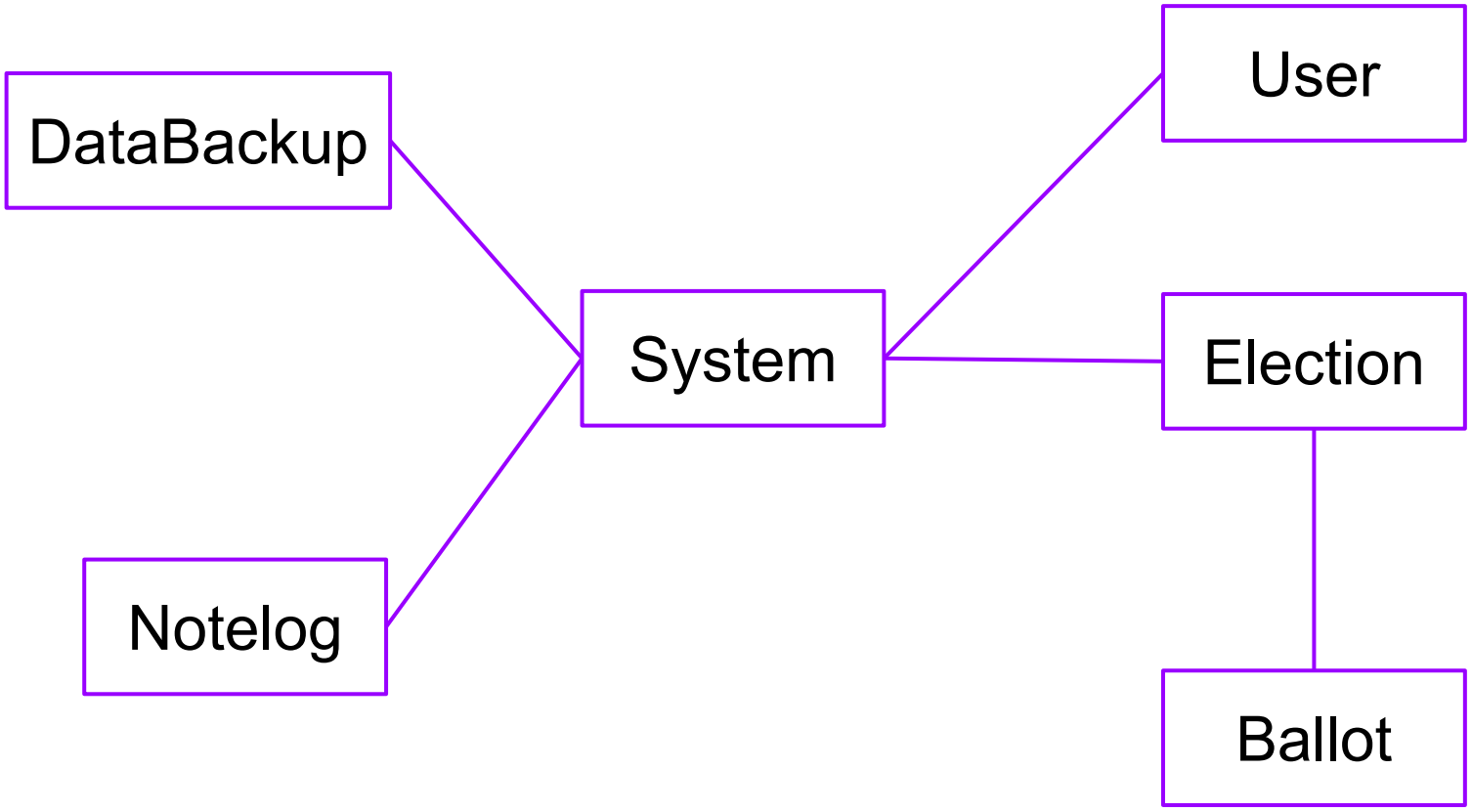
COMP 723 Voting System design

Dinghuang Ji & Tianxiang Gao

Overall Feature

- Basic: user voting, tallying, etc
- Our own features:
  - Multiple Elections
  - Different User Roles and Rights, Privacy Protection
  - Allow modification in Elections and Ballots
  - Notification System
  - Data Backup and Recovery system
- Overall design patterns:
  - Singleton, Flyweight, Chain, Composite, Factory, Observer, AOP...

