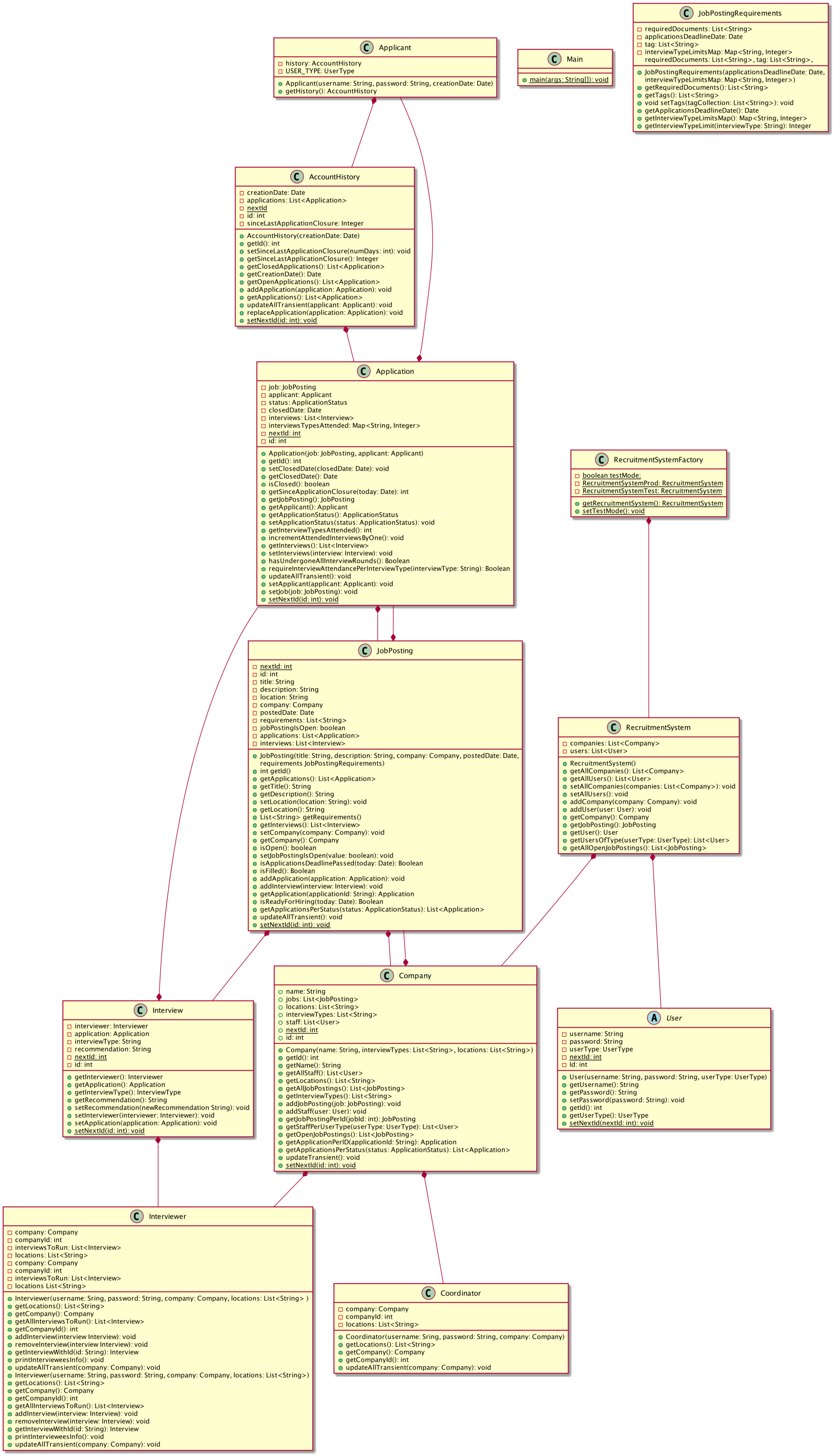
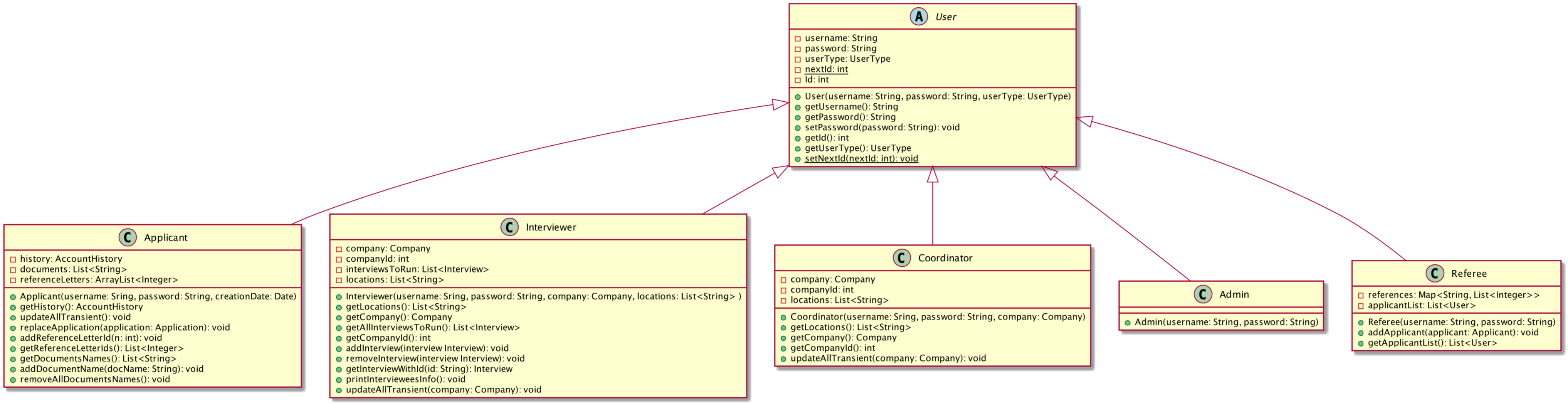
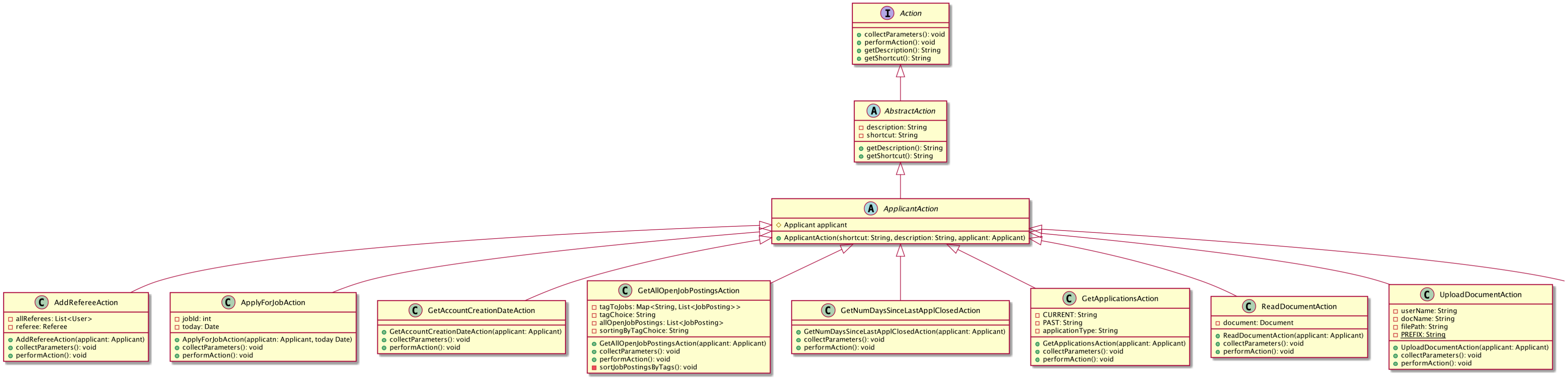
Composition


