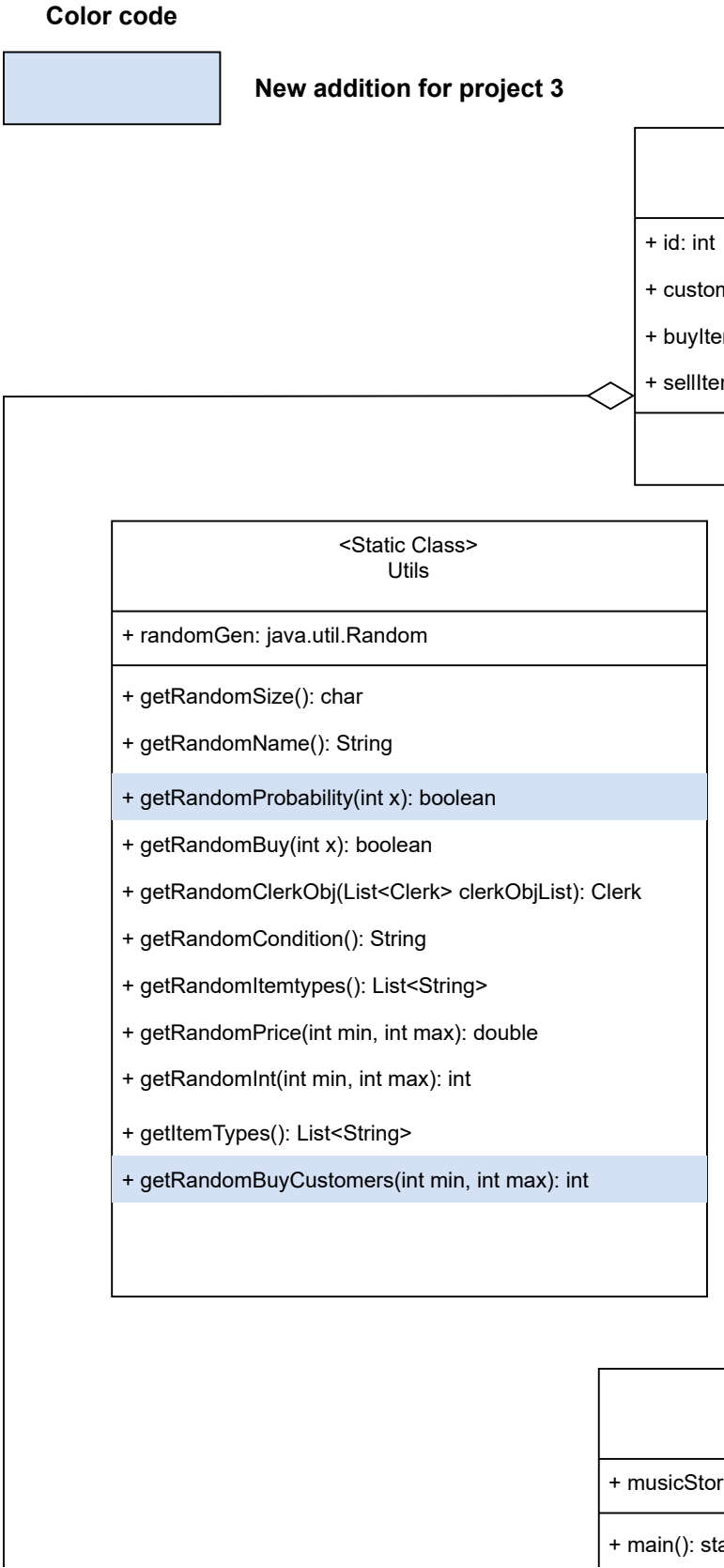
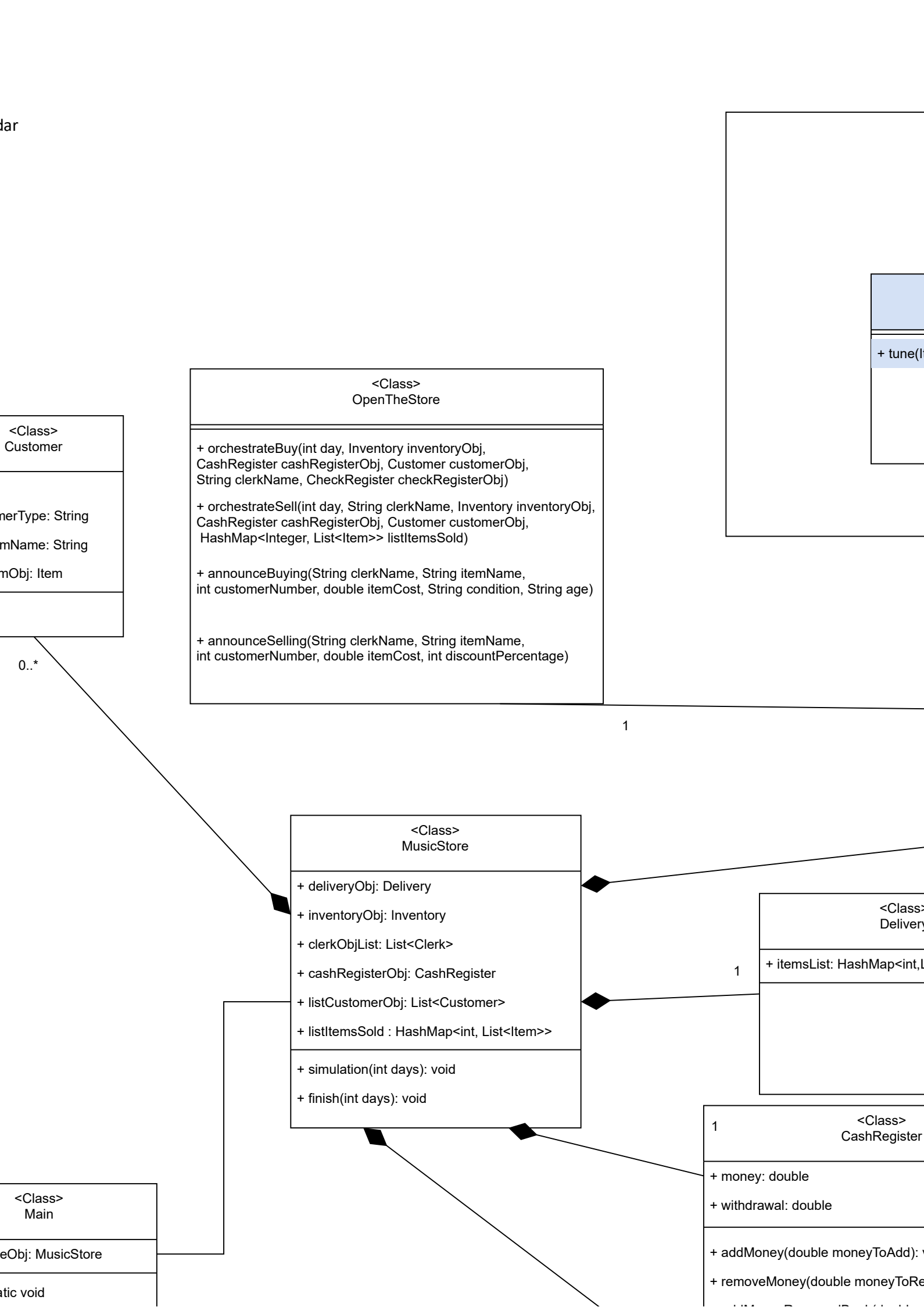
