Advanced Programming

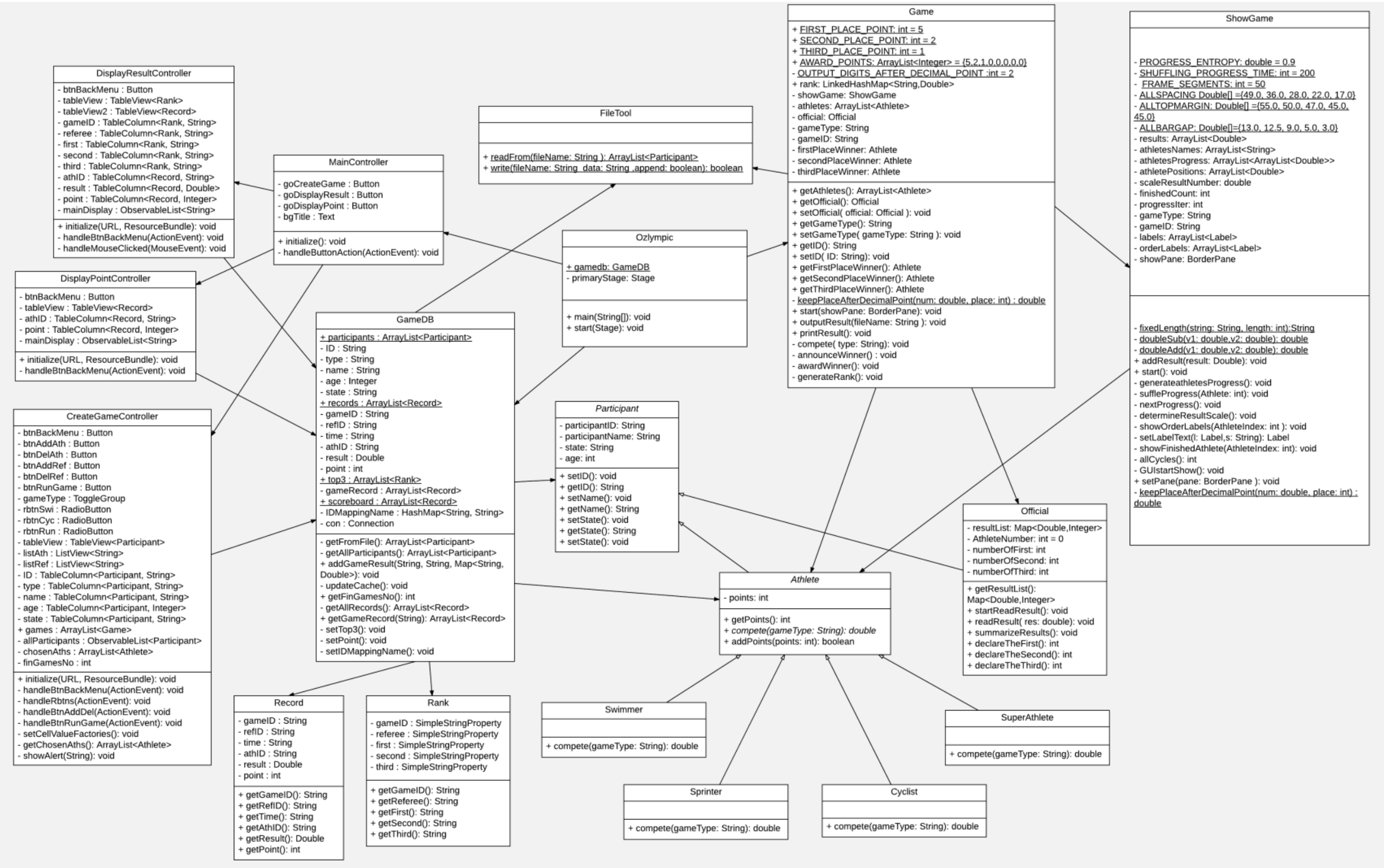
Assignment 2

Lecturer:	Andy Song	
Students:	Ying-Chieh Huang	s3598781
	Yuan-Hao Liu	s3583320

Course number: COSC1295

Date of submission: 17/05/2017

Class Diagram:


