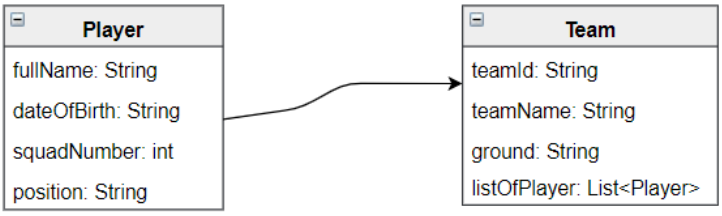
JAVA BASIC

Lab Guides

Assignment Specifications:

For the class hierarchy is as follows, the trainee let's create the java classes install this class diagram to be able to relationship between it.



» The Player class contains the information about players. Each player has its **fullName**, **dateOfBirth**, **squadNumber**, **position**.

» The Team class has contains about teams. Each team has an **teamId**, **teamName**, **ground**, **listOfPlayer**.

Functional Requirements:

a. The program has a functions to validate data, that call in setter method.

b. Create custom exception named "**IncorrectFormatException**" that will be thrown when input data incorrect format.

c. The program has a function to create a team(s) and user can add player(s) for each team.

d. The program has a function to display list of team information.

Business Rules:

» Date of birth: must be date format.

» Squad number: must be a number.

» Position: only one of ("GK", "CB", "WB", "CM","CF").

Guidelines:

» Step1. Create a project named **TMS** in Eclipse as below:

» Step2. Create package **r2s.training.entities** that contains **Player**, **Team** classes:

Player class

```
1.  package r2s.training.entities;
2.
3.  import java.text.ParseException;
4.  import java.text.SimpleDateFormat;
5.
6.  import r2s.training.exception.IncorrectFormatException;
7.  import r2s.training.util.Constant;
8.  import r2s.training.util.Validator;
9.
10. public class Player {
11.     private String fullName;
12.     private String dateOfBirth;
13.     private int squadNumber;
14.     private String position;
15.
16.     /**
17.      * Constructor for Player class without Parameters.
18.      */
19.     public Player() {
20.         super();
21.     }
22.
23.     /**
24.      * Constructor for Player class with Parameters.
25.      */
26.     public Player(String fullName, String dateOfBirth, int squadNumber, String position){
27.         super();
28.         this.fullName = fullName;
29.         this.dateOfBirth = dateOfBirth;
30.         this.squadNumber = squadNumber;
31.         this.position = position;
32.     }
33.
34.     public String getFullName() {
35.         return fullName;
36.     }
37.
38.     public void setFullName(String fullName) {
39.         this.fullName = fullName;
40.     }
41.
42.     public String getDateOfBirth() {
43.         return dateOfBirth;
44.     }
45.
46.     /**
47.      * @param dateOfBirth the dateOfBirth to set
48.      * @throws ParseException,IncorrectFormatException
49.      */
50.     public void setDateOfBirth(String dateOfBirth)
51.                                     throws ParseException, IncorrectFormatException {
52.         SimpleDateFormat dateFormat = new SimpleDateFormat("dd/mm/yyy");
53.         try {
54.             dateOfBirth = dateFormat.format(dateFormat.parse(dateOfBirth));
55.             if (Validator.isDate(dateOfBirth)) {
56.                 this.dateOfBirth = dateOfBirth;
57.             } else {
58.                 throw new IncorrectFormatException(Constant.INCORRECT_DATE_MASSAGE);
59.             }
60.         } catch (ParseException e) {
```

```
61.             throw e;
62.         }
63.     }
64.
65.     public int getSquadNumber() {
66.         return squadNumber;
67.     }
68.
69.     public void setSquadNumber(String squadNumber) throws NumberFormatException {
70.         this.squadNumber = Integer.parseInt(squadNumber);
71.
72.     }
73.
74.     public String getPosition() {
75.         return position;
76.     }
77.
78.     /**
79.      * @param position the position to set
80.      * @throws IncorrectFormatException
81.      */
82.     public void setPosition(String position) throws IncorrectFormatException {
83.         if (Validator.isPosition(position)) {
84.             this.position = position;
85.         } else {
86.             throw new IncorrectFormatException(Constant.INCORRECT_POSITION_MASSAGE);
87.         }
88.     }
89.
90.     /*
91.      * Method to display Player information.
92.      */
93.     @Override
94.     public String toString() {
95.         return "\n fullName:" + fullName +  ", dateOfBirth:"      + dateOfBirth +
96.                ", squadNumber:" + squadNumber  +
97.                ", position:"  + position;
98.     }
99.   }
```