Module Overview



System Module

- Manage users, elections and data
- Singleton pattern
- Flyweight pattern

DatabackupListener : Thread
timeGap : long

if currentTimeGap>timeGap
this.timeStamp = currentTime;
dbSystem.saveDB(System);

System
- static system: System - electionMap: map<int,Election> - userGroup: map<int,User> - dbSystem: DataBackuper - timeStamp: long
- System() + static getSingleton() : System + register(User, string, string) + login(string, string) : UserId + getUser(UserId) : User + logout(UserId) + createElection() : Election + getElection(int) : Election + updateElection(): Election + closeElection(int) + recovery(): System

User Module

User

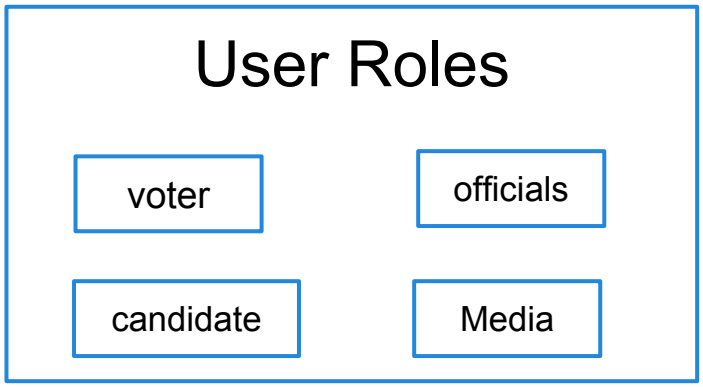
- id: UserId
- roles: int
- SSN: string
- residency: string
- birthday: time
- occupancy : map<int, Election>
- rights: int

+ User(int, int, bool, …)
+ getRole(): int
+ addRole(int)
+ revokeRole(int)
+ getRights(): int
+ addRights(): int
+ removeRights(): int
+ getProperties(string): string
+ checkID(): bool
+ encode()
+ decode()

- Manage User roles, rights and information

```
register (User, string, string)
if exist(userGroup,User.id)
        addRole (User.roles)
else
        userGroup.add(User)
encode(User)
```

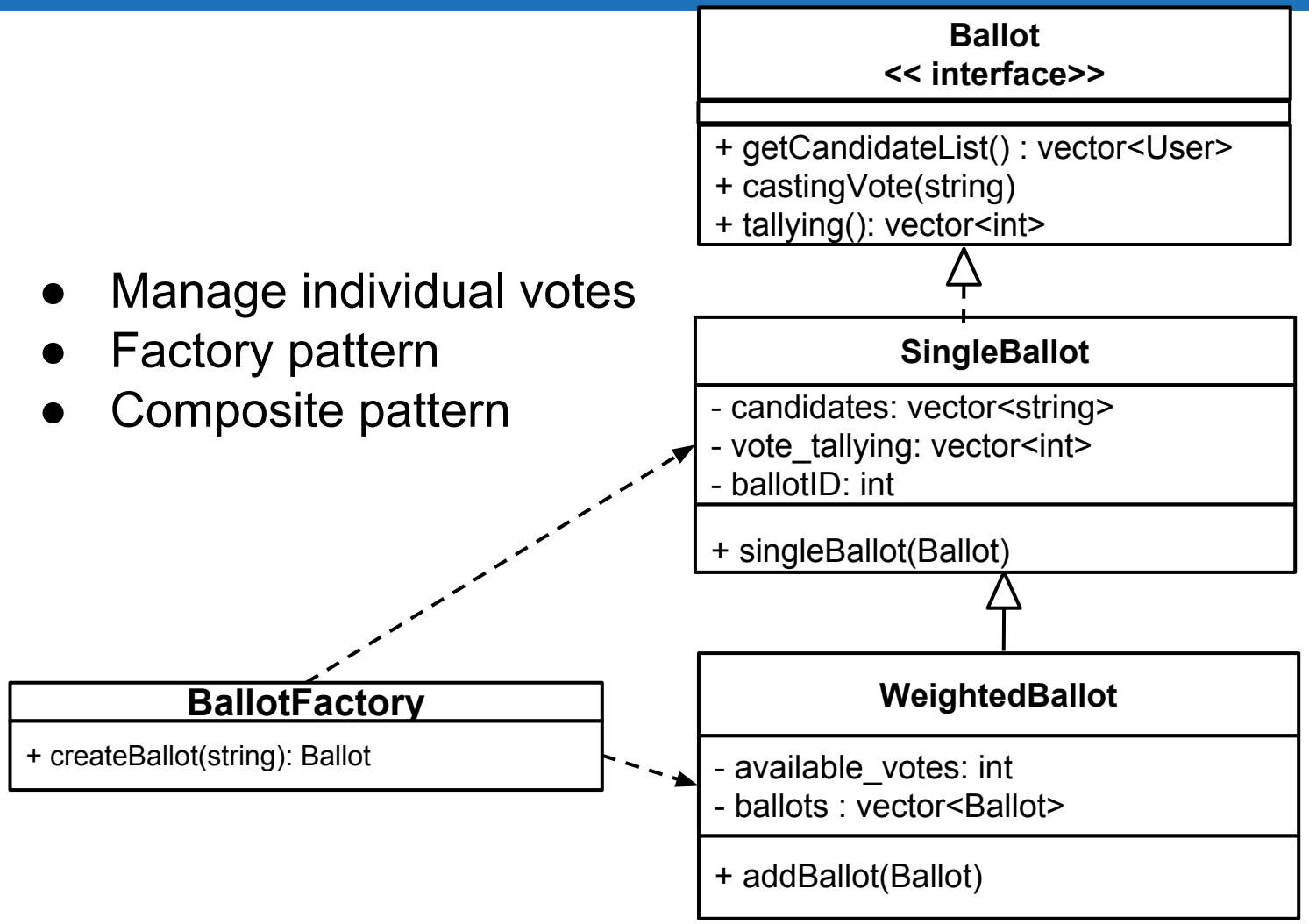
User Roles




```
enum rights = {
vote = 1,
changevote = 2,
revoke = 4,
getresult = 8,
securitycheck = 16
};
```

