AccountManager

- companies: HashMap <String, Company>
  //Key: username (i.e. companyName)
- individuals: HashMap <String, Applicant>
  //Key: username
- interviewers: HashMap<String, ArrayList<String>>
  //Key: companyName, value: list of interviewers

+ matchInterviewer(companyName: String, interviewer: String): String
+ getUser(userName: String, userType: String): User
+ registerUser(userName: String, password: String, userType: String): User
+ getInterviewers(companyName: String): ArrayList<String>
+ addInterviewer(companyName: String, interviewer:String): void

Abstract User

- username: String
- password: String

+ User (username: String, password: String, userType: String)
+ setUsername(username: String): void
+ getUsername(): void
+ getPassword(): void
+ setPassword(password: String): void
+ matchPassword(password: String): boolean

JobPostingManager

- openJobPostings: HashMap<String, ArrayList<JobPosting>>
  //key: " position"

- jobPostings HashMap<String, JobPosting>
  //key is "companyName + position + postDate"

- jobPostingsByCompany: HashMap<String, ArrayList<JobPosting>>
  //key is "jobPosting.company", all JobPosting's state is Open

- jobPostingsByCloseDate: HashMap<LocalDate, ArrayList<JobPosting>>

+ addToOpenJobPostings(jp: JobPosting): void
+ addJobPosting(jp: JobPosting): String
+ findJobPosting(key: String): JobPosting
+ searchByCompany(company: String): ArrayList<JobPosting>
+ searchByPosition(position: String): ArrayList<JobPosting>
+ closeJobPostings()

Application implements Observer extends Observable

- heading: String // contains applicant name, position
- attachedDocuments: ArrayList<String> //fileName
- jobPosting: JobPosting
- applicant: Applicant
- interviews: HashMap <String, ArrayList<Interview>>
  //Key: "upcoming", "waiting for result"
- currentState: String
  //one of "incomplete", "forward", "pending", "rejected", "hired"

+ Application(jp: JobPosting, a: Applicant)
+ getHeading(): String
+ getJobPosting(): JobPosting
+ getApplicant(): Applicant
+ getCurrentState(): String
+ setCurrentState(state: String): void
+ getAttachedDocuments(): ArrayList<String (fileName)>
+ addInterview(Interview interview): void
+ findInterviews(key: String): ArrayList<Interview>
+ attachToApplication (fileName: String): String
+ dropApplication
+ moveInterview(in: Interview, moveFrom: String, moveTo: String
+ receiveInterviewResult(result: String): void
+ update(Observable o, object arg)

DocumentManager

+ deleteAfterThirtyDays: HashMap<LocalDate, ArrayList<Applicant>>
+ attachedDocuments: HashMap<String, ArrayList<File>>
  //key is the heading of an application
+ allDocuments: HashMap<String, ArrayList<File>>
  //key is the name of an applicant
+ allFiles: HashMap<String, File>
  //key is file name

+ viewDocument(fileName:String, heading of an application: String): String
+ getAttachedDocuments(headingOfAnApplication: String): ArrayList<File>
+ getMyDocuments(nameOfAnApplicant: String): ArrayList<File>
+ createNewDoc(fileName: String, content: String, applicant: String): String
+ deleteDocument (fileName: String): void
+ updateDocument (fileName: String, content: String): void
+ deleteAllInactiveDocuments(): void
+ removeFromDeleteAfterThirtyDays(ap: Applicant): void
+ addToAttachedDocuments(fileName:String, heading of an application: String): String

Singleton System

- documentManager: DocumentManager
- jobPostingManger: JobPostingManger
- accountManager: AccountManger

+ private constructor

+ static getSystem(): System

+ main()

```
DocumentManager.deleteAllFile();

JobPostingManager.closeJobPostings();
```

Interface JobPostingState

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(application: Application): String