Team class

```
1.  package r2s.training.entities;
2.
3.  import java.util.List;
4.
5.  public class Team {
6.
7.      private String teamId;
8.      private String teamName;
9.      private String ground;
10.     private List<Player> listOfPlayer;
11.     /**
12.      *  Constructor for Team class without Parameters.
13.      */
14.     public Team() {
15.         super();
16.     }
17.
18.     /**
19.      * Constructor for Team class without Parameters.
20.      * @param teamId
21.      * @param teamName
22.      * @param ground
23.      * @param listOfPlayer
24.      */
25.     public Team(String teamId,String teamName, String ground, List<Player> listOfPlayer){
```

```
26.         super();
27.         this.teamId = teamId;
28.         this.teamName = teamName;
29.         this.ground = ground;
30.         this.listOfPlayer = listOfPlayer;
31.     }
32.
33.     /**
34.      * @return the teamId
35.      */
36.     public String getTeamId() {
37.         return teamId;
38.     }
39.
40.     /**
41.      * @param teamId the teamId to set
42.      */
43.     public void setTeamId(String teamId) {
44.         this.teamId = teamId;
45.     }
46.
47.     /**
48.      * @return the teamName
49.      */
50.     public String getTeamName() {
51.         return teamName;
52.     }
53.
54.     /**
55.      * @param teamName the teamName to set
56.      */
57.     public void setTeamName(String teamName) {
58.         this.teamName = teamName;
59.     }
60.
61.     /**
62.      * @return the ground
63.      */
64.     public String getGround() {
65.         return ground;
66.     }
67.      /**
68.      * @param ground the ground to set
69.      */
70.     public void setGround(String ground) {
71.         this.ground = ground;
72.     }
73.
74.     /**
75.      * @return the listOfPlayer
76.      */
77.     public List<Player> getListOfPlayer() {
78.         return listOfPlayer;
79.     }
80.
81.     /**
82.      * @param listOfPlayer the listOfPlayer to set
83.      */
84.     public void setListOfPlayer(List<Player> listOfPlayer) {
85.         this.listOfPlayer = listOfPlayer;
86.     }
87.
88.     /*
89.      * Method to display Team information.
90.      */
91.     @Override
92.     public String toString() {
93.         return "teamId:" + teamId + ", teamName:" + teamName +
```

```
94.                 ", ground:" + ground +"\n"+ ", listOfPlayer:"  + listOfPlayer;
95.     }
96. }
```

» Step3. Create package **r2s.training.util** that contains **Constant, Validator** classes:

Constants class:

```
1.  package r2s.training.util;
2.
3.  public class Constant {
4.      // message
5.      public static final String INCORRECT_DATE_MASSAGE = "Date value incorrect format!";
6.      public static final String INCORRECT_POSITION_MASSAGE =
7.                                                      "Position value incorrect format!";
8.      // Regex pattern
9.      public static final String POSITION_PATTERN = "^GK|CB|WB|CM|CF$";
10.     public static final String DATE_PATTERN = "^([0-2][0-9]|(3)[0-1])(\\/)(((0)[0-
    9])|((1)[0-2]))(\\/)\\d{4}$";
11. }
```