Ballot

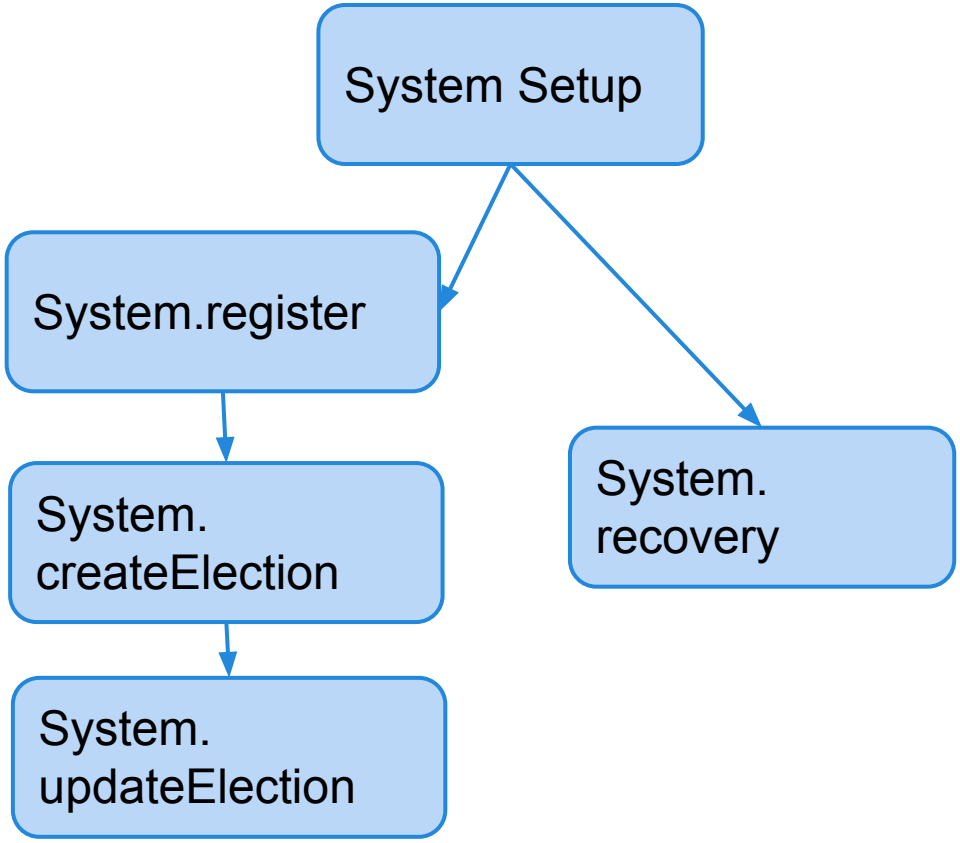
- Manage individual votes
- Factory pattern
- Composite pattern