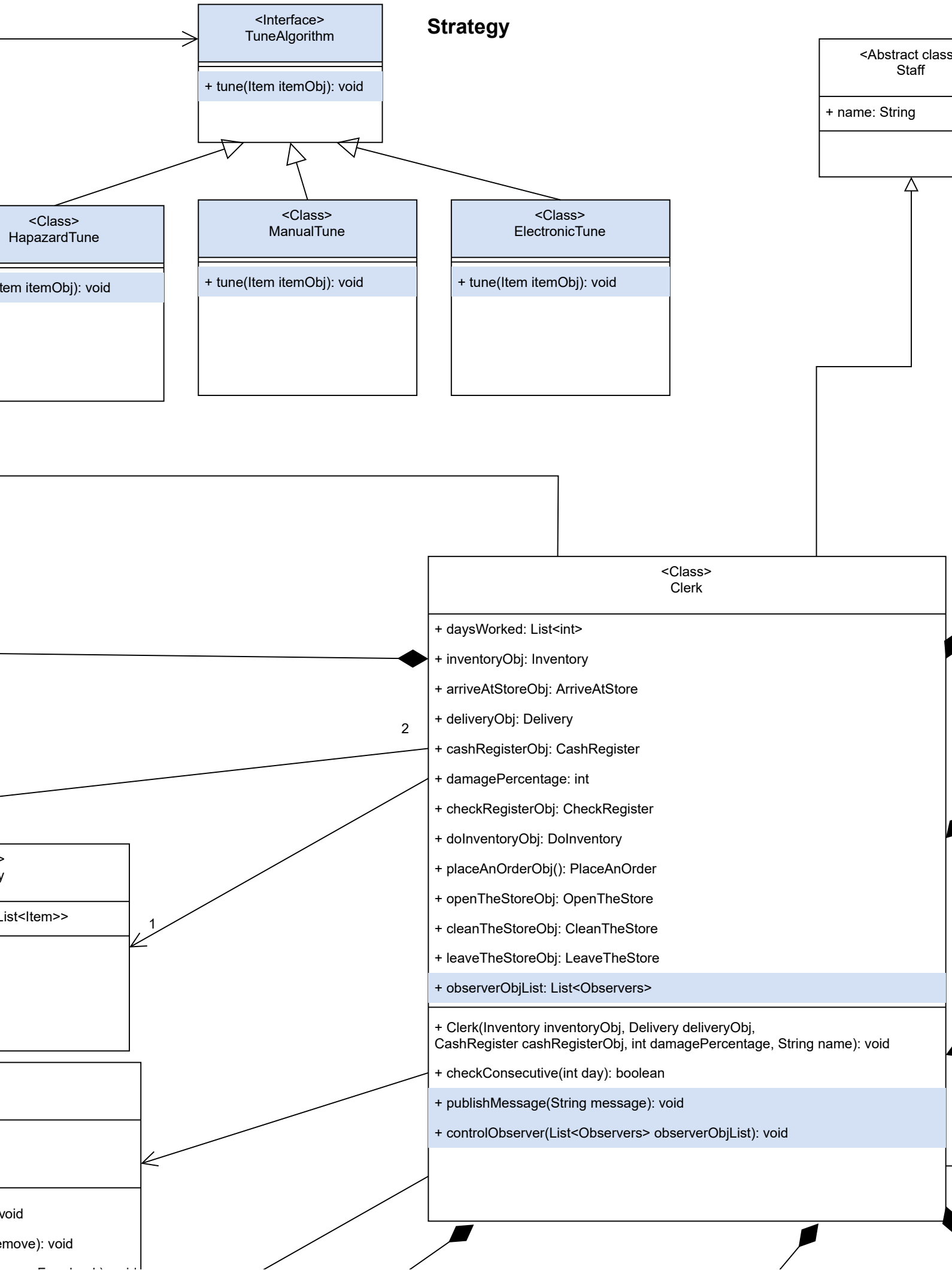
FNMS

