt.setInt(8,menu.getEfNum());
606             int result = pstmt.executeUpdate(); // □□ □□
607 
608             // □□□ □□ □□
```

```
609             if (result > 0) {
610                 System.out.println("□□□ □□□□□□□");
611             }
612         } catch (Exception e) {
613             throw new RuntimeException(e); // □□ □□ □
    □□□ □□□ □□
614         }
615 }
616 
617     public int menuNum() {
618         con = ExUtil.getConnection();
619         int EmNum = 0;
620         try{
621             String sql = "SELECT MAX(EMNUM) FROM
    MENU";
622             pstmt = con.prepareStatement(sql);
623             rs = pstmt.executeQuery(); // □□ □□
624             if (rs.next()) {
625                 EmNum = rs.getInt(1); // □□ □□ □□
    □□□□
626             }
627         } catch (Exception e) {
628             throw new RuntimeException(e);
629         }
630 
631         return EmNum+1;
632     }
633 
634 }
635 
```

```java
package ExDAO;

public class Calender {
    // ANSI □□ □□ □□ □□
    public static final String red = "\u001B[31m";     // □□□
    public static final String blue = "\u001B[34m";    // □□□
    public static final String exit = "\u001B[0m";     // □□ □□□
   // □□□ □□

    public static void calendar(int highlightDay) {
        // □□□ □□ StringBuilder □□□
        StringBuilder calendar = new StringBuilder();

        // □□ □□ □□□□
        calendar.append("┌─────────────────────────────────────┐\n");
        calendar.append("│                2024 - 09            │\n");
        calendar.append("├─────┬─────┬─────┬─────┬─────┬─────┬─────┤\n");
        calendar.append("│ " + red + "SUN" + exit + " │ MON │ TUE │ WED │ THU │ FRI │ " + blue + "SAT" + exit + " │\n");
        calendar.append("├─────┼─────┼─────┼─────┼─────┼─────┼─────┤\n");

        int startDay = 1;          // □□ □□ (SUN□ 1□□□□ □□)
        int daysInMonth = 30;      // 9□□ □□ (□: 30□)

        // □ □□ □□
        for (int week = 0; week < 5; week++) {
            // □□ □□ □□
            for (int day = 1; day <= 7; day++) {
                int currentDay = week * 7 + day - startDay + 1; // □□ □□ □□

                // □□ □□□ □□□□ □□
```

```
31                  if (currentDay > 0 && currentDay <=
   daysInMonth) {
32                      if (highlightDay == currentDay) {
33                          // □□□ □□□ CYAN □□□□ □□
34                          calendar.append(" | " + Color
   .CYAN + String.format("%2d", currentDay) + exit + " "
   );
35                      } else {
36                          // □□□ □□ □□□ □□□□□.
37                          if (day == 1) { // □□□
38                              calendar.append("| " +
   red + String.format("%2d", currentDay) + exit + " ");
39                          } else if (day == 7) { // □□□
40                              calendar.append(" | " +
   blue + String.format("%2d", currentDay) + exit + " "
   );
41                          } else { // □□
42                              calendar.append(" | " +
   String.format("%2d", currentDay) + " ");
43                          }
44                      }
45                  } else {
46                      calendar.append(" |    "); // □
   □□ □□
47                  }
48              }
49              calendar.append("|\n"); // □ □
50              if (week < 4) {
51                  calendar.append(
   "├──────┼──────┼──────┼──────┼──────┼──────┼──────┤\n"); //
   □ □ □□
52              }
53          }
54
55          calendar.append(
   "└──────┴──────┴──────┴──────┴──────┴──────┴──────┘\n"); //
   □□ □□ □□□□
56
57          System.out.print(calendar); // □□□ □□ □□
58      }
59  }
```

```java
package ExDTO;

public class EBMI {
    int EBmiNum;
    int EBmiCode;
    String EBmiId;
    String EBmiName;
    String EBmiScale;


    public int getEBmiNum() {
        return EBmiNum;
    }

    public void setEBmiNum(int EBmiNum) {
        this.EBmiNum = EBmiNum;
    }

    public String getEBmiId() {
        return EBmiId;
    }

    public void setEBmiId(String EBmiId) {
        this.EBmiId = EBmiId;
    }

    public String getEBmiName() {
        return EBmiName;
    }

    public void setEBmiName(String EBmiName) {
        this.EBmiName = EBmiName;
    }

    public String getEBmiScale() {
        return EBmiScale;
    }

    public void setEBmiScale(String EBmiScale) {
        this.EBmiScale = EBmiScale;
    }
```