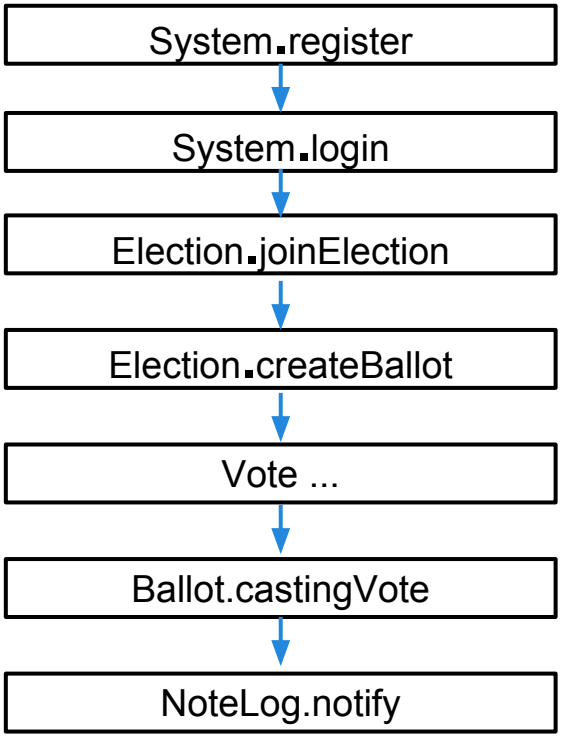
Election

Election
- electionId: int - candidates: vector<User> - ballots: vector<ballots> - status: bool
+ createElectionId(string): + joinElection(User) + createBallot() + addBallot(ballot): + getBallot(ballotID): + delBallot(ballotID) + getAllStats(): string + setCandidateList(vector<User>) + getCandidateList(): vector<User> + setExpired() + isExpired(): bool

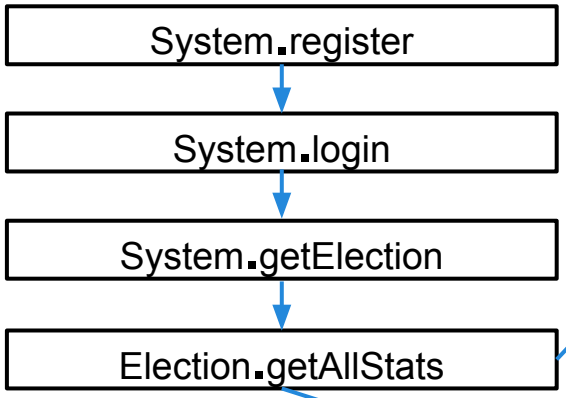
- Manage individual election information