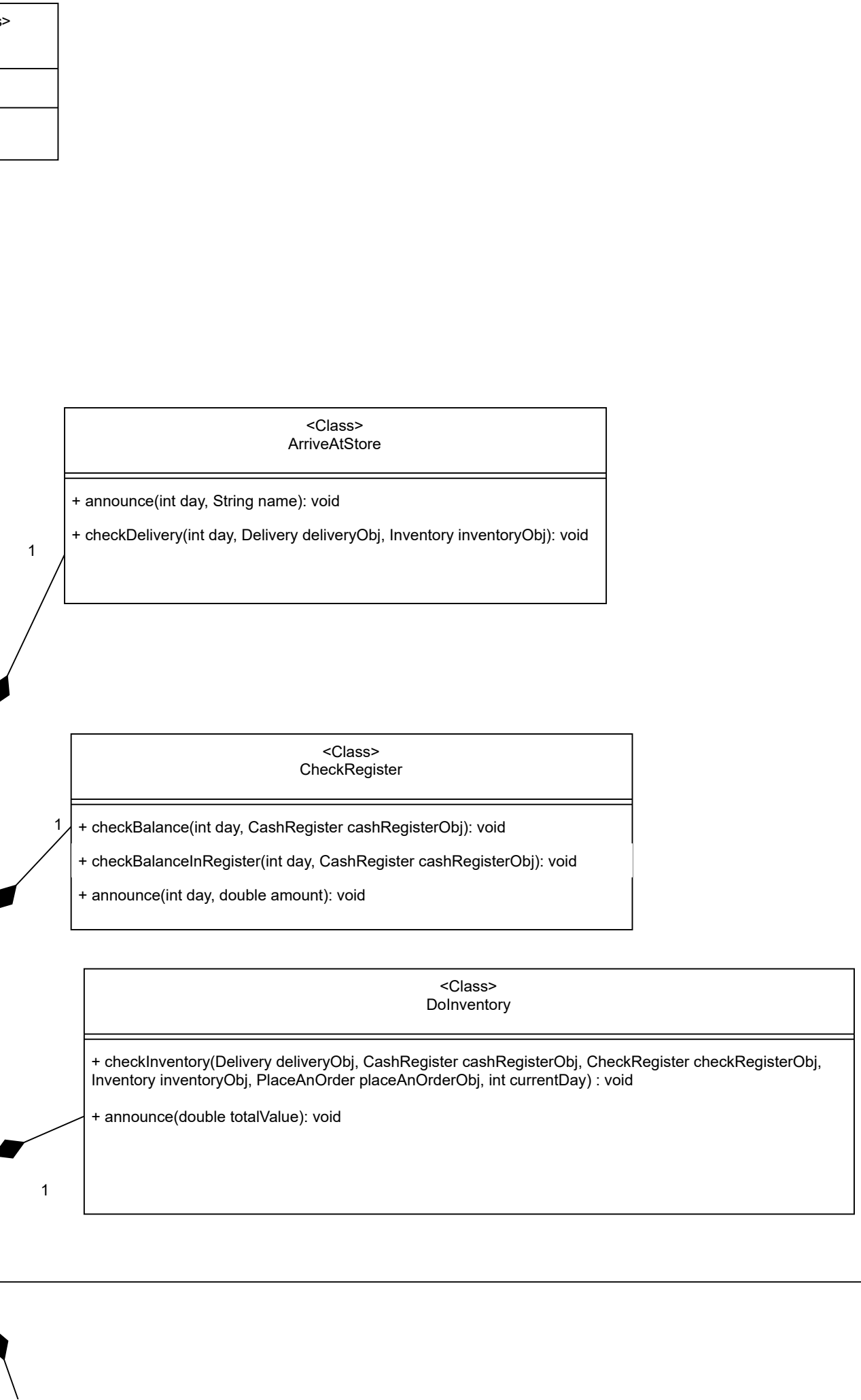
Team member: Srinivas Akhil Mallela, Likhitha Katakam, Vishal Prabhachand

