THE TRIALS

S1-Project_30

Weriwoh Mbang Goddy Clinton Junior
Weriwohmbang.g@students.salle.url.edu

INDEX

1. Summary of specifications

The project is based on trying to do a competition whereby teams participate. These competitions are separated in yearly editions and each edition has trials which are used to screen players of a participating team, till the result, if at least one is left to win and in the contrary the team loses.

There are 4 types of trials per edition, which can repeat. Paper publication, master studies, doctoral thesis, and budget request.

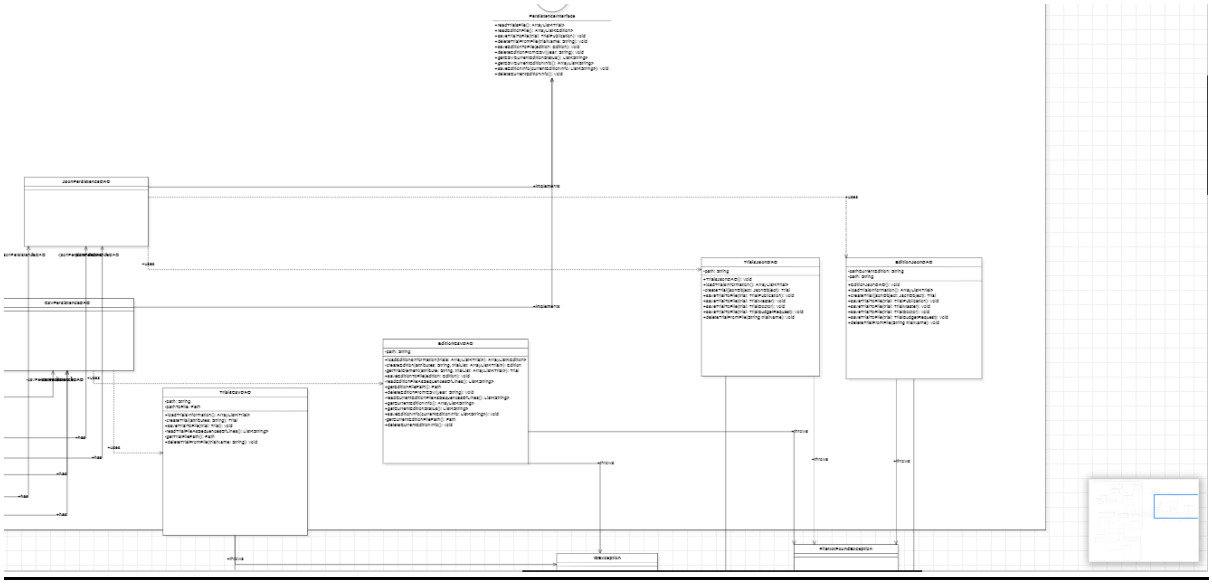
In the composer portion of the program, the user will oversee creating the trials and editions, specifying the various properties asked on the console, which vary depending on trials.

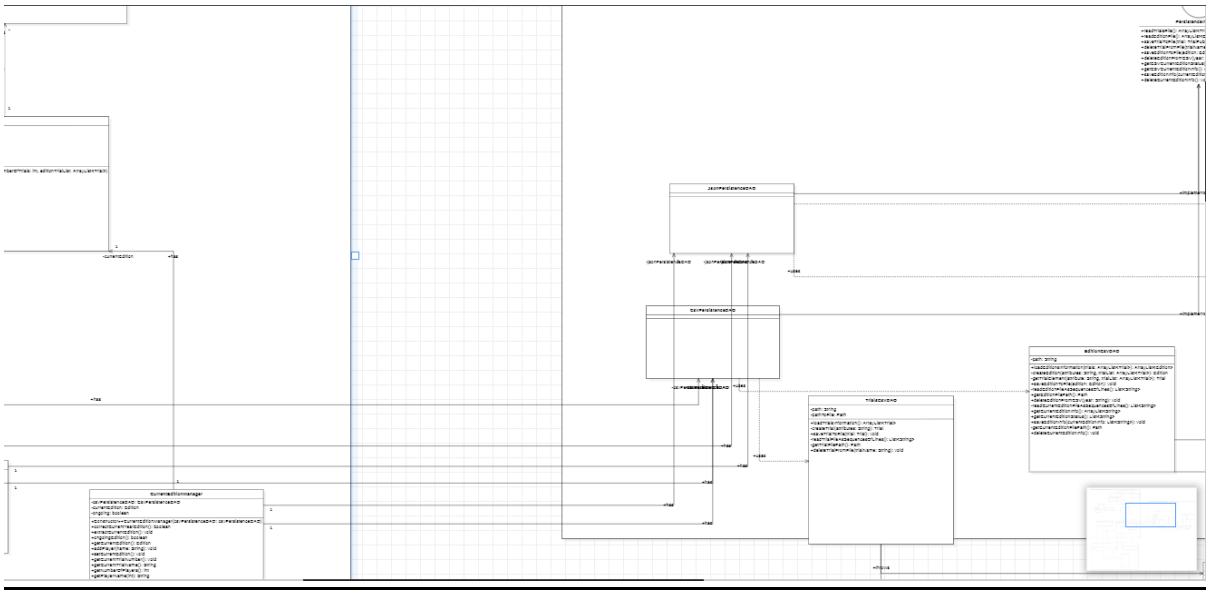
In the conductor part, the players when going through these stages will be evolving from initially engineers, then to masters, and finally doctors, depending on if they reach a PI of 10 while at a certain level, always initially starting with 5 PI. And when a player is a master, they loose just half of PI when they are to lose points, and when doctors, they combine their previous master power to gaining double PI when about to gain points.