Voting Process



Stats Collection



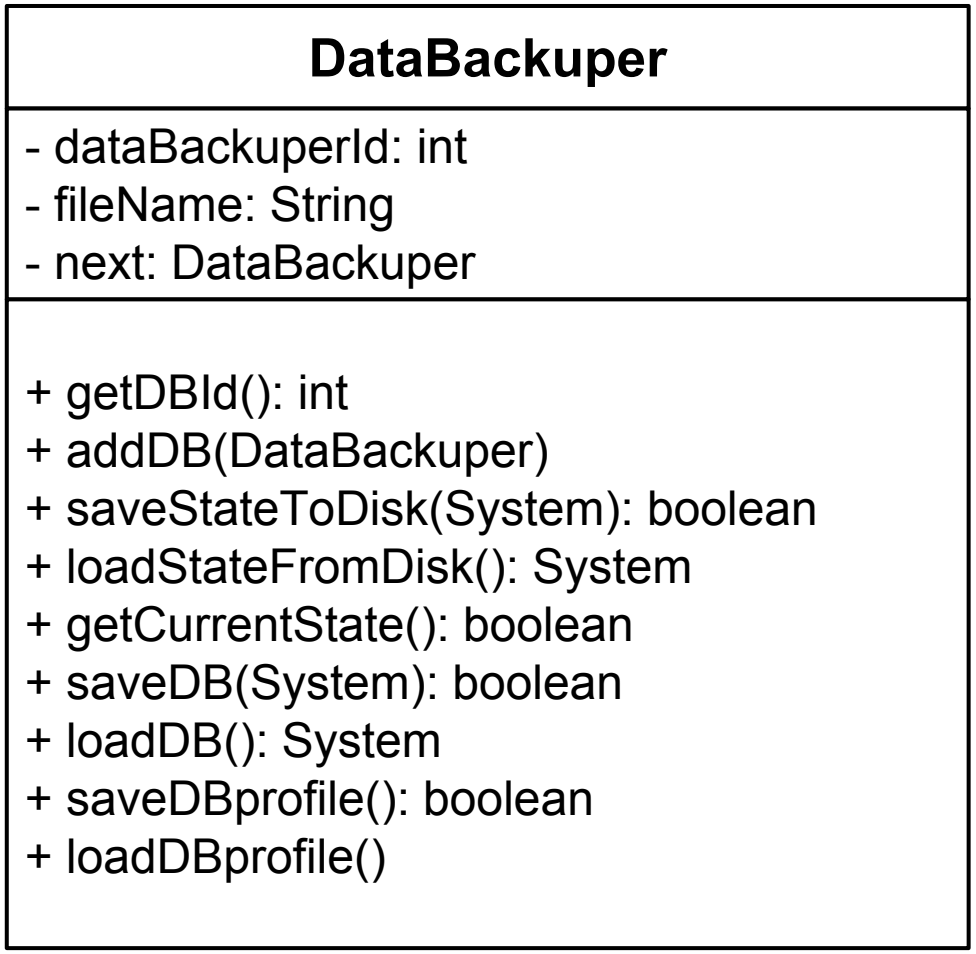
real time data

candidate id	candidate name	total votes	time	is_expired
1	Joe	100	2015.4	False
2	Tom	50	2015.4	

