**Vending Machine Design**

**Contents**

[**1.UML design** 1](#_Toc194071229)

[**2.User Story Task** 2](#_Toc194071230)

[**Task US1** 2](#_Toc194071231)

[**addCar** 2](#_Toc194071232)

[**loadCarDataFromFile** 3](#_Toc194071233)

[**TaskUS2:** 4](#_Toc194071234)

[**displayVendingMachine** 4](#_Toc194071235)

[**TaskUS3** 5](#_Toc194071236)

[**displaySorted1DArray** 5](#_Toc194071237)

[**sortByPrice** 5](#_Toc194071238)

[**sortByYear** 6](#_Toc194071239)

[**TaskUS4:** 7](#_Toc194071240)

[**retrieveCars** 7](#_Toc194071241)

[**TaskUS5** 8](#_Toc194071242)

[**displayMenu** 8](#_Toc194071243)

[**3. Main method Signature and Algorithms** 9](#_Toc194071244)

# **1.UML design**

|  |
| --- |
| VendingMachine |
| -floor: int  -space: int |
| +VendingMachine(int floor, int space)  +addCar (): void  +loadCarDataFromFile() : void  +displayVendingMachine(): void  +sortByPrice(): void  +sortByYear(): void  +retrieveCar (): void |

|  |
| --- |
| Car |
| -carBrand: String  -carModel: String  -carPrice: double  -carYear: int |
| +Car(String branch, model , double price, int year)  +getBrand(): String  +getModel(): String  +getPrice(): double  +getYear(): int |

# **2.User Story Task**

**User Story 1:** As a dealership owner, I want to store cars in a vending machine at their designated positions based on reading a file so we can keep track of the cars in the tower.

Scenario:

Given a vending machine with defined rows and columns entered by the dealer,

When file is read containing car details,

*Then the cars are placed into the vending machine at their designated positions.*

*Cars are only placed in valid positions within the vending machine.*

*A car cannot be placed in an already occupied slot, preventing overwrites.*

*Out-of-bounds positions are not allowed*.

**Task US1: ETA 2 hrs**

Create vending machines class

Create Car class

Add cars to vending machines

Read from File

If else statement

Clean up

### **addCar**

**TDD:**

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| Vending machine has valid position:  Emty at (1,1). Add car at (1,1) | Car adds to valid position  Car store at (1,1) | Valid position |
| Vending machine don’t have valid position:  Has car at (1,2), try to add another car at (1,2) | Error: position (1,2) is already occupied. Can’t not add car | Occupied position |
| Vending machine don’t have valid position:  There are no (5,1) position. Add car A to (5,1) | Error: Invalid position at (5,1). Car A can not be added | Out of given postition |

\

**Method Signature and Algorithms:**

public static void **addCar** (Car[][] carTower, int floor, int space, Car carToAdd)

if floor or space < 0 or floor or space > carTower.length

println :

Error: Invalid position at floor and space

Can not place car detail

else if carTower at floor and space != null

println:

Error: Slot at floor and space already occupied

Car detai can’t not be placed

else carTower[floor][space] = carToAdd

### **loadCarDataFromFile**

**TDD:**

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| Wrong file name: **car1**.txt | File was not found  File car1.txt was not found |  |
| Correct file name: **cars1**.txt  carTower with 4 floor and 5 space | display  Error: Invalid position at Floor: 5 Space: 3  Can not place Car Nissan Altima 2017 - $20000.0 |  |

**Method Signature and Algorithms:**

public void **loadCarDataFromFile**(*Car[][] carTower, String filename* ) {

***try***

Create a Scanner object to read file

***While***: fileScanner.hasNextLine

*//read car data from file*:

int floor = fileScanner.nextInt()

int space = fileScanner.nextInt()

int year = fileScanner.nextInt()

double price = fileScanner.nextDouble()

String brand = fileScanner.next()

String model = fileScanner.next()

*//call addCar method to add car detail to carTower:*

Car aCar = new Car(brand, model, year, price)

**addCar**(carTower, floor, space, aCar)

Close Scanner

***catch***

FileNotFoundException e

println: filename was not found

**User Story 2:** As a dealership employee, I want to **view the location** of all the cars **in**

**the vending machine** so I can show a customer.

Given a vending machine containing cars,

When I request an inventory location report,

Then the cars are printed

Display if empty if no car found at location.

