

Solution to In-class Assignment #1

Analysis:

Inputs:

- The maximum number of stores available and the number of stores in current use.
- The number of employees working and the maximum/minimum amount needed (for the mall: custodians, security guards, managers, etc).
- The number of employees working and max./min. amount for a store.
- The amount of current inventory.
- The amount of ordered/new inventory.
- The price of inventory – *maybe a constraint?*
- Boolean value for the open/close status of the store (true means open).
- Customer's actions: buying, stealing.
- The schedule of an employee

Outputs:

- The profit, or revenue, generated from sales.
- The schedule of an employee.
- The status (open/close) of a store/mall.
- Item and quantity of item(s) bought or stolen by a customer.
- The number of customers in the mall.
- Boolean value to see if an employee is working (true for working, false for not working).

Constraints:

- If the mall is open, it cannot be closed and vice-versa.
- If the mall is already opened/closed, it, respectively, cannot be opened/closed.
- If the mall is closed, customers cannot be inside.
- If the maximum number of customers allowed in the mall **are** in the mall, then no more customers may enter the mall.
- If a store does not exist, it cannot have inventory and customers cannot be in it.
- An employee can only work 1 shift at a time.
- An employee cannot be working if he/she does not have a shift.
- An item can be returned only if it has been brought back within 30 days of the purchase.
- There cannot be more than the maximum amount of stores open.

Assumptions:

- If a store exists, it has inventory.
- A store has only 1 item to sell, and thus a single record for inventory.
- If a store has inventory, the inventory has 1 set price.
- If a store does not exist, it is possible that a mall employee is inside.
- If an item has been purchased, it can (or cannot) be returned.
- If an item has not been returned within 30 days, the store that sold that item generates a profit.

Relationships:

- A mall-employee and a store-employee are both subclasses to “Employee”.
- Each employee has 1 job title.
- Each job title has 1 salary.
- An employee shift notifies how much an employee gets paid for that day

Design

Class Oaks

{

Properties:

stores: Stores[]

mallEmps: mallEmployee[]

customers: Customer[]

numStores: Static int

numMallEmp: Static int

Methods:

open(): void //opens all the stores in the mall

close(): void //closes all stores

hireEmp(): void

hireEmp(--all properties--): void

createStore(): void //adds a new store to the array of stores

createStore(--all properties--): void

enterCustomer(): void //Customer enters the mall (has position null)..adds to Customer[]

enterCustomer(--all properties--): void

exitCustomer(Customer): void //Customer must have null position before exiting..

//deletes Customer from Customer[]

}

Class Store implements Comparable

{

Properties:

private typeStore: String //electronics, clothing, restaurant etc

private name: String //name of store

open: boolean //is it open or not? – if not, no buying/selling can occur

emps: storeEmployee[]

custs: Customers[]

items: boolean[]

numitems: int //keeps track of the first number in the array that still has an item

private price: double //price of one item in the store

private money: double //all the money the store has for transactions

Methods:

--constructors

-- all accessor methods to private properties

sell(Customer stan, int quantity) //sells a quantity of items to a person...items that it sells are made false in items[]

the quantity // money is added to the store depending on the price and

//works only if customer has enough money

//store must be open

buy(int quantity) //the store buys a quantity of items (assuming it can hold all of them)...then updates items[]

//and subtracts money from the store based on the price and quantity

//store must be open

openStore() //sets open to true

closeStore()//sets open to false

payEmployee(storeEmployee Stan) //subtracts money from the store depending on employee's pay and hours worked

//sets the employee's hoursWorked to 0

payDay(storeEmployee[] stan) //pays all employees in the store, sets all hoursWorked to 0, subtracts money from the store

compareTo(Object o) //comparable method...compares the total money the store has

```
toString()
```

```
}
```

```
Class Employee
```

```
{
```

```
Properties:
```

```
protected jobTitle: String
```

```
protected name: String
```

```
protected salary: double //per hour
```

```
protected hoursWorked //hours worked since last pay
```

```
protected Shift: String //what shift the employee works
```

```
protected rating: int //on a scale of 1 to 10 how good is he?
```

```
protected schedule: String //the schedule of an employee
```

```
protected isWorking: boolean //true if the employee is currently working
```

```
Methods:
```

```
--constructors
```

```
--accessor methods
```

```
toString() //prints all properties of the Employee
```

```
Class mallEmployee extends Employee
```

```
{
```

```
Properties:
```

```
static private numMallEmployees //a class variable that stores the total number of  
mallEmployees
```

```
Methods:
```

```
--constructors //this will also count up the numMallEmployees
```

```
}
```

```
Class storeEmployee extends Employee
```

```
{
```

```
Properties:
```

```
static private numStoreEmployees //a class variable that stores the total number of  
storeEmployees
```

```
Methods:
```

```
--constructors //this will also count up the numStoreEmployees
}
```

```
Class Customer
```

```
{
```

```
Properties:
```

```
private name: String
```

```
private money:double
```

```
private position: Store
```

```
private stuff: String[] //this keeps track of where each item was bought/stolen
```

```
Methods:
```

```
buy(Store walmart, quantity): void //substracts money from customer
```

```
    //calls stores methods to update items[] and money etc.
```

```
    //updates stuff[]
```

```
    //customer must be in the store
```

```
shopLift(Store walmart): void // subtracts money from the store and updates items[] (in  
the store)
```

```
    //updates stuff[]
```

```
    //customer must be in the store
```

```
moveTo(Store walmart): void //changes the customers position
```

```
    //makes sure that the current store deletes that customer from its
```

```
custs[]
```

```
    //makes sure that the next store adds the customer to its custs[]
```

```
    //must move to mall (null position) before he/she can exit
```

```
toString(): String //returns a string with all the Customer's properties
```

```
}
```