Validator class:

```
1.  package r2s.training.util;
2.
3.  import java.util.regex.Matcher;
4.  import java.util.regex.Pattern;
5.
6.  public class Validator {
7.
8.      private static Matcher matcher = null;
9.      private static Pattern pattern = null;
10.
11.     /**
12.      * This method check format of date value.
13.      *
14.      * @param String date.
15.      * @return boolean
16.      */
17.     public static boolean isDate(String date) {
18.         pattern = Pattern.compile(Constant.DATE_PATTERN);
19.         matcher = pattern.matcher(date);
20.         return matcher.matches();
21.     }
22.
23.     /**
24.      * This method check format of "position" attribute.
25.      *
26.      * @param String position.
27.      * @return boolean
28.      */
29.     public static boolean isPosition(String position) {
30.         pattern = Pattern.compile(Constant.POSITION_PATTERN);
31.         matcher = pattern.matcher(position);
32.         return matcher.matches();
33.     }
34. }
```

» Step4. Create package **r2s.tranning.exception** that contains **IncorrectFormatException** classes:

IncorrectFormatException class:

```
1.  package r2s.training.exception;
2.
3.  public class IncorrectFormatException extends Exception {
```

```
    private static final long serialVersionUID = 1L;

    /**
     * Constructor for IncorrectFomartException without Parameters.
     */
    public IncorrectFormatException() {
        super();
    }

    /**
     * Constructor for IncorrectFomartException with message.
     */
    public IncorrectFormatException(String message) {
        super(message);
    }
}
```

» Step5. Create package **r2s.training.service** that contains **PlayerService, TeamService** classes:

PlayerService class:

```
package r2s.training.service;

import java.text.ParseException;
import java.util.Scanner;

import r2s.training.entities.Player;
import r2s.training.exception.IncorrectFormatException;

public class PlayerService {

    Player player = null;

    /**
     * This method take input player's information from keyboard.
     *
     * @param scanner
     * @return Player
     * @throws ParseException,IncorrectFormatException
     */
    public Player inputPlayer(Scanner scanner)
                                    throws ParseException, IncorrectFormatException {

        // Create new player and set attributes.
        player = new Player();
        String fullName, dateOfBirth, position, squadNumber;
        System.out.print("\nEnter full name: ");
        fullName = scanner.nextLine();
        player.setFullName(fullName);
        System.out.print("\nEnter date of birth(dd/mm/yyyy): ");
        dateOfBirth = scanner.nextLine();
        player.setDateOfBirth(dateOfBirth);
        System.out.print("\nEnter squad number: ");
        squadNumber = scanner.nextLine();
        player.setSquadNumber(squadNumber);
        System.out.print("\nEnter position( GK|CB|WB|CM|CF):");
        position = scanner.nextLine();
        player.setPosition(position);
        return player;
    }
}
```

TeamService class:

```
package r2s.training.service;

```

```
3.  import java.text.ParseException;
4.  import java.util.ArrayList;
5.  import java.util.List;
6.  import java.util.Scanner;
7.
8.  import r2s.training.entities.Player;
9.  import r2s.training.entities.Team;
10. import r2s.training.exception.IncorrectFormatException;
11.
12. public class TeamService {
13.     private String teamId;
14.     private String teamName;
15.     private String ground;
16.     private List<Player> listOfPlayer;
17.     private PlayerService playerService = new PlayerService();
18.
19.     /**
20.      * This method create take input team's information from keyboard.
21.      *
22.      * @param scanner: Scanner
23.      * @return Team
24.      * @throws ParseException, IncorrectFormatException
25.      */
26.     public Team createNewTeam(Scanner scanner)
27.                                         throws ParseException, IncorrectFormatException {
28.
29.         System.out.println("Input team information!");
30.         String choice = "y";
31.         listOfPlayer = new ArrayList<>();
32.         System.out.print("Enter team ID:");
33.         teamId = scanner.nextLine();
34.         System.out.print("Enter full name of team:");
35.         teamName = scanner.nextLine();
36.         System.out.print("Enter ground:");
37.         ground = scanner.nextLine();
38.         System.out.print("Do you want to add new player for this team? (y/n)");
39.         choice = scanner.nextLine();
40.         Team team = new Team(teamId, teamName, ground, listOfPlayer);
41.         if (choice.equalsIgnoreCase("y")) {
42.             try {
43.                 addNewPlayerToTeam(scanner, team);
44.             } catch (ParseException | IncorrectFormatException e) {
45.                 System.out.println("Add player fail, because" + e.getMessage());
46.                 e.printStackTrace();
47.                 throw e;
48.             }
49.         }
50.         return team;
51.     }
52.
53.     /**
54.      * This method add player(s) to exist team.
55.      *
56.      * @param scanner
57.      * @param team
58.      * @return Boolean
59.      * @throws ParseException, IncorrectFormatException
60.      */
61.     public Boolean addNewPlayerToTeam(Scanner scanner, Team team)
62.                                         throws ParseException, IncorrectFormatException {
63.         Player player = null;
64.         System.out.print("\nInput new player information!");
65.         player = playerService.inputPlayer(scanner);
66.         return team.getListOfPlayer().add(player);
67.     }
68.
69.     /**
70.      * This method display all team's information.
```

```
71.      *
72.      * @param teams:List<Team>.
73.      */
74.     public void displayTeam(List<Team> teams) {
75.         System.out.println("Team information!");
76.         for (Team team : teams) {
77.             System.out.println(team.toString());
78.         }
79.     }
80. }
```