DisplayResultController
- btnBackMenu : Button
- tableView : TableView<Rank>
- tableView2 : TableView<Record>
- gameID : TableColumn<Rank, String>
- referee : TableColumn<Rank, String>
- first : TableColumn<Rank, String>
- second : TableColumn<Rank, String>
- third : TableColumn<Rank, String>
- athID : TableColumn<Record, String>
- result : TableColumn<Record, Double>
- point : TableColumn<Record, Integer>
- mainDisplay : ObservableList<String>
+ initialize(URL, ResourceBundle): void
- handleBtnBackMenu(ActionEvent): void
- handleMouseClicked(MouseEvent): void
MainController
- goCreateGame : Button
- goDisplayResult : Button
- goDisplayPoint : Button
- bgTitle : Text
+ initialize(): void
- handleButtonAction(ActionEvent): void
FileTool
+ readFrom(fileName: String): ArrayList<Participant>
+ write(fileName: String data: String ,append: boolean): boolean
Game
+ FIRST_PLACE_POINT: int = 5
+ SECOND_PLACE_POINT: int = 2
+ THIRD_PLACE_POINT: int = 1
+ AWARD_POINTS: ArrayList<Integer> = {5,2,1,0,0,0,0,0}
- OUTPUT_DIGITS_AFTER_DECIMAL_POINT :int = 2
+ rank: LinkedHashMap<String,Double>
- showGame: ShowGame
- athletes: ArrayList<Athlete>
- official: Official
- gameType: String
- gameID: String
- firstPlaceWinner: Athlete
- secondPlaceWinner: Athlete
- thirdPlaceWinner: Athlete
+ getAthletes(): ArrayList<Athlete>
+ getOfficial(): Official
+ setOfficial(official: Official): void
+ getGameType(): String
+ setGameType(gameType: String): void
+ getID(): String
+ setID(ID: String): void
+ getFirstPlaceWinner(): Athlete
+ getSecondPlaceWinner(): Athlete
+ getThirdPlaceWinner(): Athlete
- keepPlaceAfterDecimalPoint(num: double, place: int) : double
+ start(showPane: BorderPane): void
+ outputResult(fileName: String): void
+ printResult(): void
- compete(type: String): void
- announceWinner() : void
- awardWinner(): void
- generateRank(): void
ShowGame
- PROGRESS_ENTROPY: double = 0.9
- SHUFFLING_PROGRESS_TIME: int = 200
- FRAME_SEGMENTS: int = 50
- ALLSPACING Double[] ={49.0, 36.0, 28.0, 22.0, 17.0}
- ALLTOPMARGIN: Double[] ={55.0, 50.0, 47.0, 45.0, 45.0}
- ALLBARGAP: Double[]={13.0, 12.5, 9.0, 5.0, 3.0}
- results: ArrayList<Double>
- athletesNames: ArrayList<String>
- athletesProgress: ArrayList<ArrayList<Double>>
- athletePositions: ArrayList<Double>
- scaleResultNumber: double
- finishedCount: int
- progressIter: int
- gameType: String
- gameID: String
- labels: ArrayList<Label>
- orderLabels: ArrayList<Label>
- showPane: BorderPane
- fixedLength(string: String, length: int):String
- doubleSub(v1: double,v2: double): double
- doubleAdd(v1: double,v2: double): double
+ addResult(result: Double): void
+ start(): void
- generateathletesProgress(): void
- suffleProgress(Athlete: int): void
- nextProgress(): void
- determineResultScale(): void
- showOrderLabels(AthleteIndex: int): void
- setLabelText(l: Label,s: String): Label
- showFinishedAthlete(AthleteIndex: int): void
- allCycles(): int
- GUIstartShow(): void
+ setPane(pane: BorderPane): void
- keepPlaceAfterDecimalPoint(num: double, place: int) : double
DisplayPointController
- btnBackMenu : Button
- tableView : TableView<Record>
- athID : TableColumn<Record, String>
- point : TableColumn<Record, Integer>
- mainDisplay : ObservableList<String>
+ initialize(URL, ResourceBundle): void
- handleBtnBackMenu(ActionEvent): void
Ozlympic
+ gamedb: GameDB
- primaryStage: Stage
+ main(String[]): void
+ start(Stage): void
GameDB
+ participants : ArrayList<Participant>
- ID : String
- type : String
- name : String
- age : Integer
- state : String
+ records : ArrayList<Record>
- gameID : String
- refID : String
- time : String
- athID : String
- result : Double
- point : int
+ top3 : ArrayList<Rank>
- gameRecord : ArrayList<Record>
+ scoreboard : ArrayList<Record>
- IDMappingName : HashMap<String, String>
- con : Connection
- getFromFile(): ArrayList<Participant>
- getAllParticipants(): ArrayList<Participant>
+ addGameResult(String, String, Map<String, Double>): void
- updateCache(): void
+ getFinGamesNo(): int
- getAllRecords(): ArrayList<Record>
+ getGameRecord(String): ArrayList<Record>
- setTop3(): void
- setPoint(): void
- setIDMappingName(): void
Participant
- participantID: String
- participantName: String
- state: String
- age: int
+ setID(): void
+ getID(): String
+ setName(): void
+ getName(): String
+ setState(): void
+ getState(): String
+ setState(): void
CreateGameController
- btnBackMenu : Button
- btnAddAth : Button
- btnDelAth : Button
- btnAddRef : Button
- btnDelRef : Button
- btnRunGame : Button
- gameType : ToggleGroup
- rbtnSwi : RadioButton
- rbtnCyc : RadioButton
- rbtnRun : RadioButton
- tableView : TableView<Participant>
- listAth : ListView<String>
- listRef : ListView<String>
- ID : TableColumn<Participant, String>
- type : TableColumn<Participant, String>
- name : TableColumn<Participant, String>
- age : TableColumn<Participant, Integer>
- state : TableColumn<Participant, String>
+ games : ArrayList<Game>
- allParticipants : ObservableList<Participant>
- chosenAths : ArrayList<Athlete>
- finGamesNo : int
+ initialize(URL, ResourceBundle): void
- handleBtnBackMenu(ActionEvent): void
- handleRbtns(ActionEvent): void
- handleBtnAddDel(ActionEvent): void
- handleBtnRunGame(ActionEvent): void
- setCellValueFactories(): void
- getChosenAths(): ArrayList<Athlete>
- showAlert(String): void
Official
- resultList: Map<Double,Integer>
- AthleteNumber: int = 0
- numberOfFirst: int
- numberOfSecond: int
- numberOfThird: int
+ getResultList(): Map<Double,Integer>
+ startReadResult(): void
+ readResult(res: double): void
+ summarizeResults(): void
+ declareTheFirst(): int
+ declareTheSecond(): int
+ declareTheThird(): int
Athlete
- points: int
+ getPoints(): int
+ compete(gameType: String): double
+ addPoints(points: int): boolean
Record
- gameID : String
- refID : String
- time : String
- athID : String
- result : Double
- point : int
+ getGameID(): String
+ getRefID(): String
+ getTime(): String
+ getAthID(): String
+ getResult(): Double
+ getPoint(): int
Rank
- gameID : SimpleStringProperty
- referee : SimpleStringProperty
- first : SimpleStringProperty
- second : SimpleStringProperty
- third : SimpleStringProperty
+ getGameID(): String
+ getReferee(): String
+ getFirst(): String
+ getSecond(): String
+ getThird(): String
Swimmer
+ compete(gameType: String): double
SuperAthlete
+ compete(gameType: String): double
Sprinter
+ compete(gameType: String): double
Cyclist
+ compete(gameType: String): double