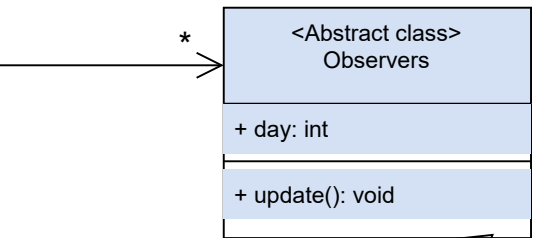
Project 3 Q2 - UML



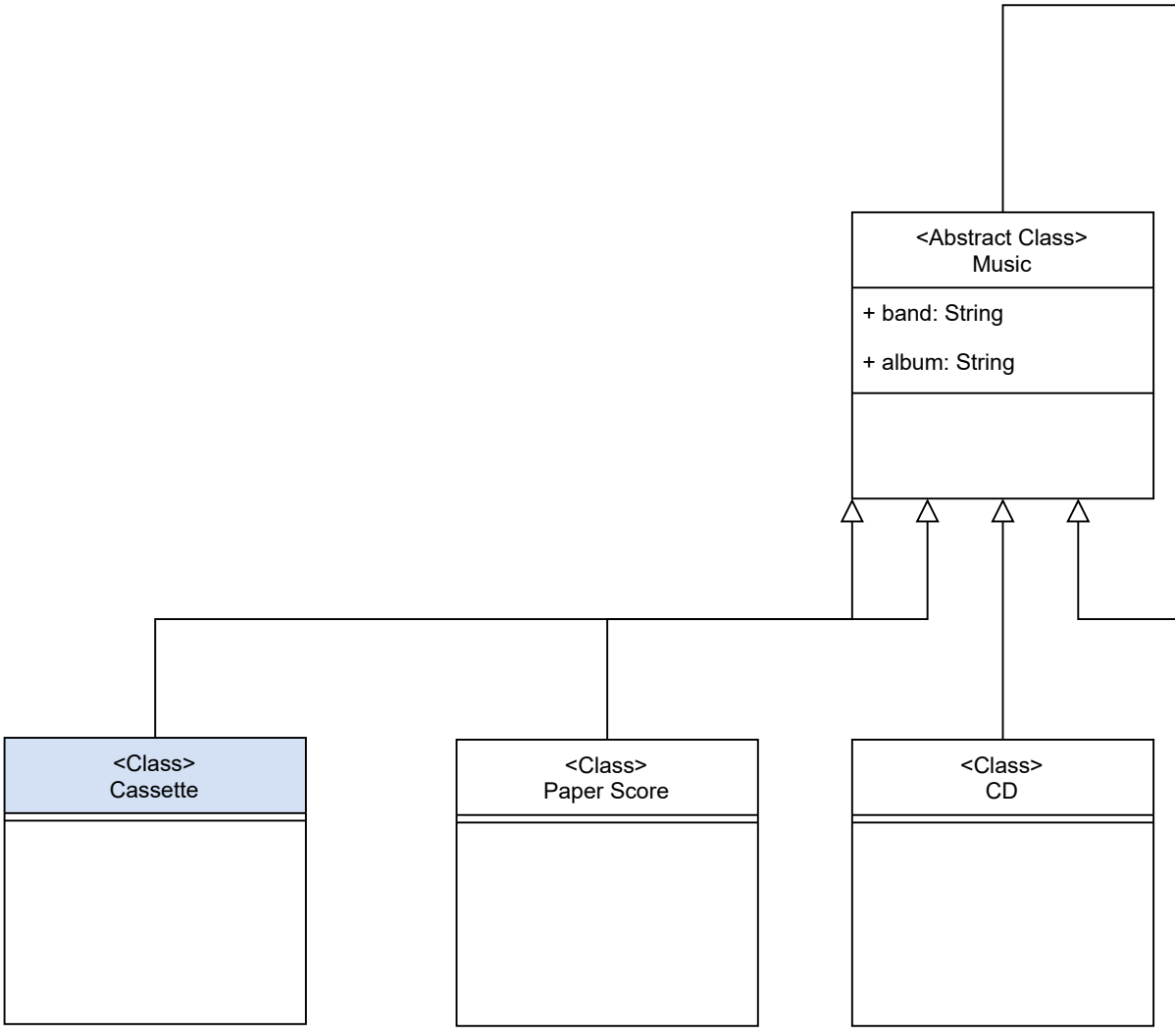

dar
<Class>
Customer
merType: String
mName: String
mObj: Item
0..*
<Class>
OpenTheStore
+ orchestrateBuy(int day, Inventory inventoryObj,
CashRegister cashRegisterObj, Customer customerObj,
String clerkName, CheckRegister checkRegisterObj)
+ orchestrateSell(int day, String clerkName, Inventory inventoryObj,
CashRegister cashRegisterObj, Customer customerObj,
HashMap<Integer, List<Item>> listItemsSold)
+ announceBuying(String clerkName, String itemName,
int customerNumber, double itemCost, String condition, String age)
+ announceSelling(String clerkName, String itemName,
int customerNumber, double itemCost, int discountPercentage)
1
+ tune(I
<Class>
MusicStore
+ deliveryObj: Delivery
+ inventoryObj: Inventory
+ clerkObjList: List<Clerk>
+ cashRegisterObj: CashRegister
+ listCustomerObj: List<Customer>
+ listItemsSold : HashMap<int, List<Item>>
+ simulation(int days): void
+ finish(int days): void
1
<Class>
Delivery
+ itemsList: HashMap<int,L
1
<Class>
CashRegister
+ money: double
+ withdrawal: double
+ addMoney(double moneyToAdd):
+ removeMoney(double moneyToRe
<Class>
Main
eObj: MusicStore
atic void