» Step6. Create package **r2s.training.app** that contains **Test** classes:

Test class:

```
1.  package r2s.training.app;
2.
3.  import java.util.ArrayList;
4.  import java.util.List;
5.  import java.util.Scanner;
6.
7.  import r2s.training.entities.Team;
8.  import r2s.training.service.TeamService;
9.
10. public class Test {
11.     public static List<Team> teams = new ArrayList<>();
12.     static TeamService teamService = new TeamService();
13.     static Scanner scanner = new Scanner(System.in);
14.
15.     public static void main(String[] args) {
16.         Team team = null;
17.         String key = "3";
18.         Loop: do {
19.             showMenu();
20.             key = scanner.nextLine();
21.             switch (key) {
22.             case "1":
23.                 try {
24.                     team = teamService.createNewTeam(scanner);
25.                     teams.add(team);
26.                     System.out.println("Create team success!");
27.                     continue Loop;
28.                 } catch (Exception e) {
29.                     System.err.println("Create team fail!");
30.                     break Loop;
31.                 }
32.             case "2":
33.                 teamService.displayTeam(teams);
34.                 break;
35.             default:
36.                 break Loop;
37.               }
38.         } while (true);
39.     }
40.
41.     public static void showMenu() {
42.         System.out.println(">> Menu");
43.         System.out.println("1. Create new team");
44.         System.out.println("2. Display list of team");
45.         System.out.println("3. Exit");
46.         System.out.print(">> Enter your choice: ");
47.     }
48. }
```

Outputs:

Select 1: Create new team.

» Trường hợp valid data

```
>> Menu
1. Create new team
2. Display list of team
3. Exit
>> Enter your choice: 1
Input team information!
Enter team ID:T01
Enter full name of team:Manchester City
Enter ground:FA
Do you want to add new player for this team? (y/n)y

Input new player information!
Enter full name: Rooney

Enter date of birth(dd/mm/yyyy): 12/2/1983

Enter squad number: 10

Enter position( GK|CB|WB|CM|CF):CF
Create team success!
```

» Invalid birth of date

```
>> Menu
1. Create new team
2. Display list of team
3. Exit
>> Enter your choice: 1
Input team information!
Enter team ID:T01
Enter full name of team:Chelsea
Enter ground:FA
Do you want to add new player for this team? (y/n)y

Input new player information!
Enter full name: Hazad

Enter date of birth(dd/mm/yyyy): 40/14/1983
Add player fail!
fa.training.exception.IncorrectFormatException: Date value incorrect format!
        at fa.training.entities.Player.setDateOfBirth(Player.java:59)
        at fa.training.service.PlayerService.inputPlayer(PlayerService.java:30)
        at fa.training.service.TeamService.addNewPlayerToTeam(TeamService.java:64)
        at fa.training.service.TeamService.createNewTeam(TeamService.java:43)
        at fa.training.app.Test.main(Test.java:24)
Create team fail!
```