Question 1. Explain the changes if you use a different design compared to your assignment 1

Answer:

Basically the main structure is same as assignment 1, just changed displaying from console to GUI. All user input would be captured by mouse clicking instead of keyboard typing.

Question 2. Explain how the new classes are organized

Answer:

We tried to categorise classes to MVC structure which fxml files and controllers are in one package separated from main classes, all test classes in one source folder, and all exception classes in one package.

Question 3. Explain the process by which your program will interact with user and external data source to run a game.

Answer:

In the first place user would see a simple menu which provides 3 functions – create a new game, display history result, and display athlete point. After user choose a group of athletes, game type and 1 official, then user could directly press the run game button. Therefore, an animation of athletes' competition will display in real-time game simulation.

After game ran finished, the result will be store into database and text file both. All participant data or history record will be read from database and store in an ArrayList as cache.

Contribution:

Ying-Chieh Huang (s3598781): Ozlympic, GameDB, Record, Rank, createGameController, DisplayResultController, DisplayPointController classes, all exception classes, most scenes, and documentation.

Yuan-Hao Liu (s3583320): Game, Participant, Official, Athlete, SuperAthlete, Swimmer, Cyclist, Sprinter, MainController, ShowGame classes, all test cases, and CSS styling.

Comment:

This assignment is more difficult than we thought in the beginning time, and JavaFX scene builder is hard to utilise neither. Now we learned how to apply the MVC structure and JUnit test in the project. In the end, we managed to produce this user friendly program.