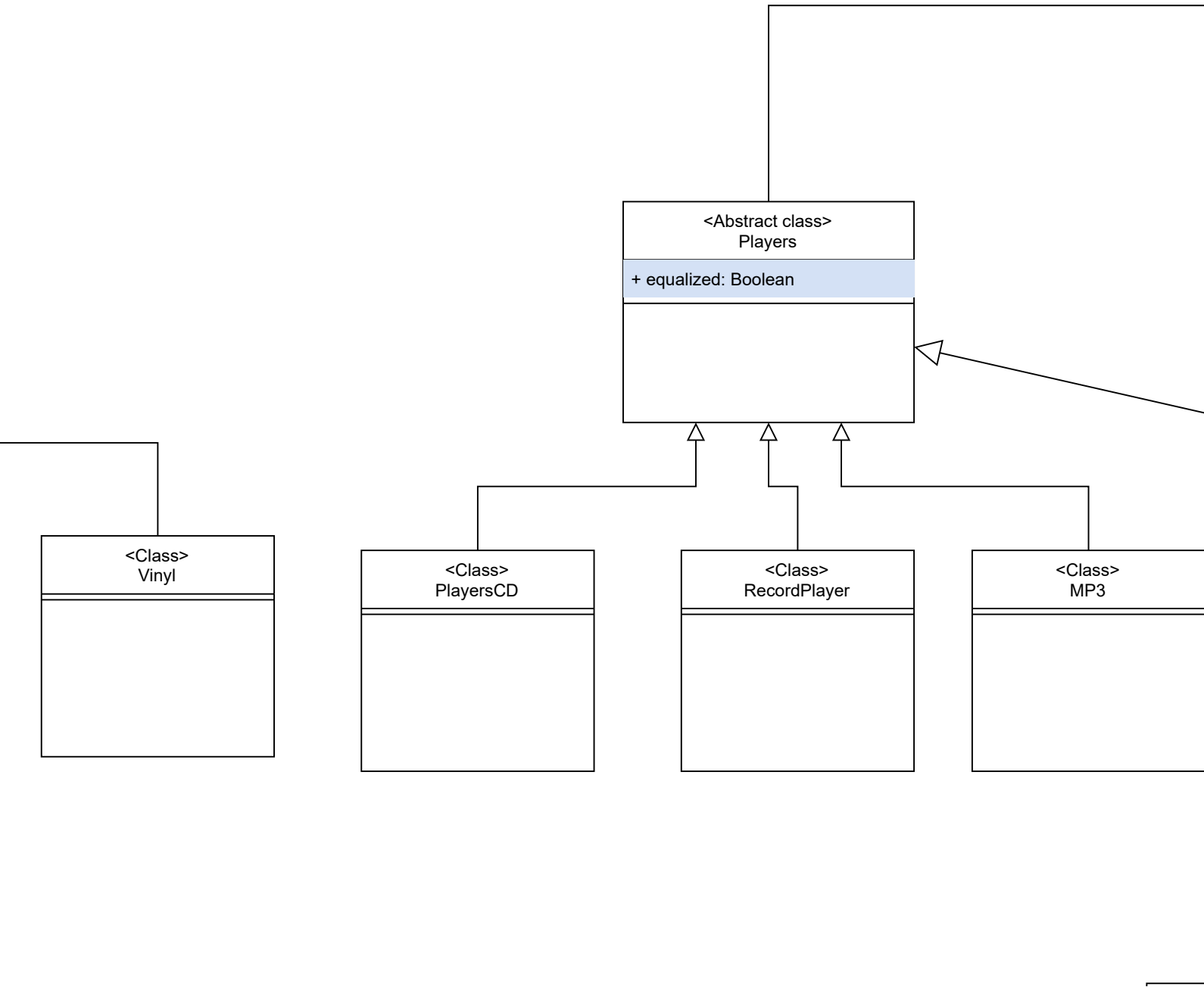

Strategy
<Interface>
TuneAlgorithm
+ tune(Item itemObj): void
<Class>
HapazardTune
tem itemObj): void
<Class>
ManualTune
+ tune(Item itemObj): void
<Class>
ElectronicTune
+ tune(Item itemObj): void
<Abstract class
Staff
+ name: String
<Class>
Clerk
+ daysWorked: List<int>
+ inventoryObj: Inventory
+ arriveAtStoreObj: ArriveAtStore
+ deliveryObj: Delivery
+ cashRegisterObj: CashRegister
+ damagePercentage: int
+ checkRegisterObj: CheckRegister
+ doInventoryObj: DoInventory
+ placeAnOrderObj(): PlaceAnOrder
+ openTheStoreObj: OpenTheStore
+ cleanTheStoreObj: CleanTheStore
+ leaveTheStoreObj: LeaveTheStore
+ observerObjList: List<Observers>
+ Clerk(Inventory inventoryObj, Delivery deliveryObj,
CashRegister cashRegisterObj, int damagePercentage, String name): void
+ checkConsecutive(int day): boolean
+ publishMessage(String message): void
+ controlObserver(List<Observers> observerObjList): void
2
1
List<Item>>
void
emove): void