```
42
43      public int getEBmiCode() {
44          return EBmiCode;
45      }
46
47      public void setEBmiCode(int EBmiCode) {
48          this.EBmiCode = EBmiCode;
49      }
50
51
52
53      @Override
54      public String toString() {
55          return "BMI □" +
56                  "EBmiNum=" + EBmiNum +
57                  ", EBmiId='" + EBmiId + '\'' +
58                  ", EBmiName='" + EBmiName + '\'' +
59                  ", EBmiScale='" + EBmiScale + '\'' +
60                  ", EBmiCode=" + EBmiCode +
61                  "□";
62      }
63
64
65  }
66
```

```java
package ExDTO;

public class EFood {
    private int FoNum;              // □□ □□
    private String FoName;          // □□ □□
    private String FoGrams;         // □□
    private int FoCals;             // □□□
    private int FoProtein;          // □□□
    private int FoCarbohydrates;    // □□□□
    private int FoFats;             // □□

    public int getFoNum() {
        return FoNum;
    }

    public void setFoNum(int foNum) {
        FoNum = foNum;
    }

    public String getFoName() {
        return FoName;
    }

    public void setFoName(String foName) {
        FoName = foName;
    }

    public String getFoGrams() {
        return FoGrams;
    }

    public void setFoGrams(String foGrams) {
        FoGrams = foGrams;
    }

    public int getFoCals() {
        return FoCals;
    }

    public void setFoCals(int foCals) {
        FoCals = foCals;
```

```
42      }
43
44      public int getFoProtein() {
45          return FoProtein;
46      }
47
48      public void setFoProtein(int foProtein) {
49          FoProtein = foProtein;
50      }
51
52      public int getFoCarbohydrates() {
53          return FoCarbohydrates;
54      }
55
56      public void setFoCarbohydrates(int
    foCarbohydrates) {
57          FoCarbohydrates = foCarbohydrates;
58      }
59
60      public int getFoFats() {
61          return FoFats;
62      }
63
64      public void setFoFats(int foFats) {
65          FoFats = foFats;
66      }
67
68      @Override
69      public String toString() {
70          return "EFood{" +
71                  "FoNum=" + FoNum +
72                  ", FoName='" + FoName + '\'' +
73                  ", FoGrams='" + FoGrams + '\'' +
74                  ", FoCals=" + FoCals +
75                  ", FoProtein=" + FoProtein +
76                  ", FoCarbohydrates=" +
    FoCarbohydrates +
77                  ", FoFats=" + FoFats +
78                  '}';
79      }
80  }
```

```java
package ExDTO;

public class E_Menu {

    private int EMNUM;
    private String EMID;
    private String EMDate;           // □□
    private int ECalories;          // □□□
    private int EProtein;           // □□□
    private int ECarbohydrates;     // □□□□
    private int EFats;              // □□
    private int EfNum;

    public int getEMNUM() {
        return EMNUM;
    }

    public void setEMNUM(int EMNUM) {
        this.EMNUM = EMNUM;
    }

    public String getEMID() {
        return EMID;
    }

    public void setEMID(String EMID) {
        this.EMID = EMID;
    }

    public String getEMDate() {
        return EMDate;
    }

    public void setEMDate(String EDate) {
        this.EMDate = EDate;
    }

    public int getECalories() {
        return ECalories;
    }

```

```
42      public void setECalories(int ECalories) {
43          this.ECalories = ECalories;
44      }
45
46      public int getEProtein() {
47          return EProtein;
48      }
49
50      public void setEProtein(int EProtein) {
51          this.EProtein = EProtein;
52      }
53
54      public int getECarbohydrates() {
55          return ECarbohydrates;
56      }
57
58      public void setECarbohydrates(int ECarbohydrates
    ) {
59          this.ECarbohydrates = ECarbohydrates;
60      }
61
62      public int getEFats() {
63          return EFats;
64      }
65
66      public void setEFats(int EFats) {
67          this.EFats = EFats;
68      }
69
70      public int getEfNum() {
71          return EfNum;
72      }
73
74      public void setEfNum(int efNum) {
75          EfNum = efNum;
76      }
77
78
79      @Override
80      public String toString() {
81          return "E_Menu{" +
```

