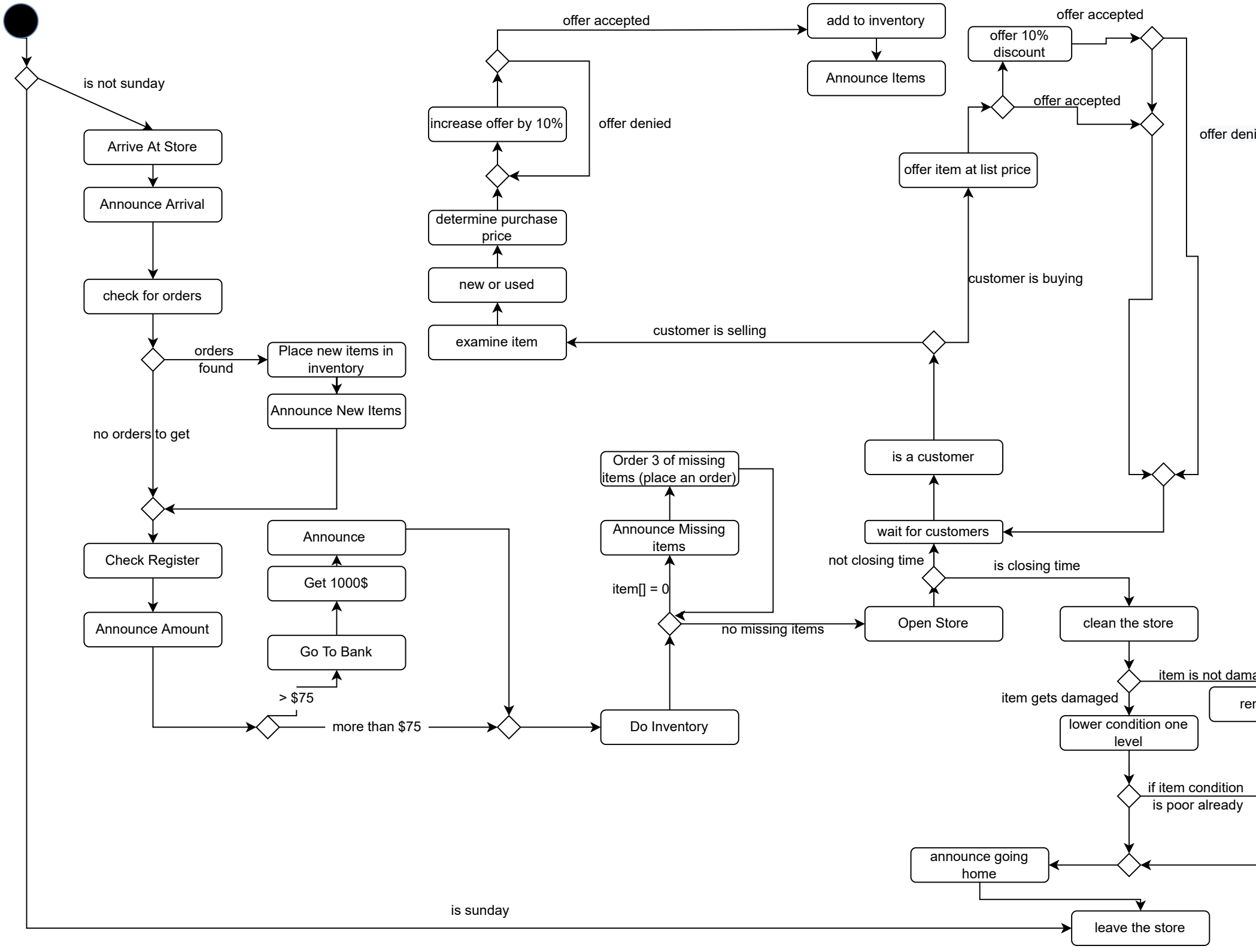
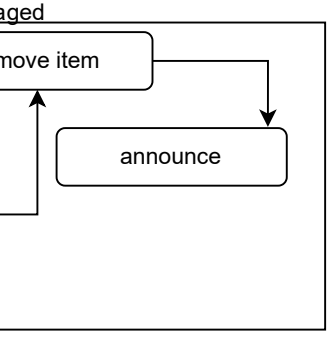