Applicant
history: AccountHistory
USER_TYPE: UserType
Applicant(username: String, password: String, creationDate: Date)
getHistory(): AccountHistory
Main
main(args: String[]): void
JobPostingRequirements
requiredDocuments: List<String>
applicationsDeadlineDate: Date
tag: List<String>
interviewTypeLimitsMap: Map<String, Integer>
requiredDocuments: List<String>, tag: List<String>,
JobPostingRequirements(applicationsDeadlineDate: Date,
interviewTypeLimitsMap: Map<String, Integer>)
getRequiredDocuments(): List<String>
getTags(): List<String>
void setTags(tagCollection: List<String>): void
getApplicationsDeadlineDate(): Date
getInterviewTypeLimitsMap(): Map<String, Integer>
getInterviewTypeLimit(interviewType: String): Integer
AccountHistory
creationDate: Date
applications: List<Application>
nextId
id: int
sinceLastApplicationClosure: Integer
AccountHistory(creationDate: Date)
getId(): int
setSinceLastApplicationClosure(numDays: int): void
getSinceLastApplicationClosure(): Integer
getClosedApplications(): List<Application>
getCreationDate(): Date
getOpenApplications(): List<Application>
addApplication(application: Application): void
getApplications(): List<Application>
updateAllTransient(applicant: Applicant): void
replaceApplication(application: Application): void
setNextId(id: int): void
Application
job: JobPosting
applicant: Applicant
status: ApplicationStatus
closedDate: Date
interviews: List<Interview>
interviewsTypesAttended: Map<String, Integer>
nextId: int
id: int
Application(job: JobPosting, applicant: Applicant)
getId(): int
setClosedDate(closedDate: Date): void
getClosedDate(): Date
isClosed(): boolean
getSinceApplicationClosure(today: Date): int
getJobPosting(): JobPosting
getApplicant(): Applicant
getApplicationStatus(): ApplicationStatus
setApplicationStatus(status: ApplicationStatus): void
getInterviewTypesAttended(): int
incrementAttendedInterviewsByOne(): void
getInterviews(): List<Interview>
setInterviews(interview: Interview): void
hasUndergoneAllInterviewRounds(): Boolean
requireInterviewAttendancePerInterviewType(interviewType: String): Boolean
updateAllTransient(): void
setApplicant(applicant: Applicant): void
setJob(job: JobPosting): void
setNextId(id: int): void
RecruitmentSystemFactory
boolean testMode;
RecruitmentSystemProd: RecruitmentSystem
RecruitmentSystemTest: RecruitmentSystem
getRecruitmentSystem(): RecruitmentSystem
setTestMode(): void
JobPosting
nextId: int
id: int
title: String
description: String
location: String
company: Company
postedDate: Date
requirements: List<String>
jobPostingIsOpen: boolean
applications: List<Application>
interviews: List<Interview>
JobPosting(title: String, description: String, company: Company, postedDate: Date,
requirements JobPostingRequirements)
int getId()
getApplications(): List<Application>
getTitle(): String
getDescription(): String
setLocation(location: String): void
getLocation(): String
List<String> getRequirements()
getInterviews(): List<Interview>
setCompany(company: Company): void
getCompany(): Company
isOpen(): boolean
setJobPostingIsOpen(value: boolean): void
isApplicationsDeadlinePassed(today: Date): Boolean
isFilled(): Boolean
addApplication(application: Application): void
addInterview(interview: Interview): void
getApplication(applicationId: String): Application
isReadyForHiring(today: Date): Boolean
getApplicationsPerStatus(status: ApplicationStatus): List<Application>
updateAllTransient(): void
setNextId(id: int): void
RecruitmentSystem
companies: List<Company>
users: List<User>
RecruitmentSystem()
getAllCompanies(): List<Company>
getAllUsers(): List<User>
setAllCompanies(companies: List<Company>): void
setAllUsers(): void
addCompany(company: Company): void
addUser(user: User): void
getCompany(): Company
getJobPosting(): JobPosting
getUser(): User
getUsersOfType(userType: UserType): List<User>
getAllOpenJobPostings(): List<JobPosting>
Interview
interviewer: Interviewer
application: Application
interviewType: String
recommendation: String
nextId: int
id: int
getInterviewer(): Interviewer
getApplication(): Application
getInterviewType(): InterviewType
getRecommendation(): String
setRecommendation(newRecommendation String): void
setInterviewer(interviewer: Interviewer): void
setApplication(application: Application): void
setNextId(id: int): void
Company
name: String
jobs: List<JobPosting>
locations: List<String>
interviewTypes: List<String>
staff: List<User>
nextId: int
id: int
Company(name: String, interviewTypes: List<String>, locations: List<String>)
getId(): int
getName(): String
getAllStaff(): List<User>
getLocations(): List<String>
getAllJobPostings(): List<JobPosting>
getInterviewTypes(): List<String>
addJobPosting(job: JobPosting): void
addStaff(user: User): void
getJobPostingPerId(jobId: int): JobPosting
getStaffPerUserType(userType: UserType): List<User>
getOpenJobPostings(): List<JobPosting>
getApplicationPerID(applicationId: String): Application
getApplicationsPerStatus(status: ApplicationStatus): List<Application>
updateTransient(): void
setNextId(id: int): void
User
username: String
password: String
userType: UserType
nextId: int
Id: int
User(username: String, password: String, userType: UserType)
getUsername(): String
getPassword(): String
setPassword(password: String): void
getId(): int
getUserType(): UserType
setNextId(nextId: int): void
Interviewer
company: Company
companyId: int
interviewsToRun: List<Interview>
locations: List<String>
company: Company
companyId: int
interviewsToRun: List<Interview>
locations List<String>
Interviewer(username: Sring, password: String, company: Company, locations: List<String>)
getLocations(): List<String>
getCompany(): Company
getAllInterviewsToRun(): List<Interview>
getCompanyId(): int
addInterview(interview Interview): void
removeInterview(interview Interview): void
getInterviewWithId(id: String): Interview
printIntervieweesInfo(): void
updateAllTransient(company: Company): void
Interviewer(username: String, password: String, company: Company, locations: List<String>)
getLocations(): List<String>
getCompany(): Company
getCompanyId(): int
getAllInterviewsToRun(): List<Interview>
addInterview(interview: Interview): void
removeInterview(interview: Interview): void
getInterviewWithId(id: String): Interview
printIntervieweesInfo(): void
updateAllTransient(company: Company): void
Coordinator
company: Company
companyId: int
locations: List<String>
Coordinator(username: Sring, password: String, company: Company)
getLocations(): List<String>
getCompany(): Company
getCompanyId(): int
updateAllTransient(company: Company): void

