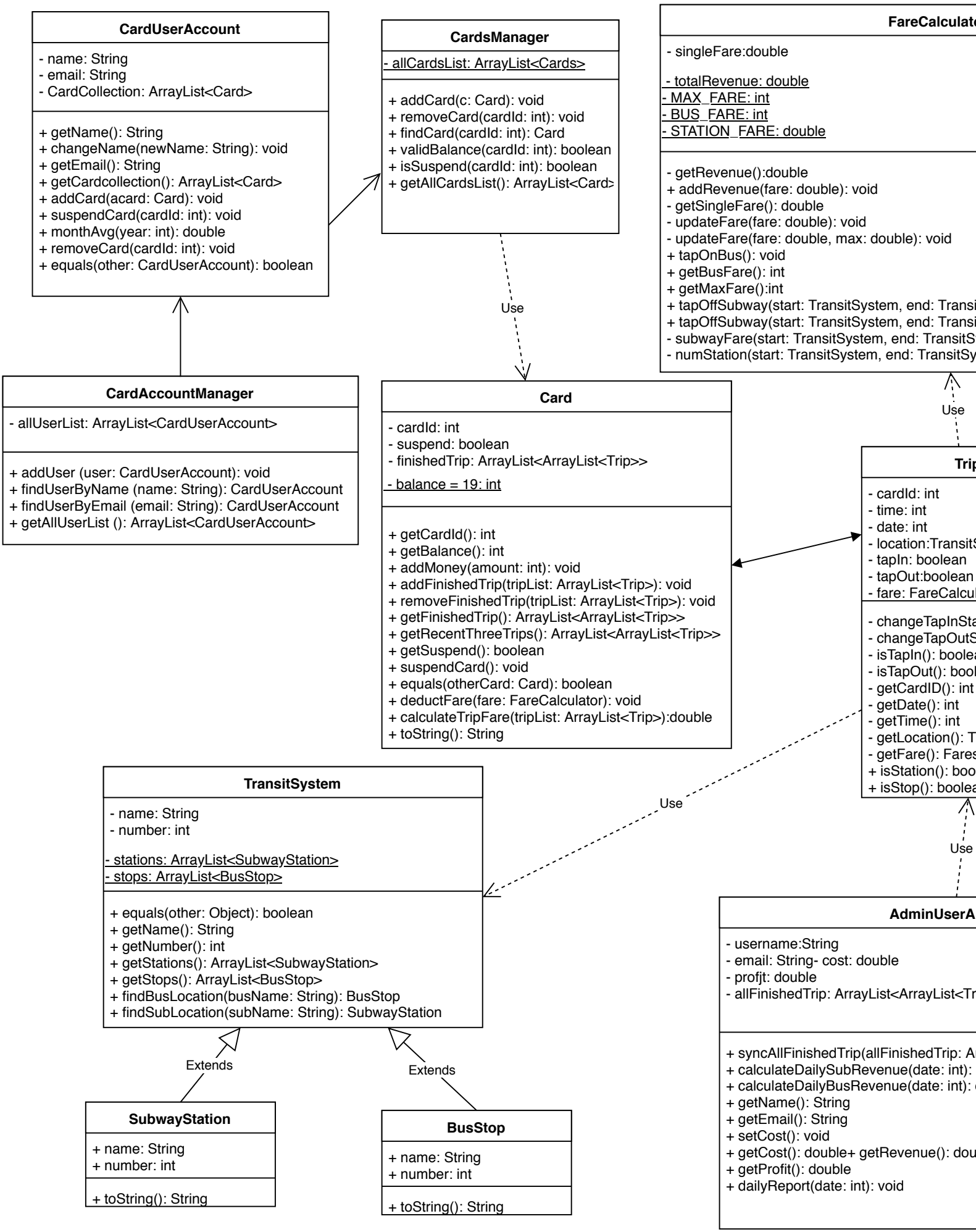
CardUserAccount
- name: String
- email: String
- CardCollection: ArrayList<Card>
+ getName(): String
+ changeName(newName: String): void
+ getEmail(): String
+ getCardcollection(): ArrayList<Card>
+ addCard(acard: Card): void
+ suspendCard(cardId: int): void
+ monthAvg(year: int): double
+ removeCard(cardId: int): void
+ equals(other: CardUserAccount): boolean
CardsManager
- allCardsList: ArrayList<Cards>
+ addCard(c: Card): void
+ removeCard(cardId: int): void
+ findCard(cardId: int): Card
+ validBalance(cardId: int): boolean
+ isSuspend(cardId: int): boolean
+ getAllCardsList(): ArrayList<Card>
FareCalculat
- singleFare:double
- totalRevenue: double
- MAX_FARE: int
- BUS_FARE: int
- STATION_FARE: double
- getRevenue():double
+ addRevenue(fare: double): void
- getSingleFare(): double
- updateFare(fare: double): void
- updateFare(fare: double, max: double): void
+ tapOnBus(): void
+ getBusFare(): int
+ getMaxFare():int
+ tapOffSubway(start: TransitSystem, end: Transi
+ tapOffSubway(start: TransitSystem, end: Transi
- subwayFare(start: TransitSystem, end: TransitS
- numStation(start: TransitSystem, end: TransitSy
Use
CardAccountManager
- allUserList: ArrayList<CardUserAccount>
+ addUser (user: CardUserAccount): void
+ findUserByName (name: String): CardUserAccount
+ findUserByEmail (email: String): CardUserAccount
+ getAllUserList (): ArrayList<CardUserAccount>
Card
- cardId: int
- suspend: boolean
- finishedTrip: ArrayList<ArrayList<Trip>>
- balance = 19: int
+ getCardId(): int
+ getBalance(): int
+ addMoney(amount: int): void
+ addFinishedTrip(tripList: ArrayList<Trip>): void
+ removeFinishedTrip(tripList: ArrayList<Trip>): void
+ getFinishedTrip(): ArrayList<ArrayList<Trip>>
+ getRecentThreeTrips(): ArrayList<ArrayList<Trip>>
+ getSuspend(): boolean
+ suspendCard(): void
+ equals(otherCard: Card): boolean
+ deductFare(fare: FareCalculator): void
+ calculateTripFare(tripList: ArrayList<Trip>):double
+ toString(): String
Use
Trip
- cardId: int
- time: int
- date: int
- location:TransitS
- tapIn: boolean
- tapOut:boolean
- fare: FareCalcul
- changeTapInSta
- changeTapOutS
- isTapIn(): boolea
- isTapOut(): bool
- getCardID(): int
- getDate(): int
- getTime(): int
- getLocation(): T
- getFare(): Fares
+ isStation(): boo
+ isStop(): boolea
Use
TransitSystem
- name: String
- number: int
- stations: ArrayList<SubwayStation>
- stops: ArrayList<BusStop>
+ equals(other: Object): boolean
+ getName(): String
+ getNumber(): int
+ getStations(): ArrayList<SubwayStation>
+ getStops(): ArrayList<BusStop>
+ findBusLocation(busName: String): BusStop
+ findSubLocation(subName: String): SubwayStation
Use
AdminUserA
- username:String
- email: String- cost: double
- profjt: double
- allFinishedTrip: ArrayList<ArrayList<Tr
+ syncAllFinishedTrip(allFinishedTrip: A
+ calculateDailySubRevenue(date: int):
+ calculateDailyBusRevenue(date: int):
+ getName(): String
+ getEmail(): String
+ setCost(): void
+ getCost(): double+ getRevenue(): dou
+ getProfit(): double
+ dailyReport(date: int): void
Extends
Extends
SubwayStation
+ name: String
+ number: int
+ toString(): String
BusStop
+ name: String
+ number: int
+ toString(): String