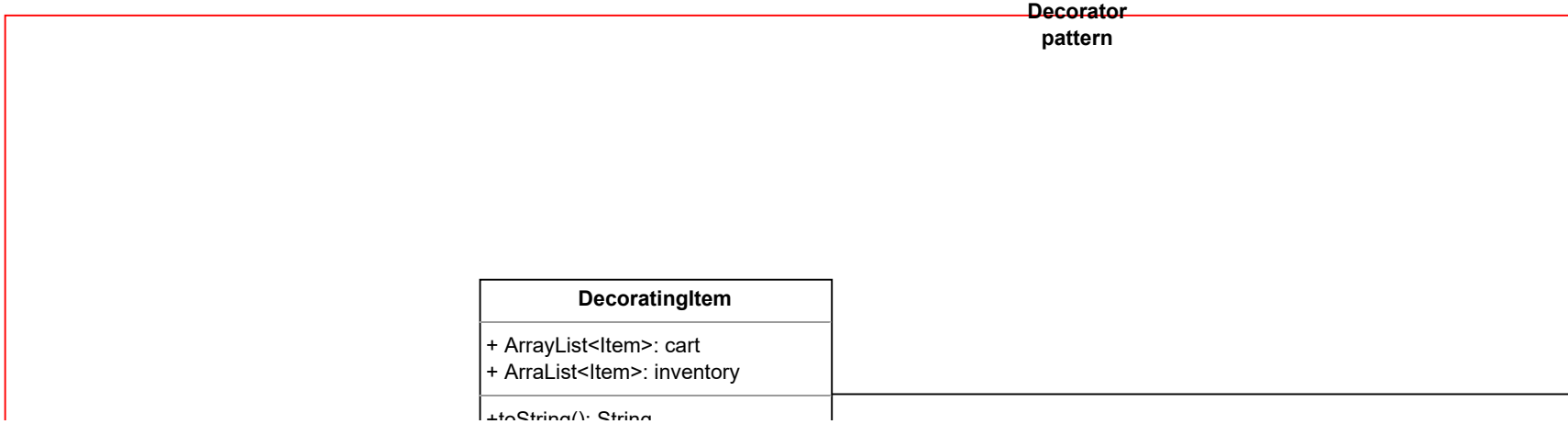
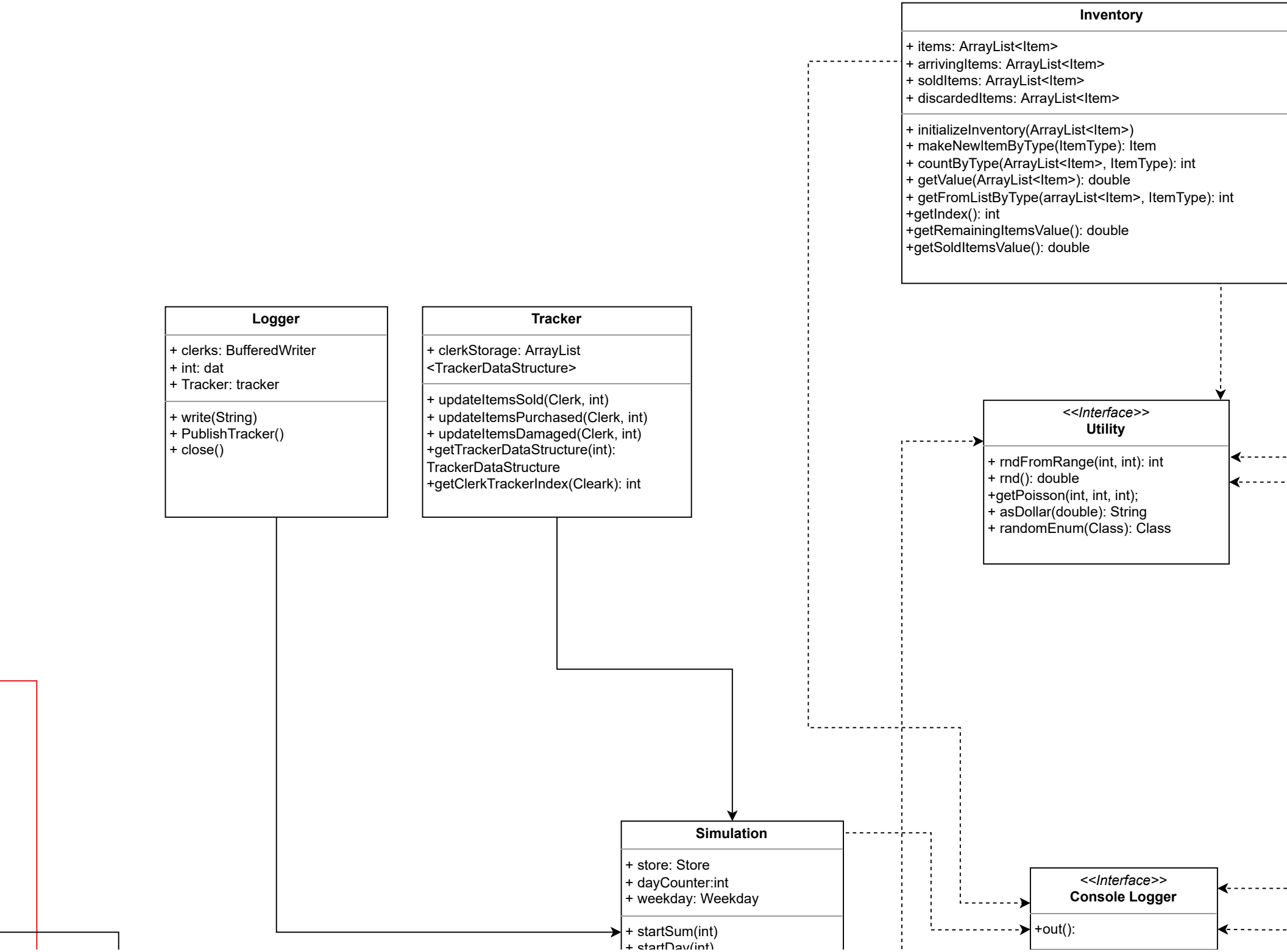