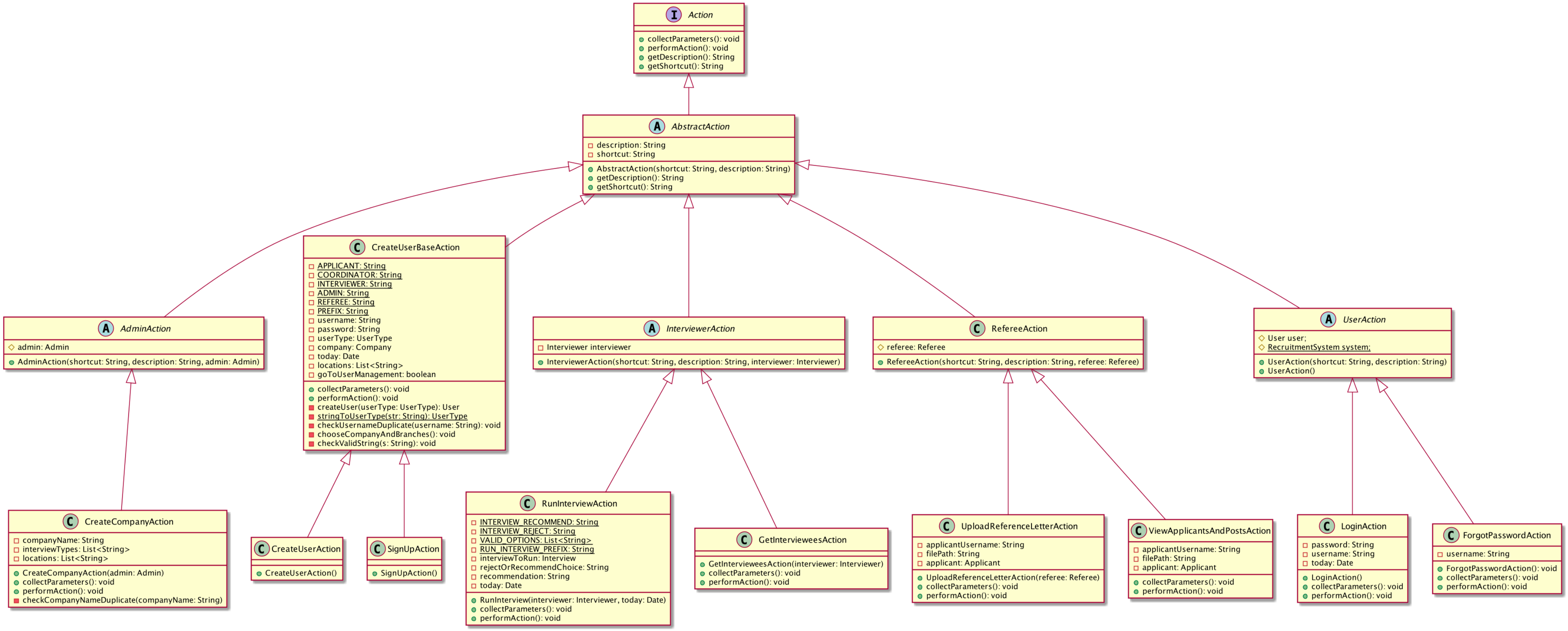
Users


A User
□ username: String
□ password: String
□ userType: UserType
□ nextId: int
□ Id: int
● User(username: String, password: String, userType: UserType)
● getUsername(): String
● getPassword(): String
● setPassword(password: String): void
● getId(): int
● getUserType(): UserType
● setNextId(nextId: int): void
C Applicant
□ history: AccountHistory
□ documents: List<String>
□ referenceLetters: ArrayList<Integer>
● Applicant(username: Sring, password: String, creationDate: Date)
● getHistory(): AccountHistory
● updateAllTransient(): void
● replaceApplication(application: Application): void
● addReferenceLetterId(n: int): void
● getReferenceLetterIds(): List<Integer>
● getDocumentsNames(): List<String>
● addDocumentName(docName: String): void
● removeAllDocumentsNames(): void
C Interviewer
□ company: Company
□ companyId: int
□ interviewsToRun: List<Interview>
□ locations: List<String>
● Interviewer(username: Sring, password: String, company: Company, locations: List<String>)
● getLocations(): List<String>
● getCompany(): Company
● getAllInterviewsToRun(): List<Interview>
● getCompanyId(): int
● addInterview(interview Interview): void
● removeInterview(interview Interview): void
● getInterviewWithId(id: String): Interview
● printIntervieweesInfo(): void
● updateAllTransient(company: Company): void
C Coordinator
□ company: Company
□ companyId: int
□ locations: List<String>
● Coordinator(username: Sring, password: String, company: Company)
● getLocations(): List<String>
● getCompany(): Company
● getCompanyId(): int
● updateAllTransient(company: Company): void
C Admin
● Admin(username: String, password: String)
C Referee
□ references: Map<String, List<Integer>>
□ applicantList: List<User>