or

tSystem):void
tSystem, maxFare: double): void
ystem,):double
ystem): int

TripManager

- finishedTrips: ArrayList<ArrayList<Trip>>
- unfinishedTrips: ArrayList<ArrayList<Trip>>
- cm: CardsManager

- syncCardManager(MainCardManager: CardsManager): void
- getFinishedTrips(): ArrayList<ArrayList<Trip>>

+ toString(tripList: ArrayList<Trip>): String
- addBusTrip(trip: Trip): void
- addSubwayTrip(trip: Trip): void
- validTransfer(oldTrip: Trip, newTrip: Trip): boolean
- doubleTapInAfterSubway(lastTrip: Trip, card1:Card, previousTripList: ArrayList<Trip>, newTrip: Trip): void
- doubleTapInAfterBus(card1: Card, previousTripList: ArrayList<Trip>, newTrip: Trip)
- transferringTrip(card1: Card, tripList: ArrayList<Trip>, newTrip: Trip):void
- tapOut(card1: Card, tripList: ArrayList<Trip>, newTrip:Trip): void
- notMatchingTapInTapOut(tripCard: Card, tripList2: ArrayList<Trip>, trip:Trip):void
- addTrip(trip: Trip): void

p

System

lator

atus(): void
Status(): void
an
ean

ransitSystem
s
lean
an

ccount

ip>>

rrayList<ArrayList<Trip>>): void
double
double

ble