Invalid birth of position

```
>> Menu
1. Create new team
2. Display list of team
3. Exit
>> Enter your choice: 1
Input team information!
Enter team ID:T01
Enter full name of team:Manchester City
Enter ground:FA
Do you want to add new player for this team? (y/n)y

Input new player information!
Enter full name: Company

Enter date of birth(dd/mm/yyyy): 2/9/1988

Enter squad number: 11

Enter position( GK|CB|WB|CM|CF):CV
Add player fail!
fa.training.exception.IncorrectFormatException: Position value incorrect format!
        at fa.training.entities.Player.setPosition(Player.java:89)
        at fa.training.service.PlayerService.inputPlayer(PlayerService.java:36)
        at fa.training.service.TeamService.addNewPlayerToTeam(TeamService.java:64)
        at fa.training.service.TeamService.createNewTeam(TeamService.java:43)
        at fa.training.app.Test.main(Test.java:24)
Create team fail!
```

Bài tập: Xây dựng một ứng dụng quản lý giao dịch ATM của một ngân hàng. Các tính năng của ứng dụng ngân hàng như sau:

- Mở một tài khoản
- Gửi tiền vào tài khoản
- Rút tiền từ tài khoản
- Hiển thị số lần giao dịch

Khi mở một tài khoản người dùng cần cung cấp các thông tin như tên, giới tính, ngày sinh, nơi sinh, số điện thoại, email.

Khi thực hiện gửi tiền vào tài khoản, người dùng phải nhập số tài khoản và số tiền muốn gửi. Số dư tài khoản = số dư hiện tại + số tiền muốn gửi.

Khi thực hiện rút tiền từ tài khoản, người dùng phải nhập số tài khoản và số tiền muốn rút. Nếu số tiền muốn rút lớn hơn hoặc bằng số dư, chương trình sẽ hiển thị thông báo. Ngược lại số dư tài khoản = số dư hiện tại – số tiền đã rút.

Bên dưới là một số hình ảnh miêu tả chương trình khi chạy

Menu của chương trình

```
Open an account (enter: open)
Make a deposit (enter: deposit)
Make a withdraw (enter: withdraw)
Show the number of transactions (enter: show)
Exit the ATM (enter: quit)

Your choice:
```

Khi người dùng chọn "open"

```
Your choice: open
Enter your name: Le Hong Ky
Enter your gender (male|female): male
Enter your birthday (dd/mm/yyyy): 16/02/1984
Enter your address: HCM
Enter your phone: 0919.365.363
Enter your email: lehongky@yahoo.com
-----Successful operation!-----
BankAccount{accNumber=001, name=Le Hong Ky, gender=male, birthday=16/02/1984, address=HCM,
```

Khi người dùng chọn "deposit"

```
Your choice: deposit
Enter your account number: 001
Enter your amount: 3000
Your current balance is 4000$
-----Successful operation!-----
```

Khi người dùng chọn "withdraw"

Giao dịch thành công

```
Your choice: withdraw
Enter your account number: 001
Enter your amount: 1000
Your current balance is 3000$
```

Giao dịch thất bại

```
Your choice: withdraw
Enter your account number: 001
Enter your amount: 3000

Sorry, but you are short 3000 $
```

Khi người dùng chọn "show": hiển thị số lần giao dịch bao gồm gửi và rút tiền

```
Your choice: show
Enter your account number: 001
The number of transactions are 2
```

Lưu ý:

1. Sử dụng try catch để xử lý nhập dữ liệu kiểu số như số tiền gửi, số tiền rút
2. Sử dụng MyException để hiển thị thông báo khi số tiền rút lớn hơn hoặc bằng số dư hiện tại
3. Cho phép quản lý nhiều tài khoản