final data

- winner
- votes for all candidates
- voting statistics

DataBackuper



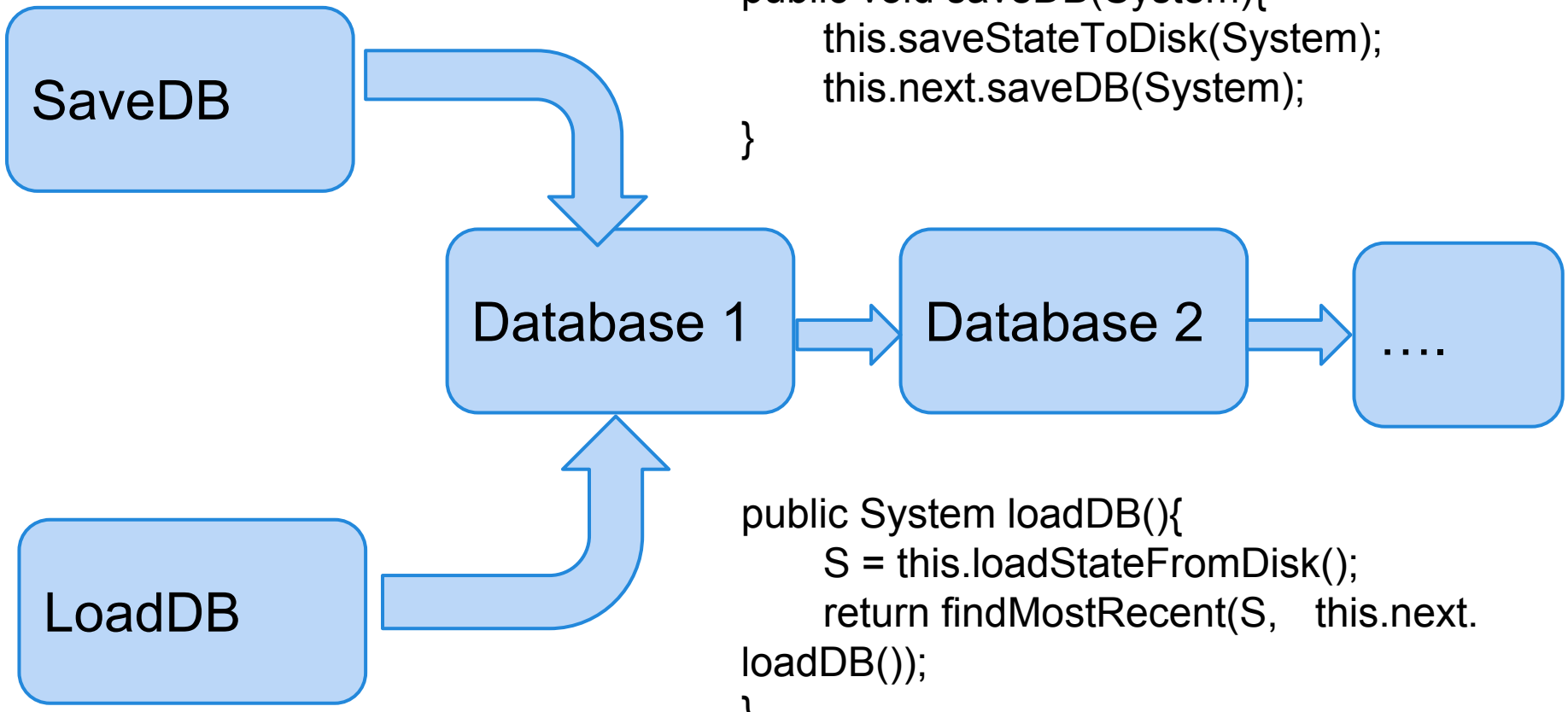
- Manage data backup
- Chain pattern

Save and load are both operated in chain

Set a new profile at first. When a new data location is added, update the profile.

Everytime in recovery mode, load the profile first, then search through the database.

