Functional Requirements.

1. Add turn types. The program let the administer create turn types with a name and a duration in minutes.

2. Show system date. When the program shows the main menu, in it also show the date and hour. Besides, on the menu options, the option “6. Show system time.” shows on screen the actual system time.

3. Update system date manually. Entering a date and time with a specified format, the program updates his date to the entered.

4. Generate report of user’s turns. The administrator chooses where the program will generate the report (on console or text file), then enter the id type and id of the user that want to generate report.

5. Generate report of turn’s users. The administrator chooses where the program will generate the report (on console or text file), then enter the code of the turn that want to generate report.

6. Suspend a inactive user. When a user stays away in his call for two times, the administrator can suspend the user for two days.

7. Generate random users. The administrator can generate a quantity of random users.

8. Generate random turns. The administrator can generate a quantity of random turns per a quantity of days

Non-functional requirements.

1. For each attended turn will wait 15 seconds between turns.

2. The date only can be updated if the entered date is newest than the date system is.

3. Date and time class must be defined in the model (Class Dte).

4. The system must be persistent with serializable.

5. The generation of random users must take values of a text file downloaded of internet.

6. When the administrator choose an option, the program must show the time that action take to be done.

7. The program must implement the methods of sort bubble, selection and insertion.

    Bubble: Class User, method bubbleSortTurnsByCode().

    Insertion: Class TurnManager, method insertionSortTurnsByBuiltD().

    Selection: Class Controller, method selectionSortUsersByIdType().

8. The program must implement binary and sequential search.

    Binary: Class User, method binarySearchTurn(String code).

Sequential: Class Controller, method searchUser(String id, String doc).

9. The program must implement 5 sorts using the method sort() of the class Arrays.

Class Controller and TurnManager, methods:

sortUsersById()

sortUsersByName()

sortUsersByIdInvert()

sortUserBySurname()

sortTurnsByType()

sortTurnsByDuration()

sortTurnsByCode()

sortTurnsByType()

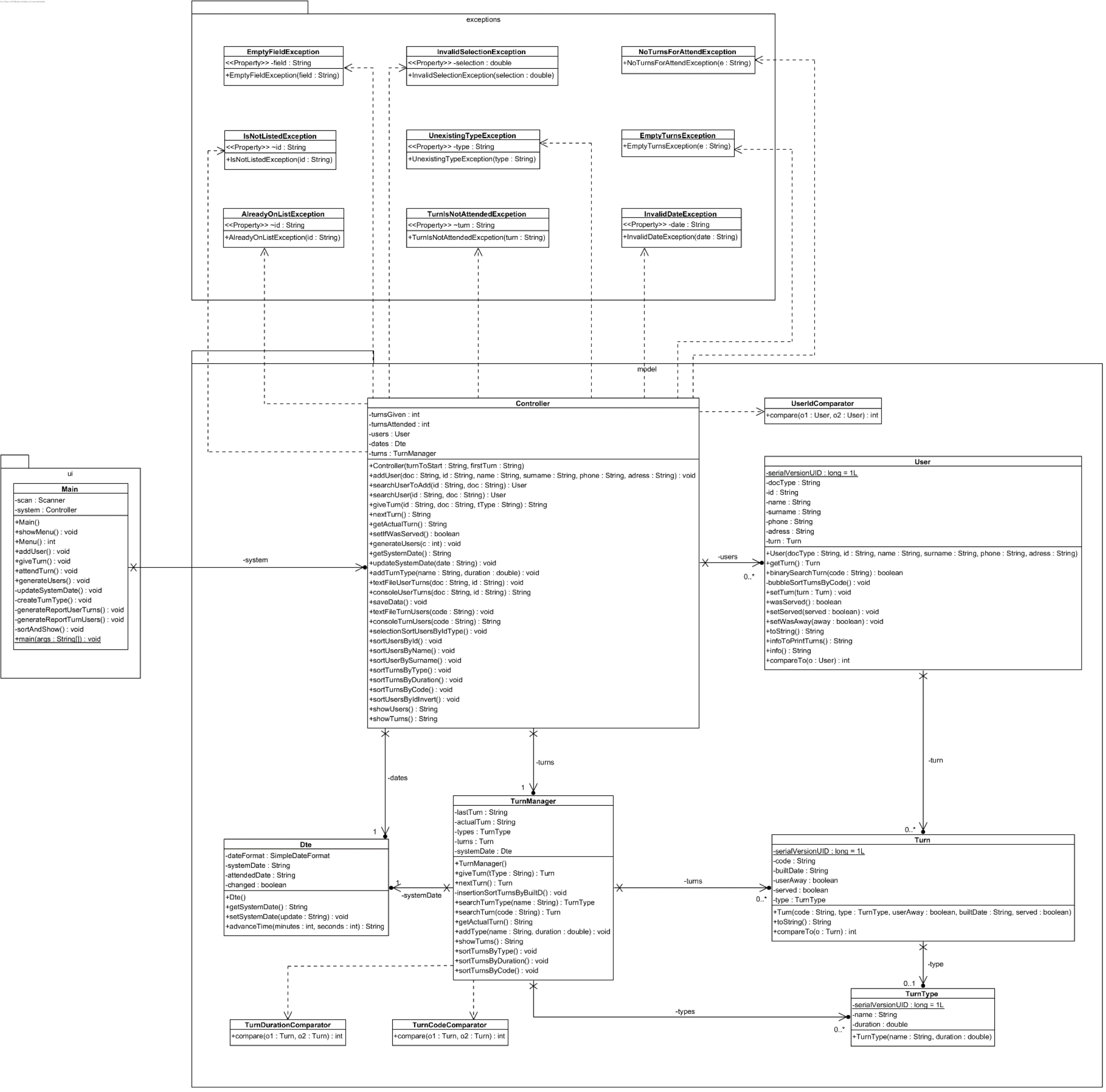
10. The program must use the interfaces Comparable and Comparator, Comparable on a class of the model, Comparator on an external class, Comparator on an anonym class and Comparator inverted of the method reverseOrder(Comparator) of the class Collections.