● Referee(username: String, password: String)
● addApplicant(applicant: Applicant): void
● getApplicantList(): List<User>

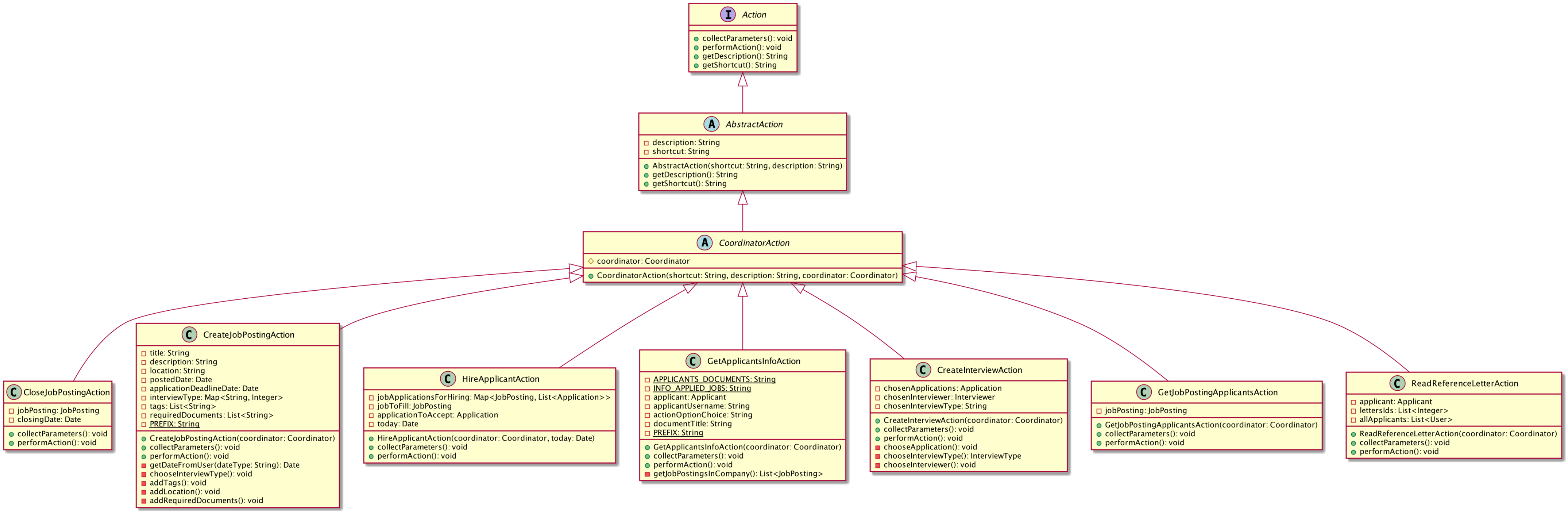
Applicant Actions


I Action
+ collectParameters(): void
+ performAction(): void
+ getDescription(): String
+ getShortcut(): String
A AbstractAction
- description: String
- shortcut: String
+ getDescription(): String
+ getShortcut(): String
A ApplicantAction
# Applicant applicant
+ ApplicantAction(shortcut: String, description: String, applicant: Applicant)
C AddRefereeAction
- allReferees: List<User>
- referee: Referee
+ AddRefereeAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void
C ApplyForJobAction
- jobId: int
- today: Date
+ ApplyForJobAction(applicatn: Applicant, today Date)
+ collectParameters(): void
+ performAction(): void
C GetAccountCreationDateAction
+ GetAccountCreationDateAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void
C GetAllOpenJobPostingsAction
- tagToJobs: Map<String, List<JobPosting>>
- tagChoice: String
- allOpenJobPostings: List<JobPosting>
- sortingByTagChoice: String
+ GetAllOpenJobPostingsAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void
- sortJobPostingsByTags(): void
C GetNumDaysSinceLastApplClosedAction
+ GetNumDaysSinceLastApplClosedAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void
C GetApplicationsAction
- CURRENT: String
- PAST: String
- applicationType: String
+ GetApplicationsAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void
C ReadDocumentAction
- document: Document
+ ReadDocumentAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void
C UploadDocumentAction
- userName: String
- docName: String
- filePath: String
- PREFIX: String
+ UploadDocumentAction(applicant: Applicant)
+ collectParameters(): void
+ performAction(): void