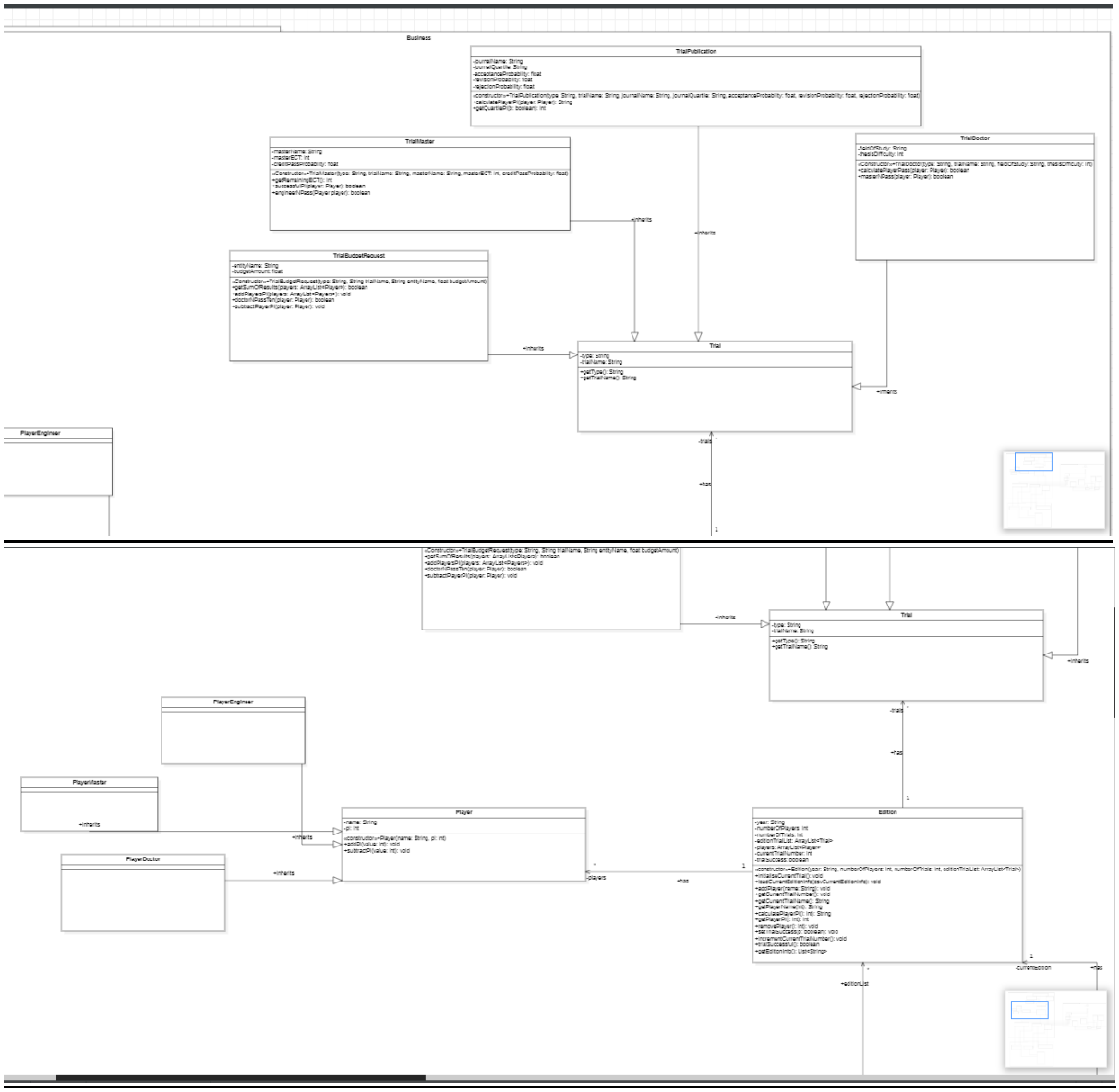
For the trials,

- Paper publication: Here, depending on quartile type, the player will receive or loose several points, when asked to play.
- Master studies: Has number of credits, that users must pass depending on some probability
- Doctoral thesis: Here, the player to pass there is an algorithm to pass a field of study of a certain difficulty.