Comparable: Class User and Turn;

Comparator external class: Class UserIdComparator, TurnDurationComparator and TurnCodeComparator.

Comparator anonym class: Class Controller method sortUserBySurname().

Comparator inverted: Class Controller method sortUsersByIdInvert().


exceptions
EmptyFieldException
<<Property>> -field : String
+EmptyFieldException(field : String)
InvalidSelectionException
<<Property>> -selection : double
+InvalidSelectionException(selection : double)
NoTurnsForAttendException
+NoTurnsForAttendException(e : String)
IsNotListedException
<<Property>> ~id : String
+IsNotListedException(id : String)
UnexistingTypeException
<<Property>> -type : String
+UnexistingTypeException(type : String)
EmptyTurnsException
+EmptyTurnsException(e : String)
AlreadyOnListException
<<Property>> ~id : String
+AlreadyOnListException(id : String)
TurnIsNotAttendedExcpetion
<<Property>> ~turn : String
+TurnIsNotAttendedExcpetion(turn : String)
InvalidDateException
<<Property>> -date : String
+InvalidDateException(date : String)
model
Controller
-turnsGiven : int
-turnsAttended : int
-users : User
-dates : Dte
-turns : TurnManager
+Controller(turnToStart : String, firstTurn : String)
+addUser(doc : String, id : String, name : String, surname : String, phone : String, adress : String) : void
+searchUserToAdd(id : String, doc : String) : User
+searchUser(id : String, doc : String) : User
+giveTurn(id : String, doc : String, tType : String) : String
+nextTurn() : String
+getActualTurn() : String
+setIfWasServed() : boolean
+generateUsers(c : int) : void
+getSystemDate() : String
+updateSystemDate(date : String) : void
+addTurnType(name : String, duration : double) : void
+textFileUserTurns(doc : String, id : String) : void
+consoleUserTurns(doc : String, id : String) : String
+saveData() : void
+textFileTurnUsers(code : String) : void
+consoleTurnUsers(code : String) : String
+selectionSortUsersByIdType() : void
+sortUsersById() : void
+sortUsersByName() : void
+sortUserBySurname() : void
+sortTurnsByType() : void
+sortTurnsByDuration() : void
+sortTurnsByCode() : void
+sortUsersByIdInvert() : void
+showUsers() : String
+showTurns() : String
UserIdComparator
+compare(o1 : User, o2 : User) : int
User
-serialVersionUID : long = 1L
-docType : String
-id : String
-name : String
-surname : String
-phone : String
-adress : String
-turn : Turn
+User(docType : String, id : String, name : String, surname : String, phone : String, adress : String)
+getTurn() : Turn
+binarySearchTurn(code : String) : boolean
-bubbleSortTurnsByCode() : void
+setTurn(turn : Turn) : void
+wasServed() : boolean
+setServed(served : boolean) : void
+setWasAway(away : boolean) : void
+toString() : String
+infoToPrintTurns() : String
+info() : String
+compareTo(o : User) : int
-users
0..*
-turn
0..*
ui
Main
-scan : Scanner
-system : Controller
+Main()
+showMenu() : void
+Menu() : int
+addUser() : void
+giveTurn() : void
+attendTurn() : void
+generateUsers() : void
-updateSystemDate() : void
-createTurnType() : void
-generateReportUserTurns() : void
-generateReportTurnUsers() : void
-sortAndShow() : void
+main(args : String[]) : void
-system
-dates
1
-turns
1
TurnManager
-lastTurn : String
-actualTurn : String
-types : TurnType
-turns : Turn
-systemDate : Dte
+TurnManager()
+giveTurn(tType : String) : Turn
+nextTurn() : Turn
-insertionSortTurnsByBuiltD() : void
+searchTurnType(name : String) : TurnType
+searchTurn(code : String) : Turn
+getActualTurn() : String
+addType(name : String, duration : double) : void
+showTurns() : String
+sortTurnsByType() : void
+sortTurnsByDuration() : void
+sortTurnsByCode() : void
Dte
-dateFormat : SimpleDateFormat
-systemDate : String
-attendedDate : String
-changed : boolean
+Dte()
+getSystemDate() : String
+setSystemDate(update : String) : void
+advanceTime(minutes : int, seconds : int) : String
1
-systemDate
-turns
0..*
Turn
-serialVersionUID : long = 1L
-code : String
-builtDate : String
-userAway : boolean
-served : boolean
-type : TurnType
+Turn(code : String, type : TurnType, userAway : boolean, builtDate : String, served : boolean)
+toString() : String
+compareTo(o : Turn) : int
-type
0..1
TurnType
-serialVersionUID : long = 1L
-name : String
-duration : double
+TurnType(name : String, duration : double)
-types
0..*
TurnDurationComparator
+compare(o1 : Turn, o2 : Turn) : int
TurnCodeComparator
+compare(o1 : Turn, o2 : Turn) : int