Base & Interviewer & Referee & User Actions

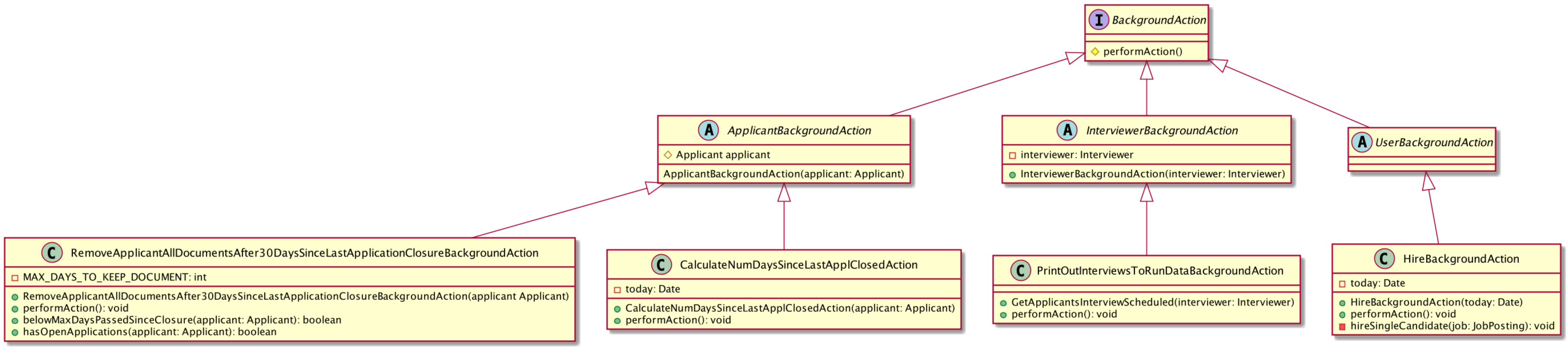

Action
collectParameters(): void
performAction(): void
getDescription(): String
getShortcut(): String
AbstractAction
description: String
shortcut: String
AbstractAction(shortcut: String, description: String)
getDescription(): String
getShortcut(): String
AdminAction
admin: Admin
AdminAction(shortcut: String, description: String, admin: Admin)
CreateUserBaseAction
APPLICANT: String
COORDINATOR: String
INTERVIEWER: String
ADMIN: String
REFEREE: String
PREFIX: String
username: String
password: String
userType: UserType
company: Company
today: Date
locations: List<String>
goToUserManagement: boolean
collectParameters(): void
performAction(): void
createUser(userType: UserType): User
stringToUserType(str: String): UserType
checkUsernameDuplicate(username: String): void
chooseCompanyAndBranches(): void
checkValidString(s: String): void
InterviewerAction
Interviewer interviewer
InterviewerAction(shortcut: String, description: String, interviewer: Interviewer)
RefereeAction
referee: Referee
RefereeAction(shortcut: String, description: String, referee: Referee)
UserAction
User user;
RecruitmentSystem system;
UserAction(shortcut: String, description: String)
UserAction()
CreateCompanyAction
companyName: String
interviewTypes: List<String>
locations: List<String>
CreateCompanyAction(admin: Admin)
collectParameters(): void
performAction(): void
checkCompanyNameDuplicate(companyName: String)
CreateUserAction
CreateUserAction()
SignUpAction
SignUpAction()
RunInterviewAction
INTERVIEW_RECOMMEND: String
INTERVIEW_REJECT: String
VALID_OPTIONS: List<String>
RUN_INTERVIEW_PREFIX: String
interviewToRun: Interview
rejectOrRecommendChoice: String
recommendation: String
today: Date
RunInterview(interviewer: Interviewer, today: Date)
collectParameters(): void
performAction(): void
GetIntervieweesAction
GetIntervieweesAction(interviewer: Interviewer)
collectParameters(): void
performAction(): void
UploadReferenceLetterAction
applicantUsername: String
filePath: String
applicant: Applicant
UploadReferenceLetterAction(referee: Referee)
collectParameters(): void
performAction(): void
ViewApplicantsAndPostsAction
applicantUsername: String
filePath: String
applicant: Applicant
collectParameters(): void
performAction(): void
LoginAction
password: String
username: String
today: Date
LoginAction()
collectParameters(): void
performAction(): void
ForgotPasswordAction
username: String
ForgotPasswordAction(): void
collectParameters(): void
performAction(): void