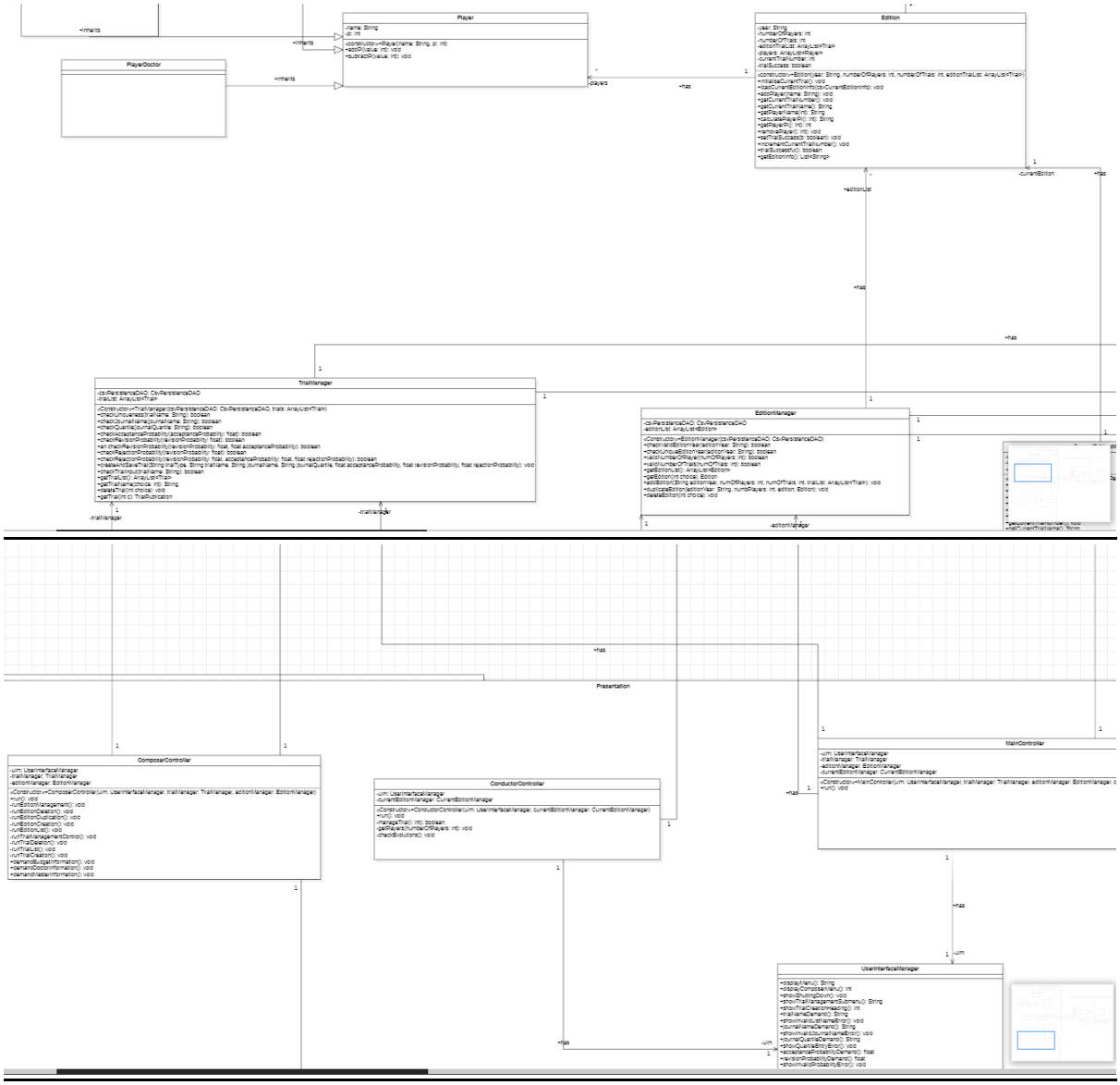
2. Class Diagram






Business
TrialPublication
-journalName: String
-journalQuartile: String
-acceptanceProbability: float
-revisionProbability: float
-rejectionProbability: float
+constructor+TrialPublication(type: String, trialName: String, journalName: String, journalQuartile: String, acceptanceProbability: float, revisionProbability: float, rejectionProbability: float)
+calculatePlayerPI(player: Player): String
+getQuartileRI(): boolean(): int
TrialMaster
-masterName: String
-masterECT: int
-creditPassProbability: float
+Constructor+TrialMaster(type: String, trialName: String, masterName: String, masterECT: int, creditPassProbability: float)
+getRemainingECT(): int
+SuccessfulPI(player: Player): boolean
+engineerPIPass(Player player): boolean
TrialDoctor
-FacOfStudy: String
-thesisDifficulty: int
+Constructor+TrialDoctor(type: String, trialName: String, facOfStudy: String, thesisDifficulty: int)
+calculatePlayerPass(player: Player): boolean
+masterPIPass(player: Player): boolean
TrialBudgetRequest
-entityName: String
-budgetAmount: float
+Constructor+TrialBudgetRequest(type: String, String trialName, String entityName, float budgetAmount)
+getSumOfResults(players: ArrayListPlayer>): boolean
+addPlayersPI(players: ArrayListPlayer>): void
+doctorPIPassTrial(player: Player): boolean
+subtractPlayerPI(player: Player): void
Trial
-type: String
-trialName: String
+getType(): String
+getTrialName(): String
PlayerEngineer
+inherits
-trials
+has
1

+Constructor+TrialBudgetRequest(type: String, String trialName, String entityName, float budgetAmount)
+getSumOfResults(players: ArrayListPlayer>): boolean
+addPlayersPI(players: ArrayListPlayer>): void
+doctorPIPassTrial(player: Player): boolean
+subtractPlayerPI(player: Player): void
+inherits
Trial
-type: String
-trialName: String
+getType(): String
+getTrialName(): String
PlayerEngineer
PlayerMaster
PlayerDoctor
Player
-name: String
-pi: int
+constructor+Player(name: String, pi: int)
+addPI(value: int): void
+subtractPI(value: int): void
Edition
-year: String
-numberOfPlayers: int
-numberOfTrials: int
-editionTrialList: ArrayListTrial>
-players: ArrayListPlayer>
-currentTrialNumber: int
-trialSuccess: boolean
+constructor+Edition(year: String, numberOfPlayers: int, numberOfTrials: int, editionTrialList: ArrayListTrial>)
+initialiseCurrentTrial(): void
+addCurrentEditionPlayersToCurrentEditionInfo(): void