**TaskUS2: EAT 1hr**

### **displayVendingMachine**

**TDD:**

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| carTower at this position empty | EMPTY |  |
| carTower at this position:  carTower[0][1] = Ford Mustang 2019 - $26000.0 | Space 1: Ford Mustang 2019 - $26000.0 |  |

**Method Signature and Algorithms:**

Public void **displayVendingMachine** (Car[][] carTower)

***for*** ( int row = 0 , row < carTower.length, row ++)

println “Floor ” + (row+1) “ :”

***for*** ( int col = 0 , col < carTower[row].length, col ++)

***if*** carTower[row][col] = null

println “Space ” + “ EMPTY”

***else*** println “Space ” + (col +1) + “: ”

+ carTower[row][col].getBrand () + “ “

+ carTower[row][col].getModel() + “ “

+ carTower[row][col].getYear + “ - “

+ carTower[row][col].getPrice

**User Story 3:** As a dealership employee, I want to **view an inventory report sorted by car price or by year** so that I can easily identify the cars.

Given a vending machine containing cars,

When I request an inventory report,

Then the cars are sorted and displayed based on my selection of “price” or “year” from lowest to highest.

**TaskUS3: EAT 4hrs**

### **displaySorted1DArray**

**TDD:**

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| Toyota Prius 2017 - $22000.0  Honda Accord 2019 - $2000.0  Ford Mustang 2019 - $26000.0  Toyota Corolla 2018 - $24000.0 | Toyota Prius 2017 - $22000.0  Honda Accord 2019 - $2000.0  Ford Mustang 2019 - $26000.0  Toyota Corolla 2018 - $24000.0 | Regular |
| Null | Empty car | Empty |

**Method Signature and Algorithms:**

public static void **displaySorted1DArray**(car[] cars)

***for*** ( int i = 0 , i < cars, i ++)

println:

+ carTower[row][col].getBrand () + “ “

+ carTower[row][col].getModel() + “ “

+ carTower[row][col].getYear + “ - “

+ carTower[row][col].getPrice

### **sortByPrice**

**TDD:**

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| Toyota Prius 2017 - $22000.0  Honda Accord 2019 - $2000.0  Ford Mustang 2019 - $26000.0  Toyota Corolla 2018 - $24000.0 | Honda Accord 2019 - $2000.0  Toyota Prius 2017 - $22000.0  Toyota Corolla 2018 - $24000.0  Ford Mustang 2019 - $26000.0 | Regular |
| Honda Accord 2019 - $2000.0 | Honda Accord 2019 - $2000.0 | Single Element |
| Honda Accord 2019 - $2000.0  Toyota Prius 2017 - $22000.0  Toyota Corolla 2018 - $24000.0  Ford Mustang 2019 - $26000.0 | Honda Accord 2019 - $2000.0  Toyota Prius 2017 - $22000.0  Toyota Corolla 2018 - $24000.0  Ford Mustang 2019 - $26000.0 | Already Sorted |
| Ford Mustang 2019 - $26000.0  Toyota Corolla 2018 - $24000.0  Toyota Prius 2017 - $22000.0  Honda Accord 2019 - $2000.0 | Honda Accord 2019 - $2000.0  Toyota Prius 2017 - $22000.0  Toyota Corolla 2018 - $24000.0  Ford Mustang 2019 - $26000.0 | Reverser Sorted |
| Toyota Prius 2017 - $22000.0  Honda Accord 2019 - $2000.0  Ford Mustang 2019 - $26000.0  Toyota Corolla 2018 - $20000.0 | Honda Accord 2019 - $2000.0  Toyota Corolla 2018 - $20000.0  Toyota Prius 2017 - $22000.0  Ford Mustang 2019 - $26000.0 | Duplicates |

**Method Signature and Algorithms**

public void **sortByPrice**(Car[][] carTower)