```
82                      "EMNUM=" + EMNUM +
83                      ", EMID='" + EMID + '\'' +
84                      ", EMDate='" + EMDate + '\'' +
85                      ", ECalories=" + ECalories +
86                      ", EProtein=" + EProtein +
87                      ", ECarbohydrates=" + ECarbohydrates
    +
88                      ", EFats=" + EFats +
89                      ", EfNum=" + EfNum +
90                      '}';
91      }
92  }
93
```

File - D:\javaworkspace\miniProject\src\ExDTO\EGoals.java

```java
package ExDTO;


public class EGoals {

    private int EGNum;       // □□ □□
    private String EGId;     // □□ □□□
    private int ECNUM;
    private String ETEx;     // Ex-Target-Exercise, □□ □□
    private int ETExTime;    // Ex-Target-Exercise, □□ □□ □□
    private String ESDate;   // Ex-Start-Date, □□ □□□
    private String EEDate;   // Ex-End-Date, □□ □□□

    public int getEGNum() {
        return EGNum;
    }

    public void setEGNum(int EGNum) {
        this.EGNum = EGNum;
    }

    public String getEGId() {
        return EGId;
    }

    public void setEGId(String EGId) {
        this.EGId = EGId;
    }

    public int getECNUM() {
        return ECNUM;
    }

    public void setECNUM(int ECNUM) {
        this.ECNUM = ECNUM;
    }

    public String getETEx() {
        return ETEx;
```

```
40      }
41
42      public void setETEx(String ETEx) {
43          this.ETEx = ETEx;
44      }
45
46      public int getETExTime() {
47          return ETExTime;
48      }
49
50      public void setETExTime(int ETExTime) {
51          this.ETExTime = ETExTime;
52      }
53
54      public String getESDate() {
55          return ESDate;
56      }
57
58      public void setESDate(String ESDate) {
59          this.ESDate = ESDate;
60      }
61
62      public String getEEDate() {
63          return EEDate;
64      }
65
66      public void setEEDate(String EEDate) {
67          this.EEDate = EEDate;
68      }
69
70      @Override
71      public String toString() {
72          return "┌──────────────□□□──────────────┐\n"
    +
73                  "□□ □□ : ["+EGNum+"]\n" +
74                  "□□□ : ["+EGId+"]\n" +
75                  "□□ □□  : ["+ECNUM+"]\n" +
76                  "□□ □□ : ["+ETEx+"]\n" +
77                  "□□ □□□□ : ["+ETExTime+"]\n" +
78                  "□□ □□□ : ["+ESDate+"]\n" +
79                  "□□ □□□ : ["+EEDate+"]\n" +
```

```
80
   "└──────────────────────────────────────┘\n";
81      }
82 }
83
```

File - D:\javaworkspace\miniProject\src\ExDTO\EMember.java

```java
package ExDTO;

public class EMember {
    private String EId;     //□□ □□□
    private String EPw;     //□□ □□
    private String EName;   //□□ □□
    private int EAge;       //□□ □□
    private String EGender; //□□ □□
    private String EEmail;  //□□ □□□
    private String EPhone;  //□□ □□□□
    private int EHeight;    //□□ □
    private int EWeight;    //□□ □□□

    public String getEId() {
        return EId;
    }

    public void setEId(String EId) {
        this.EId = EId;
    }

    public String getEPw() {
        return EPw;
    }

    public void setEPw(String EPw) {
        this.EPw = EPw;
    }

    public String getEName() {
        return EName;
    }

    public void setEName(String EName) {
        this.EName = EName;
    }

    public int getEAge() {
        return EAge;
    }

```

```
42      public void setEAge(int EAge) {
43          this.EAge = EAge;
44      }
45      public String getEGender() {
46          return EGender;
47      }
48
49      public void setEGender(String EGender) {
50          this.EGender = EGender;
51      }
52      public String getEEmail() {
53          return EEmail;
54      }
55
56      public void setEEmail(String EEmail) {
57          this.EEmail = EEmail;
58      }
59
60      public String getEPhone() {
61          return EPhone;
62      }
63
64      public void setEPhone(String EPhone) {
65          this.EPhone = EPhone;
66      }
67
68      public int getEHeight() {
69          return EHeight;
70      }
71
72      public void setEHeight(int EHeight) {
73          this.EHeight = EHeight;
74      }
75
76      public int getEWeight() {
77          return EWeight;
78      }
79
80      public void setEWeight(int EWeight) {
81          this.EWeight = EWeight;
82      }
```