+addPlayer(name: String): void
+getCurrentTrialNumber(): void
+getCurrentTrialName(): String
+getPlayerName(int): String
+calculatePlayerPI(): int): String
+getPlayerPI(): int): int
+removePlayer(): int): void
+setTrialSuccess(b: boolean): void
+incrementCurrentTrialNumber(): void
+trialSuccessful(): boolean
+getEditionInfo(): List<String>
+inherits
+has
-players
-trials
-currentEdition
+editionList
1



3. Class diagram explanation

First and foremost, I divided my code into Business, Persistence, Presentation for a better and more organised structure to reduce coupling and increase cohesion between classes, thereby using MVC modelling. Some Business classes(managers) basically serve as link between some Presentation classes (controllers) with the Persistence classes.

Presentation Package

Here, we can see that I have three controllers, this is to ease my code visibility and space out methods associated to who the user chooses to be, by divide the composer processes from conductor processes (conductor and composer controllers). The main controller decides when to use which controller, depending on the user initial choice to be a composer or a conductor hence the dependency relationship, because it basically just calls the appropriate controller to pass to it, the information it will need.

- We see that composer controller is in a one-way association relationship with the edition, trial and currentEdition manager classes so to pass them to the corresponding controllers when need.
- We see also that the conductor controller is in a one-way association relationship with the current edition manager which will communicate with the edition class and manage the competition (hence the 1:1 relationship, as we manage one competition at the time) and at the same time will access the Persistence DAO so to retrieve or send information to the files when needed.
- Composer controller is in a one to one, one-way relationship with trial and edition manager classes so as to manage the creation and configuration of trials and editions hence the 1:N relationship with these classes. And can access the persistence DAO, to store or retrieve data from files.
- All these controllers are in a 1:1 one-way association relationship with the user interface manager for displaying of information on the screen each time data is required from or to the user, which is indispensable.