Coordinator Actions

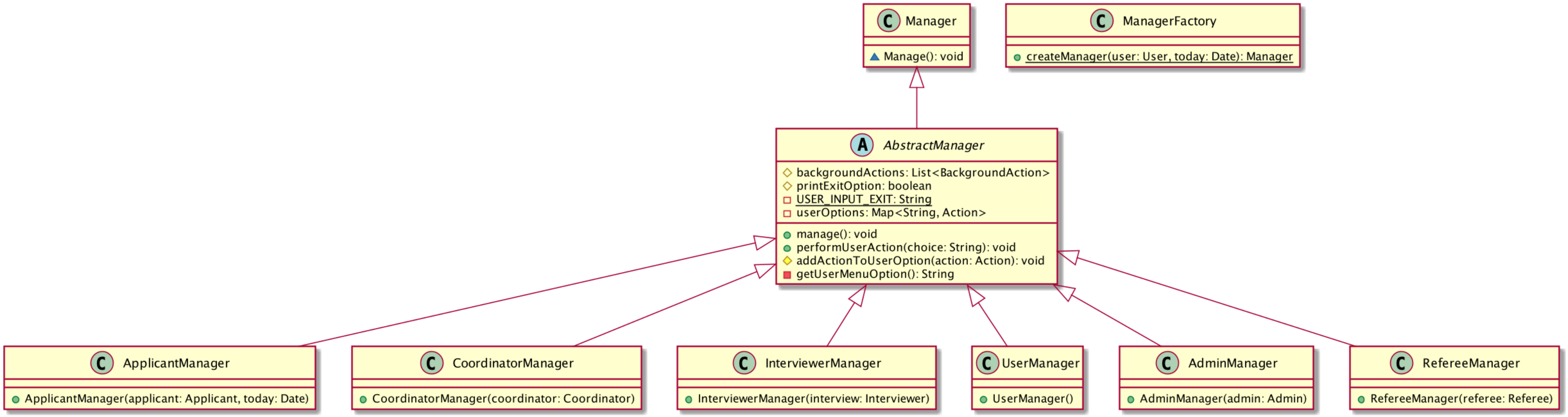


Action
collectParameters(): void
performAction(): void
getDescription(): String
getShortcut(): String
AbstractAction
description: String
shortcut: String
AbstractAction(shortcut: String, description: String)
getDescription(): String
getShortcut(): String
CoordinatorAction
coordinator: Coordinator
CoordinatorAction(shortcut: String, description: String, coordinator: Coordinator)
CloseJobPostingAction
jobPosting: JobPosting
closingDate: Date
collectParameters(): void
performAction(): void
CreateJobPostingAction
title: String
description: String
location: String
postedDate: Date
applicationDeadlineDate: Date
interviewType: Map<String, Integer>
tags: List<String>
requiredDocuments: List<String>
PREFIX: String
CreateJobPostingAction(coordinator: Coordinator)
collectParameters(): void
performAction(): void
getDateFromUser(dateType: String): Date
chooseInterviewType(): void
addTags(): void
addLocation(): void
addRequiredDocuments(): void
HireApplicantAction
jobApplicationsForHiring: Map<JobPosting, List<Application>>
jobToFill: JobPosting
applicationToAccept: Application
today: Date
HireApplicantAction(coordinator: Coordinator, today: Date)
collectParameters(): void
performAction(): void
GetApplicantsInfoAction
APPLICANTS_DOCUMENTS: String
INFO_APPLIED_JOBS: String
applicant: Applicant
applicantUsername: String
actionOptionChoice: String
documentTitle: String
PREFIX: String
GetApplicantsInfoAction(coordinator: Coordinator)
collectParameters(): void
performAction(): void
getJobPostingsInCompany(): List<JobPosting>
CreateInterviewAction
chosenApplications: Application
chosenInterviewer: Interviewer
chosenInterviewType: String
CreateInterviewAction(coordinator: Coordinator)
collectParameters(): void
performAction(): void
chooseApplication(): void
chooseInterviewType(): InterviewType
chooseInterviewer(): void
GetJobPostingApplicantsAction
jobPosting: JobPosting
GetJobPostingApplicantsAction(coordinator: Coordinator)
collectParameters(): void
performAction(): void
ReadReferenceLetterAction
applicant: Applicant
lettersIds: List<Integer>
allApplicants: List<User>
ReadReferenceLetterAction(coordinator: Coordinator)
collectParameters(): void
performAction(): void