is not sunday
Arrive At Store
Announce Arrival
check for orders
orders found
Place new items in inventory
Announce New Items
no orders to get
Check Register
Announce Amount
Announce
Get 1000$
Go To Bank
> $75
more than $75
Order 3 of missing items (place an order)
Announce Missing items
item[] = 0
no missing items
Do Inventory
offer accepted
add to inventory
Announce Items
increase offer by 10%
offer denied
determine purchase price
new or used
examine item
customer is selling
offer accepted
offer 10% discount
offer accepted
offer item at list price
customer is buying
offer deni
is a customer
wait for customers
not closing time
is closing time
Open Store
clean the store
item is not dama
item gets damaged
lower condition one level
if item condition is poor already
announce going home
is sunday
leave the store

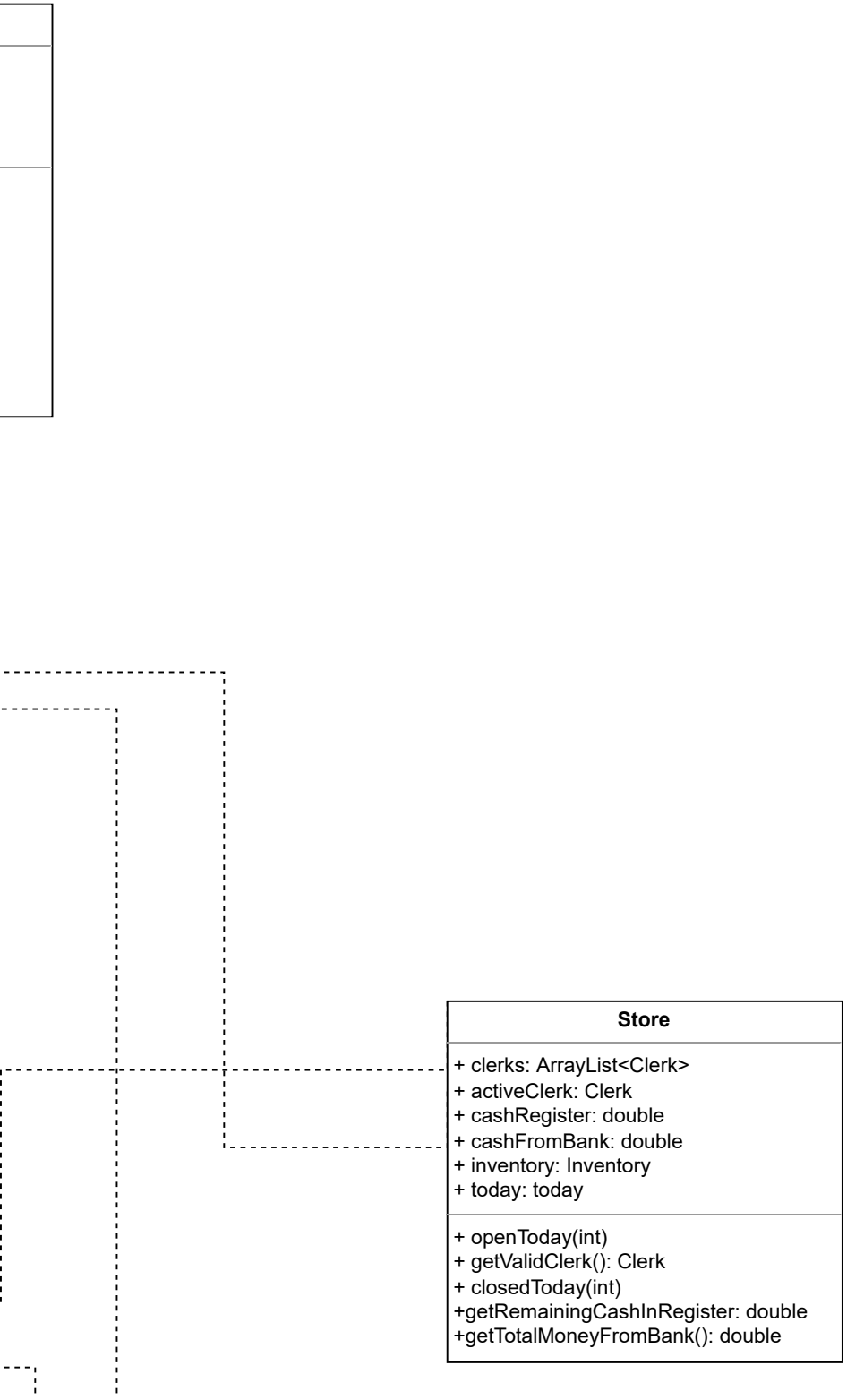
ed

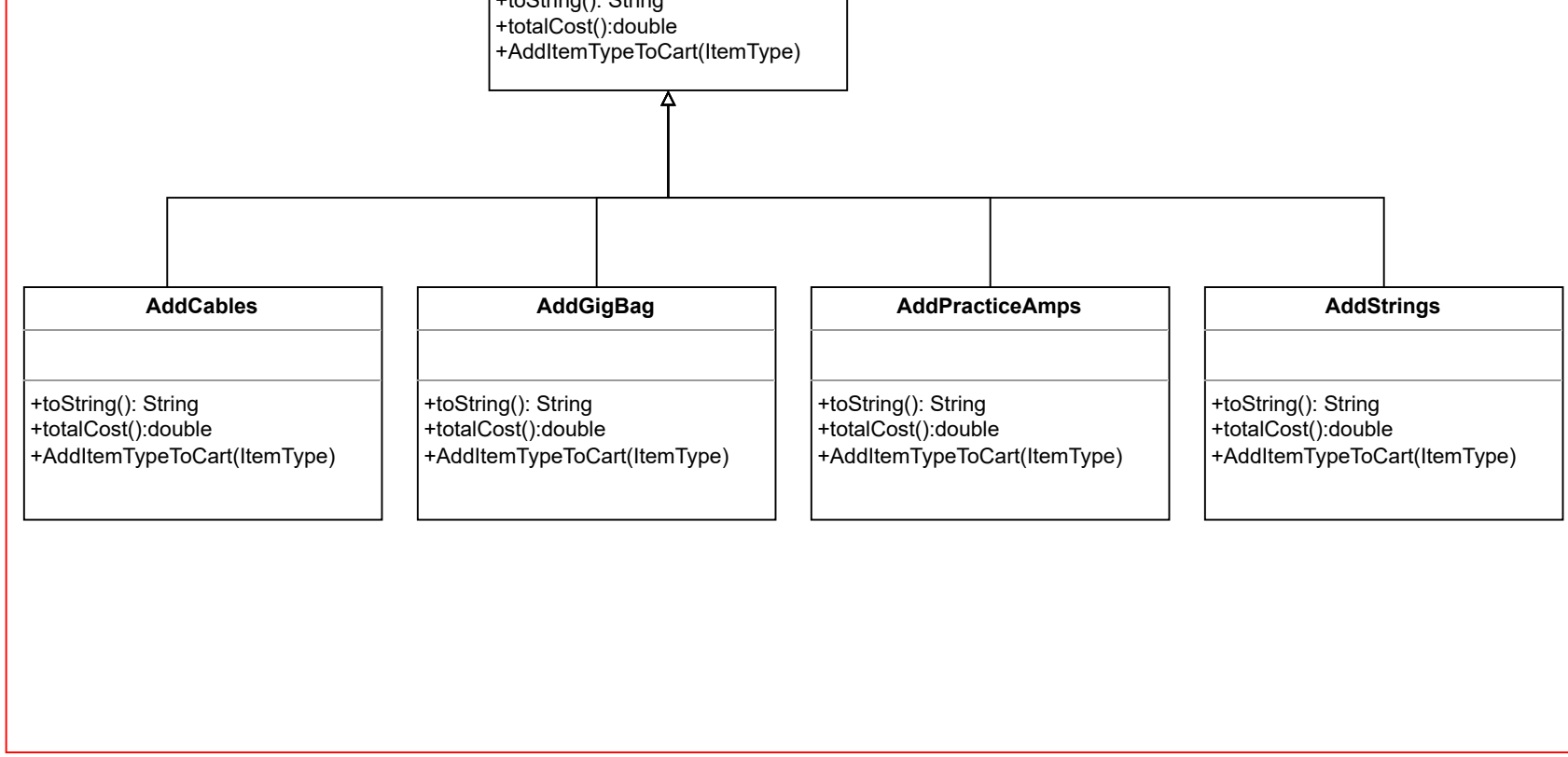
aged

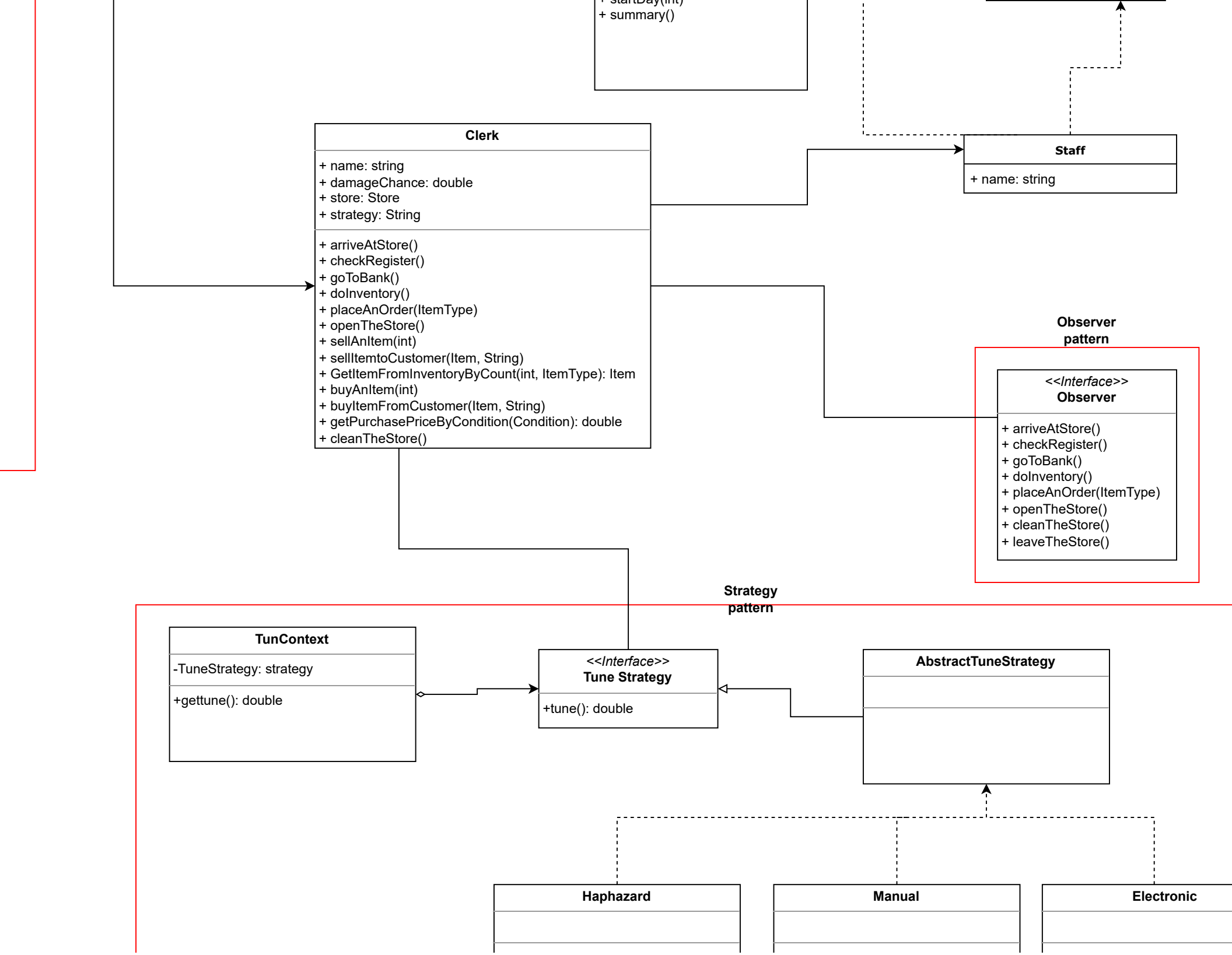


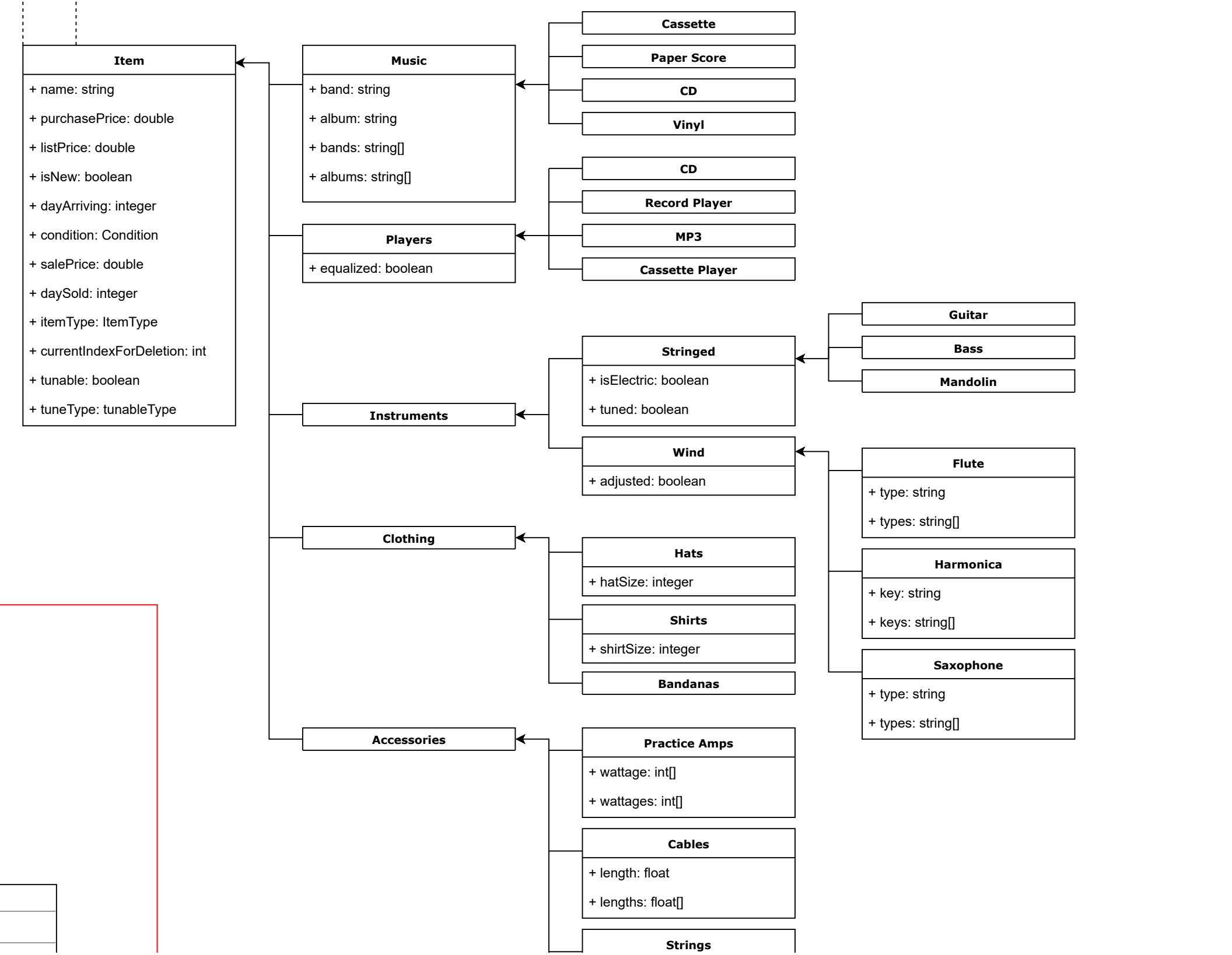

Decorator
pattern
DecoratingItem
+ ArrayList<Item>: cart
+ ArraList<Item>: inventory
+toString(): String