Business Package

Contains our entities with their various managers and basically serves as a link between the Presentation and Persistence packages through managers.

- We see the inheritance relationships of the different player types with player class, and with trial types with the trial class, this is to override the common classes these types have with their parent classes and add methods that are particular to each type.
- We notice the one-way 1:N relationship between edition and trial because at the running of the competition, the edition will need the list of trials to function. Same scenario between edition and player class.
- Each other entity is related to its managers depending on the reasons explained in presentation package.

Persistence Package

Contains the classes that permit the access to the storage files.

- We notice the one to one relationship between manager classes and the CSV and JSON persistence classes, which both implement the Persistence Interface. This is because depending on storage method the user chooses to do at very start.
- The Persistence Interface, which has methods which are implemented by csv persistence DAO and this DAO uses at some point trial CSV DAO and edition CSV DAO to store data to CSV files, hence the dependency relationship between them.
- Also, the Persistence Interface, which has methods which are implemented by JSON persistence DAO and this DAO uses at some point trial JSON DAO and edition JSON DAO to store data to JSON files, hence the dependency relationship between them.

4. Dedication

50 hours spent.

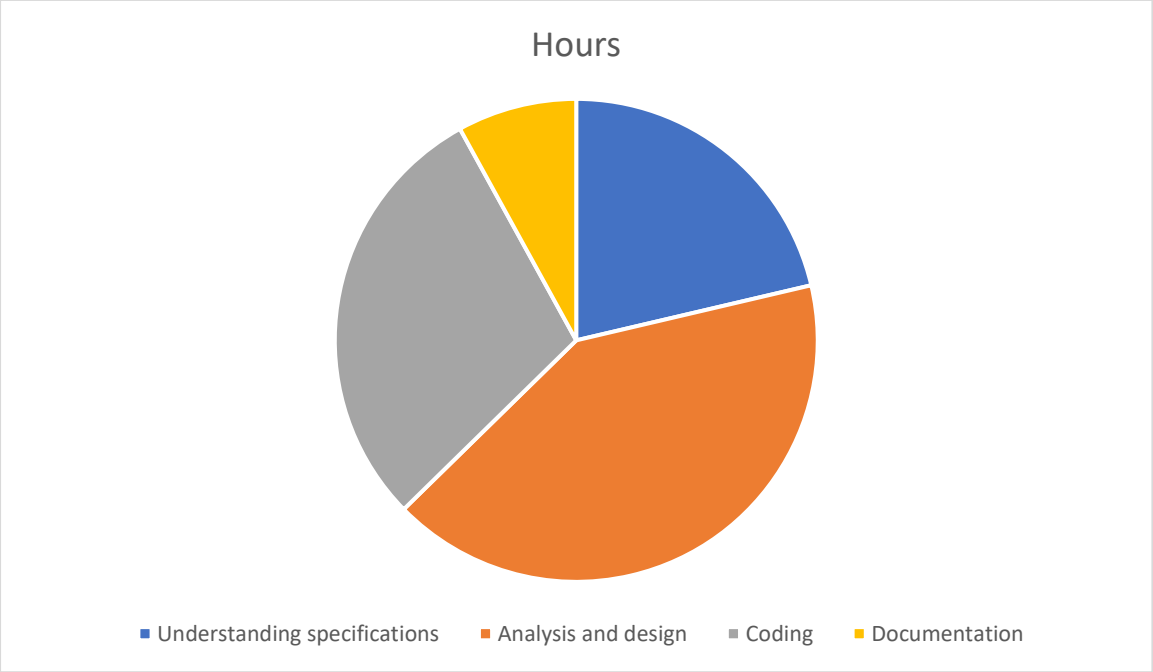
(1) understanding the specifications – 8hours

(2) analysis and design – 25hours

(3) coding – 15hours

(4) documentation – 2hours

5. Pie Chart



2.1 Conclusions

The project helped me understand the use of Json files mostly, how to code efficiently and most importantly, how to follow the MVC method. To understand most java functions, how to do research also. At personal level, it helped me to focus more on the resolution of problem than the problem itself. In general, this project was a combination of efficiency, thought and discovery.

3 Bibliography

https://www.tutorialspoint.com/how-to-read-parse-json-array-using-java

https://crunchify.com/how-to-use-gson-fromjson-to-convert-the-specified-json-into-an-object-of-the-specified-class/

https://www.microfocus.com/documentation/silk-performer/210/en/silkperformer-210-webhelp-en/GUID-6AFC32B4-6D73-4FBA-AD36-E42261E2D77E.html

https://stackoverflow.com/questions/39689507/how-to-parse-json-array-from-file