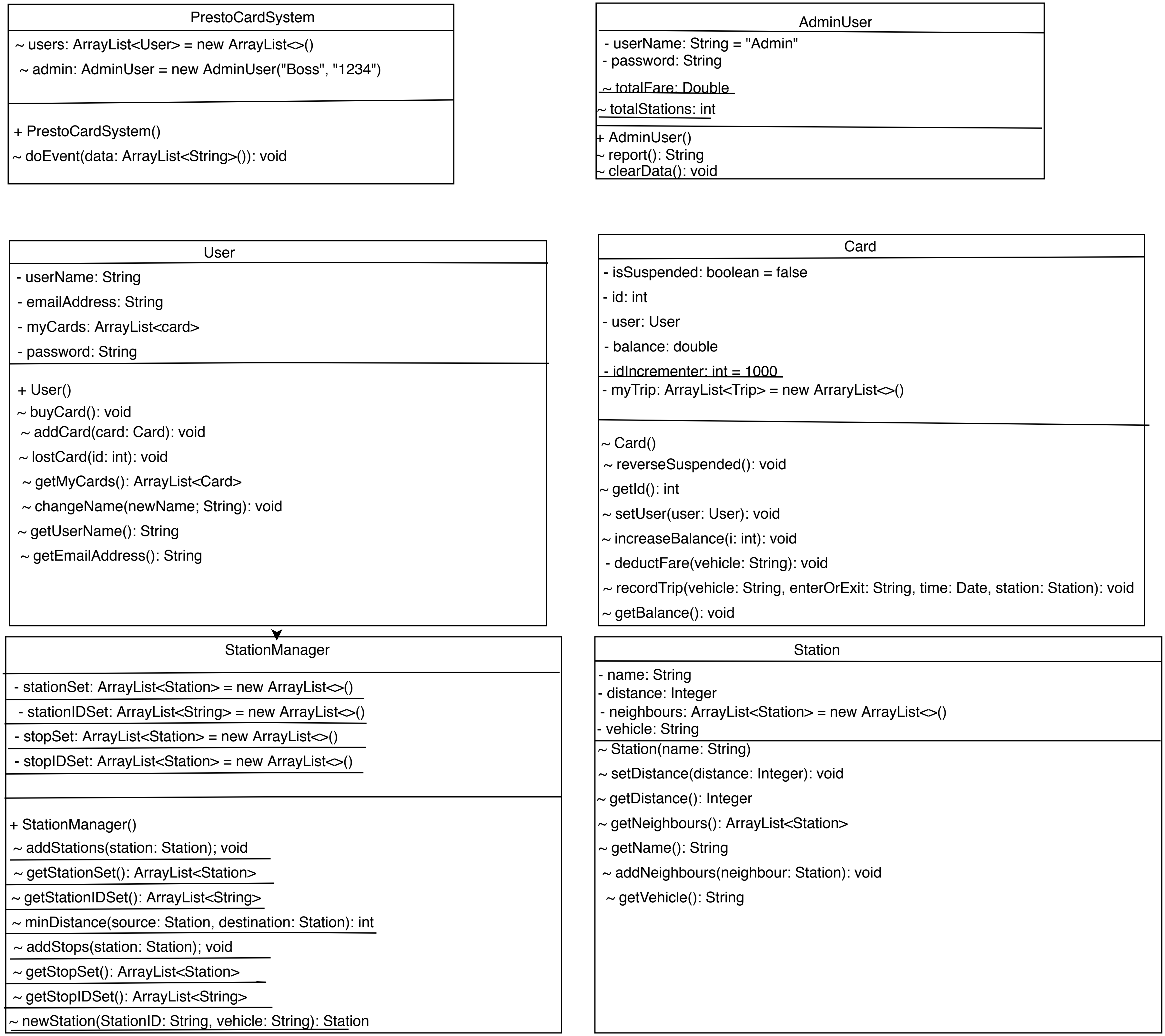


PrestoCardSystem

~ users: ArrayList<User> = new ArrayList<>()

~ admin: AdminUser = new AdminUser("Boss", "1234")

+ PrestoCardSystem()

~ doEvent(data: ArrayList<String>()): void

AdminUser

- userName: String = "Admin"

- password: String

~ totalFare: Double

~ totalStations: int

+ AdminUser()

~ report(): String

~ clearData(): void

User

- userName: String

- emailAddress: String

- myCards: ArrayList<card>

- password: String

+ User()

~ buyCard(): void

~ addCard(card: Card): void

~ lostCard(id: int): void

~ getMyCards(): ArrayList<Card>

~ changeName(newName; String): void

~ getUserName(): String

~ getEmailAddress(): String

Card

- isSuspended: boolean = false

- id: int

- user: User

- balance: double

- idIncrementer: int = 1000

- myTrip: ArrayList<Trip> = new ArraryList<>()