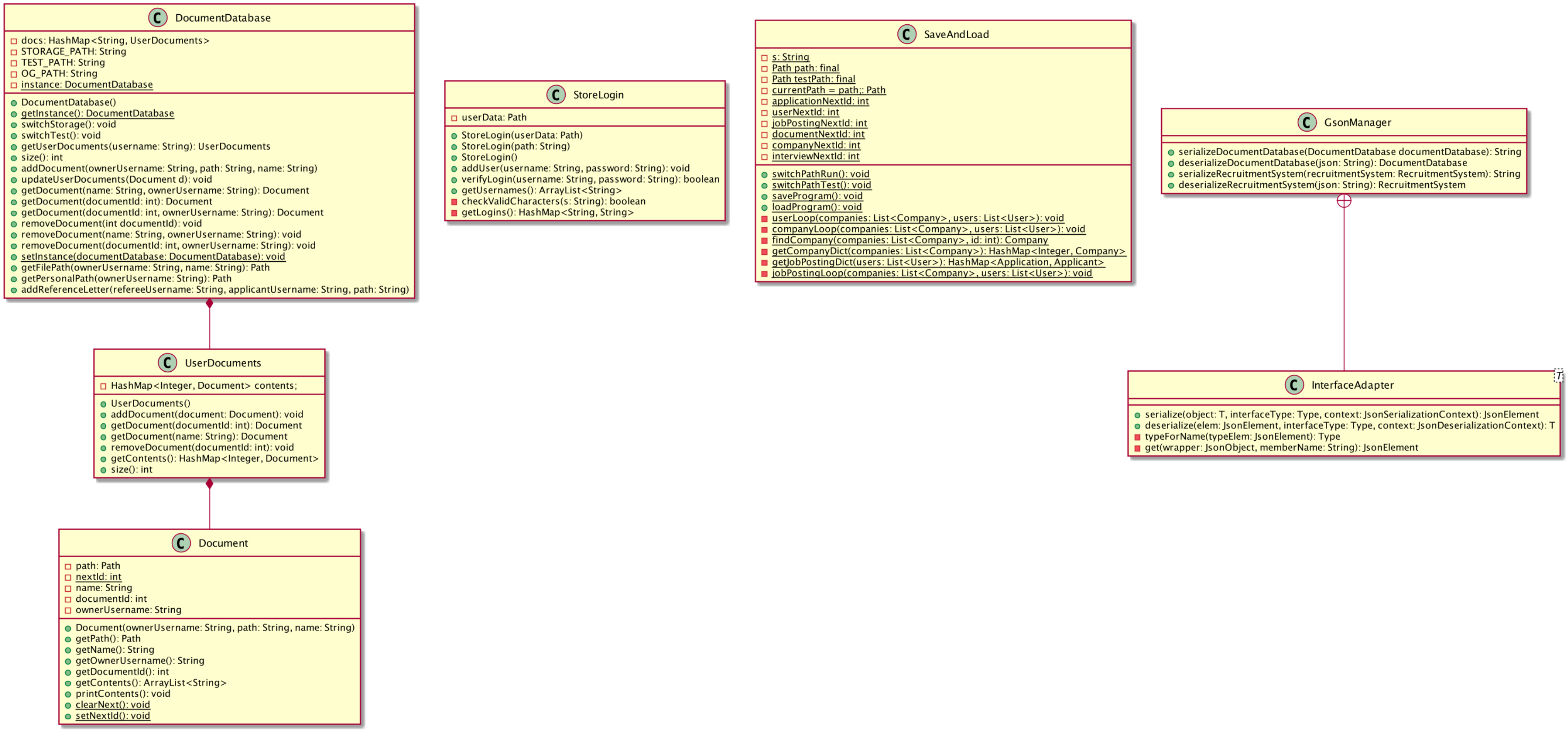
Background Actions


BackgroundAction
◇ performAction()
ApplicantBackgroundAction
◇ Applicant applicant
ApplicantBackgroundAction(applicant: Applicant)
InterviewerBackgroundAction
□ interviewer: Interviewer
● InterviewerBackgroundAction(interviewer: Interviewer)
UserBackgroundAction
RemoveApplicantAllDocumentsAfter30DaysSinceLastApplicationClosureBackgroundAction
□ MAX_DAYS_TO_KEEP_DOCUMENT: int
● RemoveApplicantAllDocumentsAfter30DaysSinceLastApplicationClosureBackgroundAction(applicant Applicant)
● performAction(): void
● belowMaxDaysPassedSinceClosure(applicant: Applicant): boolean
● hasOpenApplications(applicant: Applicant): boolean
CalculateNumDaysSinceLastApplClosedAction
□ today: Date
● CalculateNumDaysSinceLastApplClosedAction(applicant: Applicant)
● performAction(): void
PrintOutInterviewsToRunDataBackgroundAction
● GetApplicantsInterviewScheduled(interviewer: Interviewer)
● performAction(): void
HireBackgroundAction
□ today: Date
● HireBackgroundAction(today: Date)
● performAction(): void
■ hireSingleCandidate(job: JobPosting): void