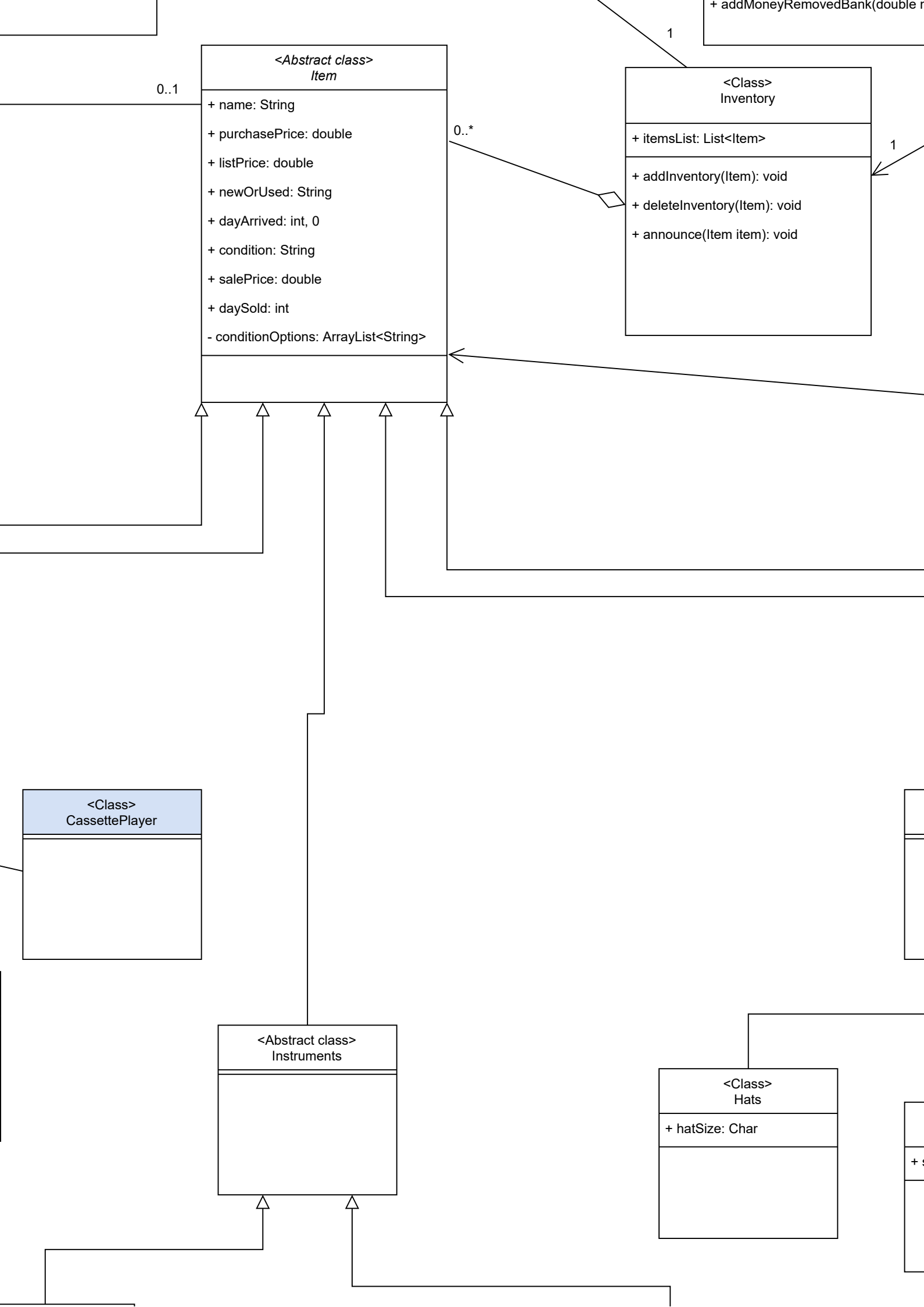

<Class>
ArriveAtStore
+ announce(int day, String name): void
+ checkDelivery(int day, Delivery deliveryObj, Inventory inventoryObj): void
1
<Class>
CheckRegister
1
+ checkBalance(int day, CashRegister cashRegisterObj): void
+ checkBalanceInRegister(int day, CashRegister cashRegisterObj): void
+ announce(int day, double amount): void
<Class>
DoInventory
+ checkInventory(Delivery deliveryObj, CashRegister cashRegisterObj, CheckRegister checkRegisterObj, Inventory inventoryObj, PlaceAnOrder placeAnOrderObj, int currentDay) : void
+ announce(double totalValue): void
1



Observer


<Abstract Class>
Music
+ band: String
+ album: String
<Class>
Cassette
<Class>
Paper Score
<Class>
CD


<Abstract class>
Players
+ equalized: Boolean
<Class>
Vinyl
<Class>
PlayersCD
<Class>
RecordPlayer
<Class>
MP3


+ addMoneyRemovedBank(double
1
<Abstract class>
Item
0..1
+ name: String
+ purchasePrice: double
+ listPrice: double
+ newOrUsed: String
+ dayArrived: int, 0
+ condition: String
+ salePrice: double
+ daySold: int
- conditionOptions: ArrayList<String>
0..*
<Class>
Inventory
+ itemsList: List<Item>
+ addInventory(Item): void
+ deleteInventory(Item): void
+ announce(Item item): void
1
<Class>
CassettePlayer
<Abstract class>
Instruments
<Class>
Hats
+ hatSize: Char

moneyFrombank): void

1

<Class>
LeaveTheStore

+ announce(String name, int day)

1

<Class>
CleanTheStore

+ announce(int damagePercentage, String itemType, double listPrice, String condition):void