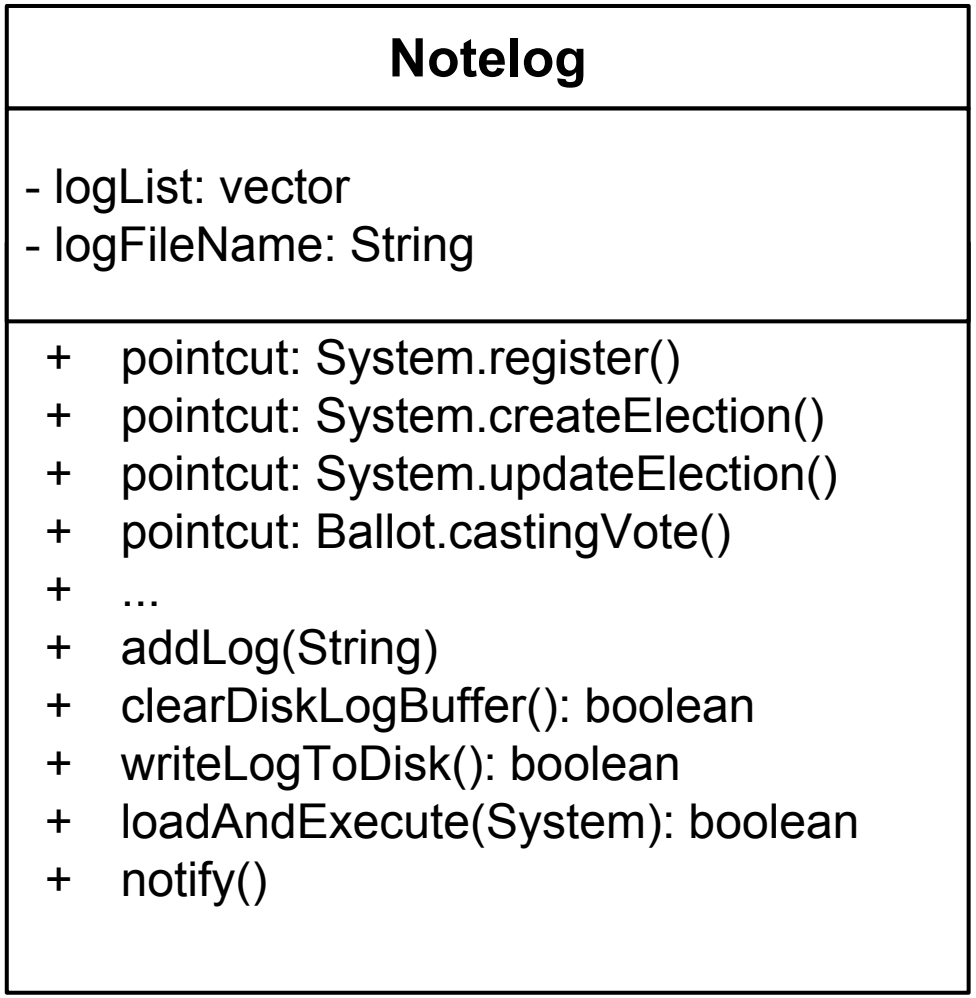
DataBackuper Chain

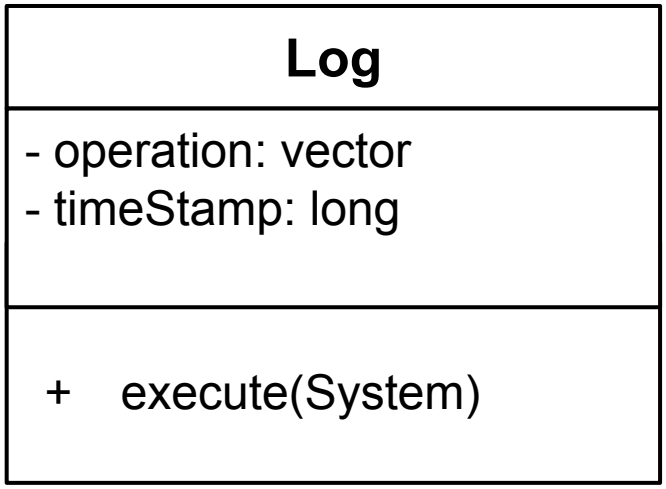



```
public void saveDB(System){
        this.saveStateToDisk(System);
        this.next.saveDB(System);
}
```

```
public System loadDB(){
        S = this.loadStateFromDisk();
        return findMostRecent(S,    this.next.
loadDB());
}
```

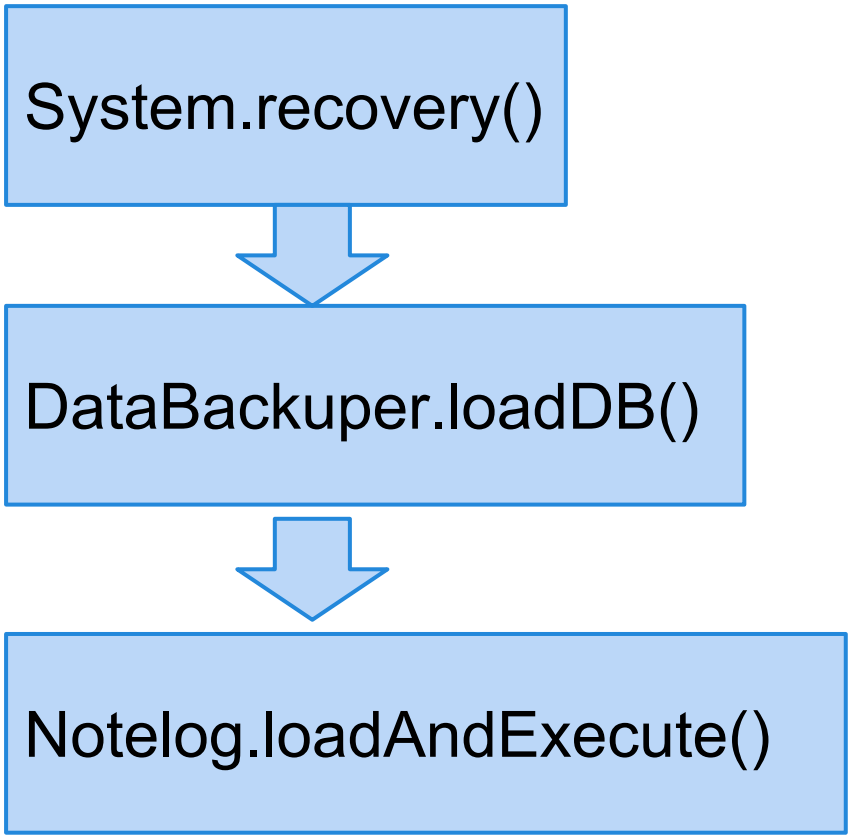
Notification & Log





- Manage logs and notifications about the voting system
- Aspect Oriented Programming

Recovery Process



If any problem happened during the election process and we want to recover the data, we go to the recovery mode. Load DB profile settings.

Load from backup disks in a chain, and ask for the backup with most recent time stamp

Load from logs and execute the logs to recover to the most recent state

Thanks

Questions & Comments