Interviewing implements JobPostingState

- jobPosting: JobPosting

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(a: Application): String

WaitingForNextRoung implements JobPostingState

- jobPosting: JobPosting

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(a: Application): String

Unfilled implements JobPostingState

- jobPosting: JobPosting

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(a: Application): String

Pending implements JobPostingState

- jobPosting: JobPosting

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(a: Application): String

Open implements JobPostingState

- jobPosting: JobPosting

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(a: Application): String

Filled implements JobPostingState

- jobPosting: JobPosting

+ getStatus(): String

+ matchInterview(interviewer, application, date): String

+ hire(a: Application): String

JobPosting implements Observer

- position: String
- numOfPositions: int
- company: Company
- postDate: LocalDate
- closeDate: LocalDate
- requirements: ArrayList <String> //names of required documents

- currentState: JobPostingState
- unmatchedApplicants: int = 0
- numOfRounds: int = 0
- numOfHired: int = 0

- remainingApplications: HashMap<String, Application>
  //Key is the heading of the Application

+ JobPosting(fromTextFileds: ArrayList<Object>, company: Company)
+ getId(): String
+ getCurrentState(): JobPostingState
+ getCloseDate(): LocalDate
+ getNumOfHired(): int
+ getNumOfRounds(): int
+ getRemainingApplications(): ArrayList<Application>
+ setCurrentState(state: JobPostingState): void
+ addNumOfHired(): void
+ addNumOfRounds(): void
+ removeApplication(ap: Application): void
+ receiveApplication(ap: Application): void
  unmatchedApplicants ++
+findApplication(heading: String): Application
+ toString()
+ 3 methods in JobPostingState
+ update(Observable o, Object arg)

Interview extends Observable

- interviewer: String
- application: Application
- date: LocalDate
- currentState: String //"upcoming, pending, rejected, forward"
- heading: String (contains, interviewer, applicant's username at date)
- company: Company

+ Interview(interviewer: String, ap: Application, date: Date)
+ update(String state): void
+ giveRecommendation(result: String): String
  (result is "rejected" or "forward")
+ getDate(): Date
+ toString() //what will show in 3(3)

Applicant extends User implements Observer

- applicationsByState: HashMap <String, ArrayList<Application>>
  //Key: "incomplete", "forward", "pending", "rejected", "hired"

- applications: HashMap <String, Application>
  //Key: application's heading

+ Applicant(username: String, password: String)

+ getApplications(key: String): ArrayList<Application>
+ getApplication(key: String): Application
+ addApplication(app: Application): void
+ moveApplication(app: Application, state: String): void
+ checkActive(): void
+ applyJob(jp: JobPosting, applicant: Applicant):
+ removeApplication(application: Application): void
+ update(Observable o, object arg)

Company extends User

- jobPostingsBySate: HashMap<String, ArrayList<JobPosting>>
  //Key: "open", "interviewing", "waiting for next round", "pending", "filled", "unfilled"

- upcomingInterviews: HashMap<String, ArrayList<Interview>>
  //Key is interviewer
- waitingForResultInterviews: HashMap<String, ArrayList<Interview>>
  //key is interviewer

+ Company(username:String, password: String)
+ getCompanyName(): String

+ assignInterview(interviewer: String, application: Application, date: LocalDate): String
  //create new interview i,
  //add i to application's list
  //return "Interview Matched"

+ findJobPostings(state: String): ArrayList<JobPosting>

+ postJob(ArrayList<Object>): String
  //create new jobPosting jp, JobPostingManager.addJobPosting(jp)
  // add jp to JobPostings
  //return "Job Posted"

+ findUpcomingInterveiws(interviewer: String): ArrayList<Interview>

+ findWaitingForResultInterviews(interviewer: String): ArrayList<Interview>

+ moveInterview(interview: Interview): void
  //move interview from upcomingInterviews to waitingForResultInterviews

+ removeInterview(interview: Interview): void
  //remove interview form waitingForResultInterviews

+ addToUpcomingInterviews(interview: Interview): void