*(Extract all cars from the 2D array into a 1D list)*

int numRow = carTower.length;

int numCol = carTower[row].length

Car oneDCars = new Car [ numRow\*numCol ]

int index = 0

***for*** ( int row = 0 , row < carTower.length, row ++)

***for*** ( int col = 0 , col < carTower[row].length, col ++)

oneDCars[index] = carTower[row][col]

update ***index****++*

(Create a 1D array of non-null cars)

Car[] arrayToSort = new Car[***index***];

int index1 = 0

***for*** ( int i = 0; i < oneDCars.length; i++)

***if*** (oneDCars[i] != null)

arrayToSort[index1] = oneDCars[i]

update index1++

*(Sort the* 1D array of non-null cars *using the Insert Sort)*

f***or*** ( int i = 1; i < arrayToSort.length; i++)

Car key = arrayToSort [i]

int j= i

***while*** (j > 0 && arrayToSort [j-1].getPrice > key. getPrice)

arrayToSort [j] = arrayToSort [j-1]

j--

arrayToSort [j] = key

*(Display sorted array)*

Println : Sorted Inventory by Price

**displaySorted1DArray**()

### **sortByYear**

Same sortByPrice, change getPrice by getYear to compare:

while (j > 0 && arrayToSort [j-1].**getYear** > key. **getYear**)

**User Story 4**: As a dealership employee, I want **retrieve cars from a location for clients** to test drive

Scenario:

Given a vending machine containing cars,

When I request a car by floor and space,

Then the car is retrieved

Display if no car found at location.

Display car retrieved and the details if found

**TaskUS4: EAT 1hr**

### **retrieveCars**

**TDD:**

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| Car[0][0] carTower empty | No car located at Floor 1 Location 1 |  |
| Car[2][3] carTower : Ford Mustang 2019 - $26000.0 | Car retrieved from Floor 3 Location 4: Ford Mustang 2019 - $26000.0 |  |

**Method Signature and Algorithms**:

Public void **retrieveCars**(int floor, int space, Car[][] carTower)

***if*** carTower[floor-1][space-1] = null

println “No car located at Floor” + floor + “ Location” + space

***else*** println “Car retrieved from Floor ” + floor + “ Location” + space + “ : “

+ carTower[floor -1][space -1].getBrand () + “ “

+ carTower[floor -1][space -1].getModel() + “ “

+ carTower[floor -1][space -1].getYear + “ - “

+ carTower[floor -1][space -1].getPrice

**User Story 5**: As a dealership employee, I want **a menu driven system to select an action**.

Scenario:

Given a menu list,

When I input a number,

That action is completed

**TaskUS5: EAT 1hr**

### **displayMenu**

**TDD**:

|  |  |  |
| --- | --- | --- |
| Precondition | Postcondition | Comment |
| === Car Vending Machine Menu ===  1. Load Car Data  2. Display Vending Machine  3. Retrieve a Car  4. Print Sorted Inventory (Price)  5. Print Sorted Inventory (Year)  6. Exit | === Car Vending Machine Menu ===  1. Load Car Data  2. Display Vending Machine  3. Retrieve a Car  4. Print Sorted Inventory (Price)  5. Print Sorted Inventory (Year)  6. Exit |  |

**Method Signature and Algorithms**

Public static void displayMenu()

println

=== Car Vending Machine Menu ===

1. Load Car Data

2. Display Vending Machine

3. Retrieve a Car

4. Print Sorted Inventory (Price)

5. Print Sorted Inventory (Year)

6. Exit

# **3. Main method Signature and Algorithms**

**public** **static** **void** main(String[] args)

Create Scanner object

int ***choiceOption***

Using Scanner to input ***floor*** and ***space*** of vendingMachine

Create ***vendingMachine*** object and 2D Car array( ***carTower***) base on given ***space*** and ***floor***

***do***

**displayMenu**

input ***choiceOption***

***Switch (****choiceOption****)***

case 1***: loadCarDataFromFile***

case 2***: displayVendingMachine***

case 3: ***retrieveCars***

case 4: ***sortByPrice***

case 5: ***sortByYear***

case 6: Exiting program. Goodbye!

***default***: Please enter an option from 1 to 6

***while*** choiceOption != 6