```
83
84      @Override
85      public String toString() {
86          return "┌──────────────────□□□──────────────────┐\n
    " +
87                  "              □□□ : ["+EId+"]\n" +
88                  "              □□□□ : ["+EPw+"]\n" +
89                  "              □□ : ["+EName+"]\n" +
90                  "              □□ : ["+EAge+"]\n" +
91                  "              □□ : ["+EGender+"]\n" +
92                  "              □□□ : ["+EEmail+"]\n" +
93                  "              □□□ : ["+EPhone+"]\n" +
94                  "              □ : ["+EHeight+"]\n" +
95                  "              □□□ : ["+EWeight+"]\n" +
96
    "└──────────────────────────────────────┘\n";
97      }
98  }
99
```

```java
package ExDTO;



public class ExTypes {
    private int EXNUM;
    private String ExName;        // □□□□
    private String ExCategory;    // □□□□□□
    private int ExCalories;    // □□□□□□

    public int getEXNUM() {
        return EXNUM;
    }

    public void setEXNUM(int EXNUM) {
        this.EXNUM = EXNUM;
    }

    public String getExName() {
        return ExName;
    }

    public void setExName(String exName) {
        ExName = exName;
    }

    public String getExCategory() {
        return ExCategory;
    }

    public void setExCategory(String exCategory) {
        ExCategory = exCategory;
    }

    public int getExCalories() {
        return ExCalories;
    }

    public void setExCalories(int exCalories) {
        ExCalories = exCalories;
    }
```

```
42
43      @Override
44      public String toString() {
45          return
46                  "□□□□ : " + ExCategory +
47                  "□□□□ : " + ExName;
48      }
49  }
50
51
```

```java
package ExDTO;


public class ERecords {
    private int ERNum;           // □□ □□
    private String ERId;// □□ □□□
    private int ERHNum;
    private String EExType;      // □□ □□
    private String EDate;        // □□ □□□
    private int ETime;           // □□ □□ □□
    private int ECalolies;       // □□ □□ □□□

    public int getERNum() {
        return ERNum;
    }

    public void setERNum(int ERNum) {
        this.ERNum = ERNum;
    }

    public String getERId() {
        return ERId;
    }

    public void setERId(String ERId) {
        this.ERId = ERId;
    }

    public int getERHNum() {
        return ERHNum;
    }

    public void setERHNum(int ERHNum) {
        this.ERHNum = ERHNum;
    }

    public String getEExType() {
        return EExType;
    }

    public void setEExType(String EExType) {
```

```
42          this.EExType = EExType;
43      }
44
45      public String getEDate() {
46          return EDate;
47      }
48
49      public void setEDate(String EDate) {
50          this.EDate = EDate;
51      }
52
53      public int getETime() {
54          return ETime;
55      }
56
57      public void setETime(int ETime) {
58          this.ETime = ETime;
59      }
60
61      public int getECalolies() {
62          return ECalolies;
63      }
64
65      public void setECalolies(int ECalolies) {
66          this.ECalolies = ECalolies;
67      }
68
69      @Override
70      public String toString() {
71          return "                    \uD83C\uDFC3\u200D
    \uFE0F"+ "\n"+
72                  "┌──────── · · ▯▯▯▯▯▯ · · ────────┐"
     +"\n"+
73                  "      ▯▯▯▯ : " + ERNum +"\n" +
74                  "      ▯▯▯ : " + ERId + "\n" +
75                  "      ▯▯▯▯ : " + ERHNum + "\n"+
76                  "      ▯▯▯▯ : " + EExType + "\n" +
77                  "      ▯▯▯▯▯ : " + EDate + "\n" +
78                  "      ▯▯▯▯▯▯ : " + ETime +"\n" +
79                  "      ▯▯▯▯▯▯▯ : " + ECalolies +"\n"+
80
```

```
80 "└──────────────────────────────────────┘";
81     }
82 }
83
```

```java
package ExMain;

import ExDAO.Color;
import ExDAO.ExSQL;
import ExDTO.EFood;
import ExDTO.EGoals;
import ExDTO.EMember;
import ExDTO.ERecords;

import Util.*;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int menu = 0;
        int menu1 = 0;
        boolean run = true;
        boolean run2;
        ExSQL sql = new ExSQL();
        EMember member = new EMember();
        ERecords records = new ERecords();
        EGoals goals = new EGoals();
        BmiUtil bUtil = new BmiUtil();
        GoalsUtil gUtil = new GoalsUtil();
        RecordUtil rUtil = new RecordUtil();
        MenuUtil mUtil = new MenuUtil();
        MemberUtil memUtil = new MemberUtil();
        String EId;
        String EPw;