~ Card()

~ reverseSuspended(): void

~ getId(): int

~ setUser(user: User): void

~ increaseBalance(i: int): void

- deductFare(vehicle: String): void

~ recordTrip(vehicle: String, enterOrExit: String, time: Date, station: Station): void

~ getBalance(): void

StationManager

- stationSet: ArrayList<Station> = new ArrayList<>()

- stationIDSet: ArrayList<String> = new ArrayList<>()

- stopSet: ArrayList<Station> = new ArrayList<>()

- stopIDSet: ArrayList<Station> = new ArrayList<>()

+ StationManager()

~ addStations(station: Station); void

~ getStationSet(): ArrayList<Station>

~ getStationIDSet(): ArrayList<String>

~ minDistance(source: Station, destination: Station): int

~ addStops(station: Station); void

~ getStopSet(): ArrayList<Station>

~ getStopIDSet(): ArrayList<String>

~ newStation(StationID: String, vehicle: String): Station

Station

- name: String

- distance: Integer

- neighbours: ArrayList<Station> = new ArrayList<>()

- vehicle: String

~ Station(name: String)

~ setDistance(distance: Integer): void

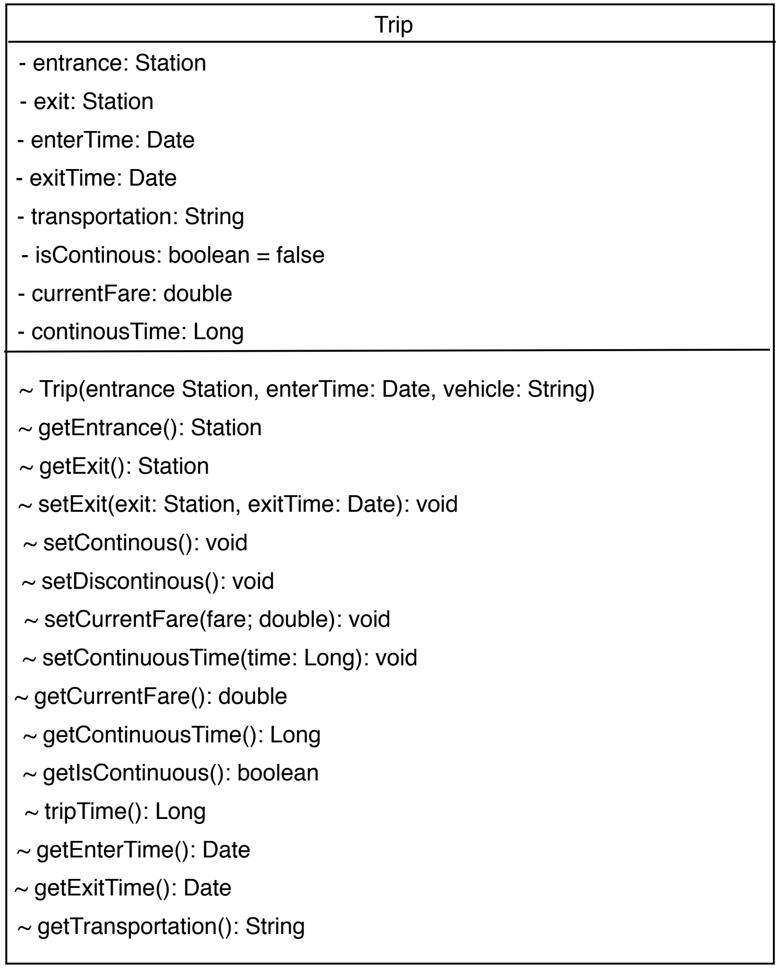
~ getDistance(): Integer

~ getNeighbours(): ArrayList<Station>

~ getName(): String

~ addNeighbours(neighbour: Station): void

~ getVehicle(): String



Trip

- entrance: Station
- exit: Station
- enterTime: Date
- exitTime: Date
- transportation: String
- isContinous: boolean = false
- currentFare: double
- continousTime: Long

~ Trip(entrance Station, enterTime: Date, vehicle: String)
~ getEntrance(): Station
~ getExit(): Station
~ setExit(exit: Station, exitTime: Date): void
~ setContinous(): void
~ setDiscontinous(): void
~ setCurrentFare(fare; double): void
~ setContinuousTime(time: Long): void
~ getCurrentFare(): double
~ getContinuousTime(): Long
~ getIsContinuous(): boolean
~ tripTime(): Long
~ getEnterTime(): Date
~ getExitTime(): Date
~ getTransportation(): String