+ orchestrate(int damagePercentage, Inventory inventoryObj):void

<Abstract class>
AccessoriesDecorator

Deco

<Abstract class>
Accessories

<Abstract class>
Clothing

<Class>
Practise Amps

+ wattage: int

<Class>
Cabels

+ length: int

+ ty

<Class>
Bandanas

<Class>
Shirts

shirtSize: Char

1

<Class>
PlaceAnOrder

+ orderItem(String itemStr, Delivery deliverObj, CashRegister cashRegisterObj, CheckRegister checkRegObj,int currentDay):void

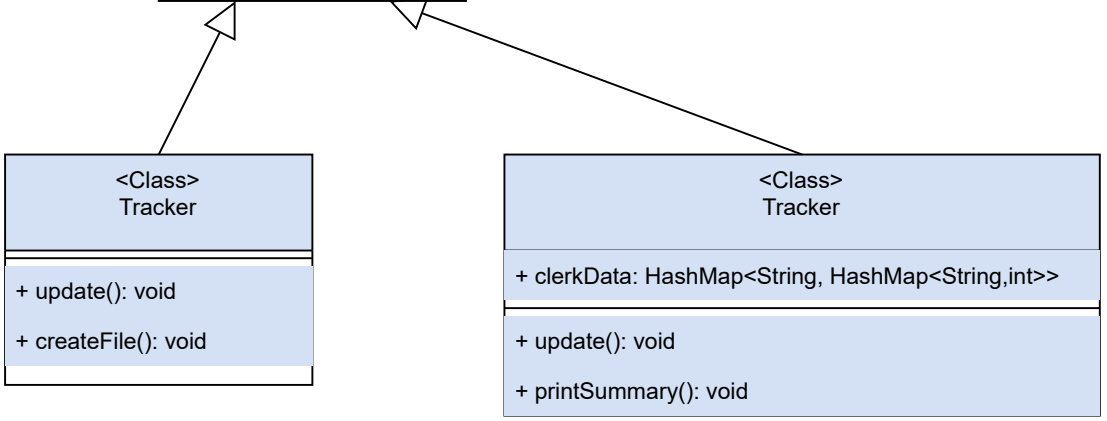
+ announce(List<Item> itemObjs, int expectedDay, int currentDay): void