        System.out.println(Color.BRIGHT_RED +
"                    @                       \n" +

"                  @@@@@@                        \n
" +

"                @@@@   @@@                     \n
" +

```

```
36 "              @@@@@        @@@                          \n
   " +
37                    Color.BRIGHT_YELLOW +
38
   "       @@@@@@             @@        @@@@                \n
   " +
39
   "     @@@    @@@             @@@    @@@ @@@              \n
   " +
40
   "     @@@      @@@@@           @@@@@    @@@              \n
   " +
41
   "     @@@         @@@                  @@@@              \n
   " +
42                    Color.BRIGHT_GREEN +
43
   "       @@@         @@                   @@@              \n
   " +
44
   "         @@@@@     @@@                @@@@@@           \n
   " +
45
   "              @@@@@  @@@                 @@@@           \n
   " +
46
   "                 @@@@  @@@            @@ @@           \n
   " +
47                    Color.BRIGHT_CYAN +
48
   "                    @@@ @@@@           @@@@@@          \n
   " +
49
   "                      @@    @@@@@@          @@@@        \n
   " +
50
   "                       @@@       @@@@      @@@@@       \n
   " +
51
   "                         @@@@      @@@          @@@     \n
   " +
```

```
52                  Color.BRIGHT_MAGENTA +
53
   "                          @@@@    @@@@          @@@    \n
   " +
54
   "                             @@@@    @@@@@       @@    \n
   " +
55
   "                                @@@@@@   @@@@@@@@     \n
   " +
56
   "                                         @@@@@@@         \n
   " + Color.RESET);
57
58          while(run){
59              System.out.println("==================
   ▯▯▯▯====================");
60              System.out.println("[1]▯▯▯▯\t\t\t[2]▯▯▯\t
   \t\t\t[3]▯▯");
61              System.out.println(
   "=============================================");
62              System.out.println("▯▯ >>");
63              menu = sc.nextInt();
64              switch (menu) {
65                  case 1:
66                      memUtil.exjoin(member);
67                      sql.join(member);
68                      break;
69                  case 2:
70                      System.out.println("▯▯▯ ▯▯ : ");
71                      EId = sc.next();
72                      System.out.println("▯▯▯▯ ▯▯ : ");
73                      EPw = sc.next();
74                      run = true;
75                      run2 = true;
76                      boolean login = sql.login(EId,
   EPw);
77                      if(login) {
78                          member = sql.emList(EId);
79                          System.out.println(member.
   getEName() + "▯▯▯ ▯▯▯ ▯▯▯▯▯!!");
```

```
 80                          while (run2) {
 81                              System.out.println(
    "==========================================");
 82                              System.out.println("[1]
    BMI□□\t\t[2]□□□□\t\t[3]□□□□\t\t");
 83                              System.out.println("[4]
    □□□□\t\t[5]□□□□□\t\t[6]□□□□\t\t");
 84                              System.out.println(
    "==========================================");
 85                              System.out.println("□□
     >>");
 86
 87                              menu1 = sc.nextInt();
 88                              switch (menu1) {
 89                                  case 1:
 90                                      bUtil.bmicul(EId
    );
 91                                      break;
 92                                  case 2:
 93                                      gUtil.goal(goals
    , EId);
 94                                      break;
 95                                  case 3:
 96                                      rUtil.record(
    records, EId);
 97                                      break;
 98                                  case 4:
 99                                      mUtil.menu(EId);
100                                      break;
101                                  case 5:
102                                      memUtil.Info(EId
    );
103                                      break;
104                                  case 6:
105                                      run2 = false;
106                                      System.out.
    println("□□□□□ □□□□□□ □□□□□ ");
107                                  default:
108                                      System.out.
    println("□□ □□□□□□□ □□ □□□□□");
109                              }
```

```
110                             }
111                         }
112                         break;
113                     case 3:
114                         run = false;
115                         sql.conClose();
116                         break;
117                     default:
118                         System.out.println("□□ □□□□□");
119                         break;
120                 }
121             }
122         }
123 }
124
```