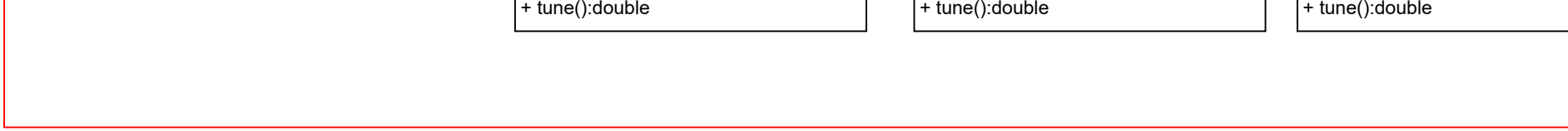

Inventory
+ items: ArrayList<Item>
+ arrivingItems: ArrayList<Item>
+ soldItems: ArrayList<Item>
+ discardedItems: ArrayList<Item>
+ initializeInventory(ArrayList<Item>)
+ makeNewItemByType(ItemType): Item
+ countByType(ArrayList<Item>, ItemType): int
+ getValue(ArrayList<Item>): double
+ getFromListByType(arrayList<Item>, ItemType): int
+getIndex(): int
+getRemainingItemsValue(): double
+getSoldItemsValue(): double
Logger
+ clerks: BufferedWriter
+ int: dat
+ Tracker: tracker
+ write(String)
+ PublishTracker()
+ close()
Tracker
+ clerkStorage: ArrayList
<TrackerDataStructure>
+ updateItemsSold(Clerk, int)
+ updateItemsPurchased(Clerk, int)
+ updateItemsDamaged(Clerk, int)
+getTrackerDataStructure(int):
TrackerDataStructure
+getClerkTrackerIndex(Cleark): int
<<Interface>>
Utility
+ rndFromRange(int, int): int
+ rnd(): double
+getPoisson(int, int, int);
+ asDollar(double): String
+ randomEnum(Class): Class
Simulation
+ store: Store
+ dayCounter:int
+ weekday: Weekday
+ startSum(int)
<<Interface>>
Console Logger
+out():

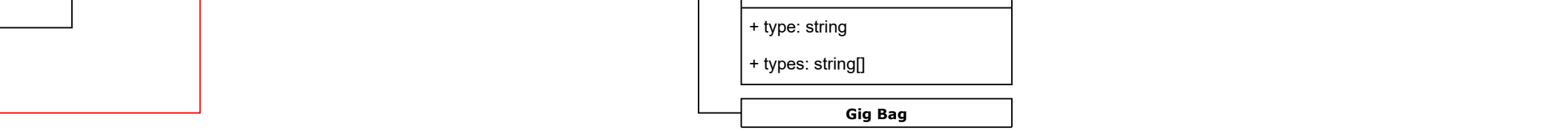

Store
+ clerks: ArrayList<Clerk>
+ activeClerk: Clerk
+ cashRegister: double
+ cashFromBank: double
+ inventory: Inventory
+ today: today
+ openToday(int)
+ getValidClerk(): Clerk
+ closedToday(int)
+getRemainingCashInRegister: double
+getTotalMoneyFromBank(): double