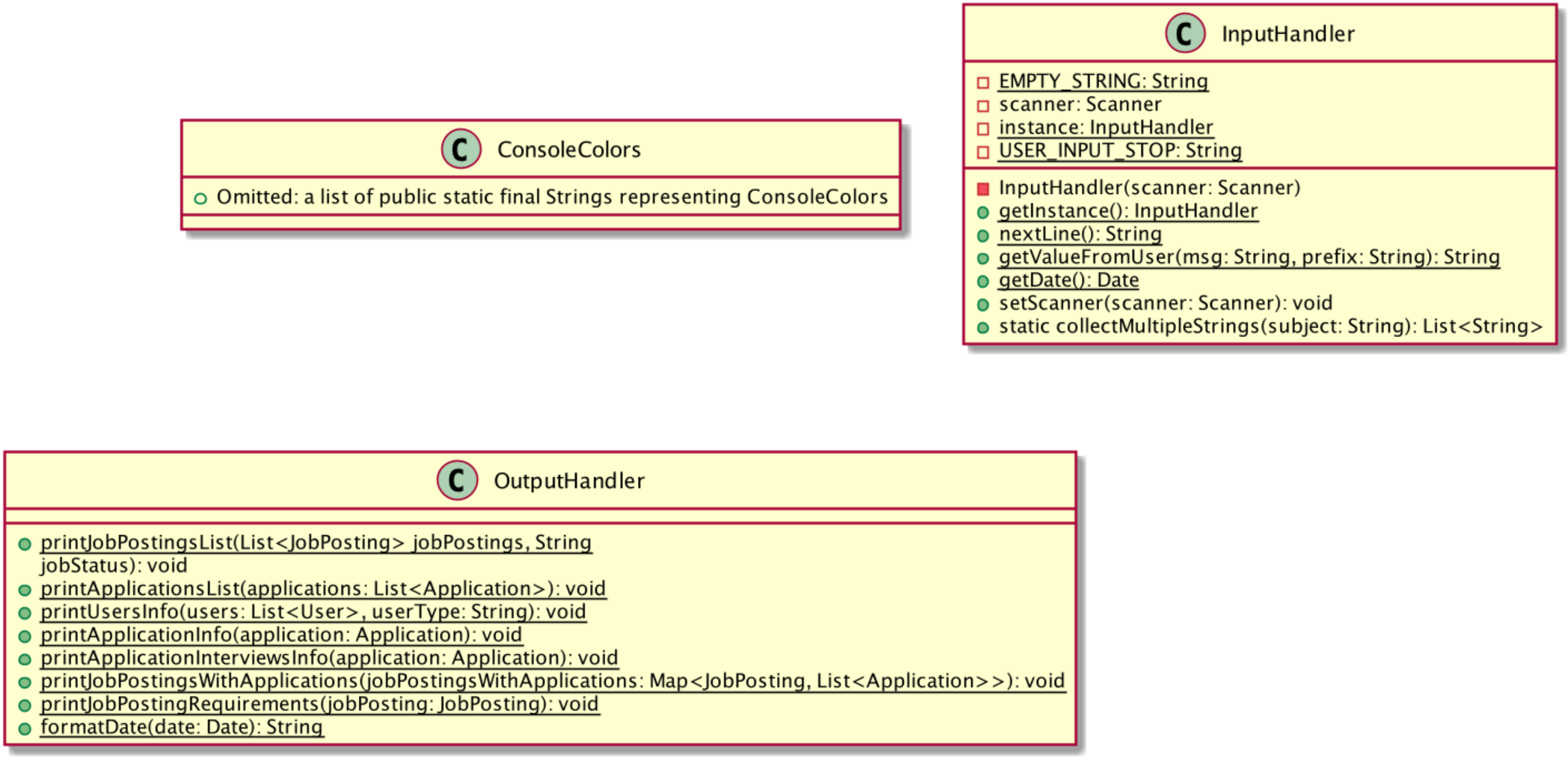
Managers


Manager
▲ Manage(): void
ManagerFactory
● createManager(user: User, today: Date): Manager
AbstractManager
◇ backgroundActions: List<BackgroundAction>
◇ printExitOption: boolean
□ USER_INPUT_EXIT: String
□ userOptions: Map<String, Action>
● manage(): void
● performUserAction(choice: String): void
◆ addActionToUserOption(action: Action): void
■ getUserMenuOption(): String
ApplicantManager
● ApplicantManager(applicant: Applicant, today: Date)
CoordinatorManager
● CoordinatorManager(coordinator: Coordinator)
InterviewerManager
● InterviewerManager(interview: Interviewer)
UserManager
● UserManager()
AdminManager
● AdminManager(admin: Admin)
RefereeManager
● RefereeManager(referee: Referee)

Documents


DocumentDatabase
docs: HashMap<String, UserDocuments>
STORAGE_PATH: String
TEST_PATH: String
OG_PATH: String
instance: DocumentDatabase
DocumentDatabase()
getInstance(): DocumentDatabase
switchStorage(): void
switchTest(): void
getUserDocuments(username: String): UserDocuments
size(): int
addDocument(ownerUsername: String, path: String, name: String)
updateUserDocuments(Document d): void
getDocument(name: String, ownerUsername: String): Document
getDocument(documentId: int): Document
getDocument(documentId: int, ownerUsername: String): Document
removeDocument(int documentId): void
removeDocument(name: String, ownerUsername: String): void
removeDocument(documentId: int, ownerUsername: String): void
setInstance(documentDatabase: DocumentDatabase): void
getFilePath(ownerUsername: String, name: String): Path
getPersonalPath(ownerUsername: String): Path
addReferenceLetter(refereeUsername: String, applicantUsername: String, path: String)
UserDocuments
HashMap<Integer, Document> contents;
UserDocuments()
addDocument(document: Document): void
getDocument(documentId: int): Document
getDocument(name: String): Document
removeDocument(documentId: int): void
getContents(): HashMap<Integer, Document>
size(): int
Document
path: Path
nextId: int
name: String
documentId: int
ownerUsername: String
Document(ownerUsername: String, path: String, name: String)
getPath(): Path
getName(): String
getOwnerUsername(): String
getDocumentId(): int
getContents(): ArrayList<String>
printContents(): void
clearNext(): void
setNextId(): void
StoreLogin
userData: Path
StoreLogin(userData: Path)
StoreLogin(path: String)
StoreLogin()
addUser(username: String, password: String): void
verifyLogin(username: String, password: String): boolean
getUsernames(): ArrayList<String>
checkValidCharacters(s: String): boolean
getLogins(): HashMap<String, String>
SaveAndLoad
s: String
Path path: final
Path testPath: final
currentPath = path;: Path
applicationNextId: int
userNextId: int
jobPostingNextId: int
documentNextId: int
companyNextId: int
interviewNextId: int
switchPathRun(): void
switchPathTest(): void
saveProgram(): void
loadProgram(): void
userLoop(companies: List<Company>, users: List<User>): void
companyLoop(companies: List<Company>, users: List<User>): void
findCompany(companies: List<Company>, id: int): Company
getCompanyDict(companies: List<Company>): HashMap<Integer, Company>
getJobPostingDict(users: List<User>): HashMap<Application, Applicant>
jobPostingLoop(companies: List<Company>, users: List<User>): void
GsonManager
serializeDocumentDatabase(DocumentDatabase documentDatabase): String
deserializeDocumentDatabase(json: String): DocumentDatabase
serializeRecruitmentSystem(recruitmentSystem: RecruitmentSystem): String
deserializeRecruitmentSystem(json: String): RecruitmentSystem
InterfaceAdapter
T
serialize(object: T, interfaceType: Type, context: JsonSerializationContext): JsonElement
deserialize(elem: JsonElement, interfaceType: Type, context: JsonDeserializationContext): T
typeForName(typeElem: JsonElement): Type
get(wrapper: JsonObject, memberName: String): JsonElement

Tools


C ConsoleColors
○ Omitted: a list of public static final Strings representing ConsoleColors
C InputHandler
□ EMPTY_STRING: String
□ scanner: Scanner
□ instance: InputHandler
□ USER_INPUT_STOP: String
■ InputHandler(scanner: Scanner)
● getInstance(): InputHandler
● nextLine(): String
● getValueFromUser(msg: String, prefix: String): String
● getDate(): Date
● setScanner(scanner: Scanner): void
● static collectMultipleStrings(subject: String): List<String>
C OutputHandler
● printJobPostingsList(List<JobPosting> jobPostings, String jobStatus): void
● printApplicationsList(applications: List<Application>): void
● printUsersInfo(users: List<User>, userType: String): void
● printApplicationInfo(application: Application): void
● printApplicationInterviewsInfo(application: Application): void
● printJobPostingsWithApplications(jobPostingsWithApplications: Map<JobPosting, List<Application>>): void
● printJobPostingRequirements(jobPosting: JobPosting): void
● formatDate(date: Date): String