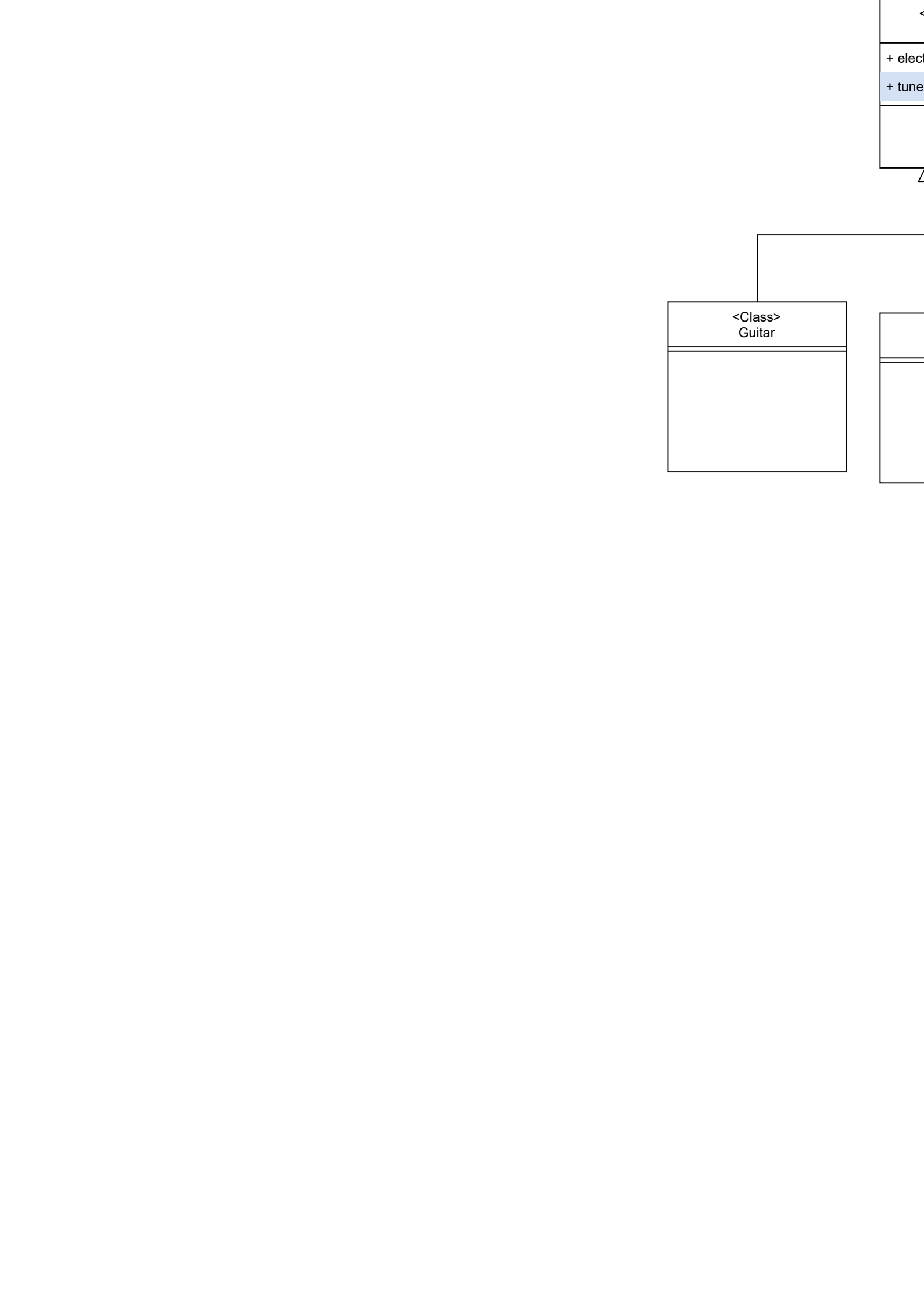
orator

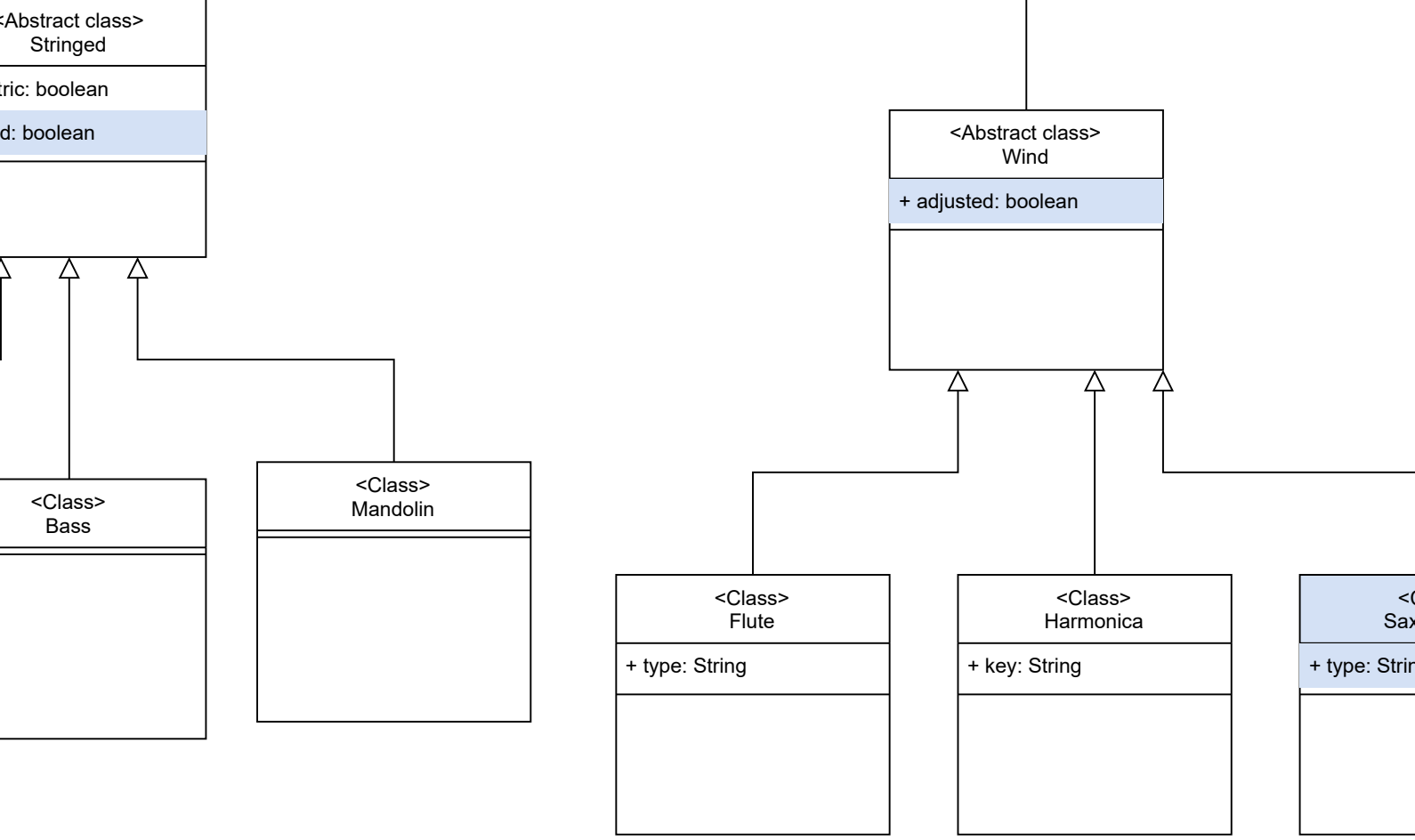
<Class>
GigBag

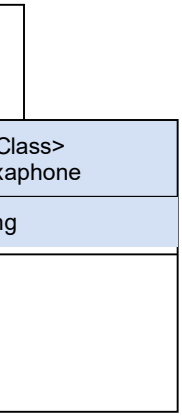
<Class>
Strings

pe: String


<Class>
Tracker
+ update(): void
+ createFile(): void
<Class>
Tracker
+ clerkData: HashMap<String, HashMap<String,int>>
+ update(): void
+ printSummary(): void


+ elect
+ tune
<Class>
Guitar


<Abstract class>
Stringed
tric: boolean
d: boolean
<Class>
Bass
<Class>
Mandolin
<Abstract class>
Wind
+ adjusted: boolean
<Class>
Flute
+ type: String
<Class>
Harmonica
+ key: String
+ type: Strin


Class>
xaphone
ng