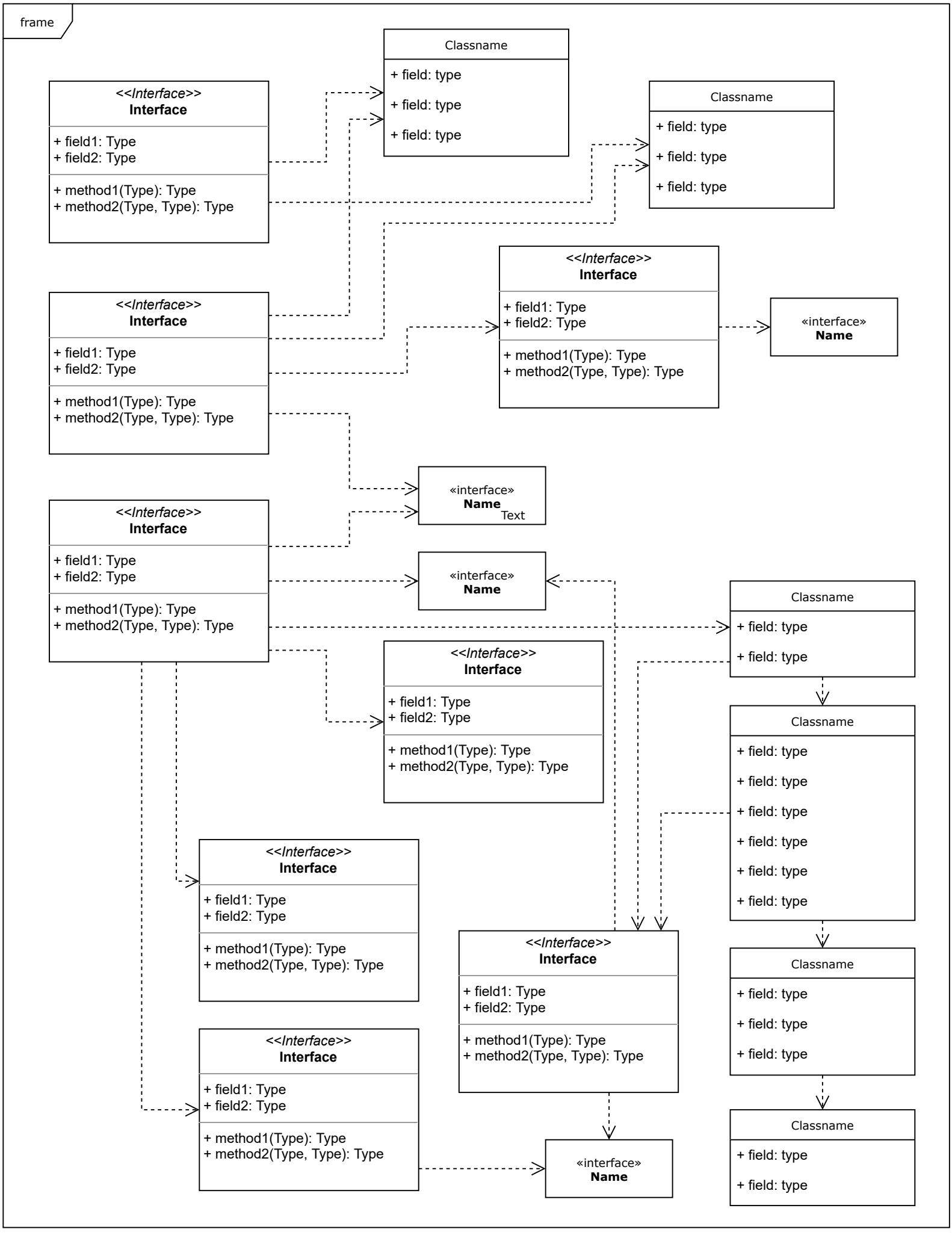

+toString(): String
+totalCost():double
+AddItemTypeToCart(ItemType)
AddCables
+toString(): String
+totalCost():double
+AddItemTypeToCart(ItemType)
AddGigBag
+toString(): String
+totalCost():double
+AddItemTypeToCart(ItemType)
AddPracticeAmps
+toString(): String
+totalCost():double
+AddItemTypeToCart(ItemType)
AddStrings
+toString(): String
+totalCost():double
+AddItemTypeToCart(ItemType)


+ startDay(int)
+ summary()
Clerk
+ name: string
+ damageChance: double
+ store: Store
+ strategy: String
+ arriveAtStore()
+ checkRegister()
+ goToBank()
+ doInventory()
+ placeAnOrder(ItemType)
+ openTheStore()
+ sellAnItem(int)
+ sellItemtoCustomer(Item, String)
+ GetItemFromInventoryByCount(int, ItemType): Item
+ buyAnItem(int)
+ buyItemFromCustomer(Item, String)
+ getPurchasePriceByCondition(Condition): double
+ cleanTheStore()
Staff
+ name: string
Observer pattern
<<Interface>>
Observer
+ arriveAtStore()
+ checkRegister()
+ goToBank()
+ doInventory()
+ placeAnOrder(ItemType)
+ openTheStore()
+ cleanTheStore()
+ leaveTheStore()
Strategy pattern
TunContext
-TuneStrategy: strategy
+gettune(): double
<<Interface>>
Tune Strategy
+tune(): double
AbstractTuneStrategy
Haphazard
Manual
Electronic


Item
+ name: string
+ purchasePrice: double
+ listPrice: double
+ isNew: boolean
+ dayArriving: integer
+ condition: Condition
+ salePrice: double
+ daySold: integer
+ itemType: ItemType
+ currentIndexForDeletion: int
+ tunable: boolean
+ tuneType: tunableType
Music
+ band: string
+ album: string
+ bands: string[]
+ albums: string[]
Cassette
Paper Score
CD
Vinyl
Players
+ equalized: boolean
CD
Record Player
MP3
Cassette Player
Instruments
Stringed
+ isElectric: boolean
+ tuned: boolean
Guitar
Bass
Mandolin
Wind
+ adjusted: boolean
Flute
+ type: string
+ types: string[]
Harmonica
+ key: string
+ keys: string[]
Saxophone
+ type: string
+ types: string[]
Clothing
Hats
+ hatSize: integer
Shirts
+ shirtSize: integer
Bandanas
Accessories
Practice Amps
+ wattage: int[]
+ wattages: int[]
Cables
+ length: float
+ lengths: float[]
Strings


+ tune():double
+ tune():double
+ tune():double


+ type: string
+ types: string[]
Gig Bag


frame
Classname
+ field: type
+ field: type
+ field: type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
Classname
+ field: type
+ field: type
+ field: type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
«interface»
Name
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
«interface»
Name
Text
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
«interface»
Name
Classname
+ field: type
+ field: type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
Classname
+ field: type
+ field: type
+ field: type
+ field: type
+ field: type
+ field: type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
Classname
+ field: type
+ field: type
+ field: type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type
«interface»
Name
Classname
+ field: type
+ field: type
+ field: type