Ivan Solodyankin

Simon Mai

Dr. Kent Jones

CS 172

12 December 2018

Final Project Specifications

The implementation of the food court inspired final project is as follows: there will be six classes, with five of those classes representing a store one would find in a food court. The base class, called store, will hold the control structure for the eater to select a store of their choice to eat from. This control structure will be in the form of a switch statement that will instruct the user to pick a store using their designated number. Once a store is selected, an object of that store constructed beforehand will be initiate several member functions of the store class chosen.

The workings of a specific store class will be as follows: a vector of objects comprising of that class (for example the Pizza class will have Pizza objects) will be created, and six menu items will be put into the vector as objects. Each object has a name and a price, made by a constructor. The user will then see that vector displayed before them and they are able to pick one item of their choice using the designated number. Once they pick their item, another vector will be created that will hold the orders. If the first numbered food item was selected for example, the index of that food item will be read and put into the orders vector. The user is then directed to buy a drink if desired, and this follows a similar format to the food items as described previously. After both food and drink have been ordered, the user then must see a summary of their order. This order summary will then also be printed onto a “receipt” (text file), for which there will be two: a local receipt for just that store (customer copy) and a master receipt that will hold all orders made by the user for use by the businesses.

After purchasing their food and drink from a venue, the user will then be directed if they wish to make more purchases. Should they choose yes, they are able to go back to the venue they were at previously or they can choose a new venue. This will be done with a do-while loop to ensure that the person visits a venue at least once before visiting it again or another venue.

Project Design (Class UMLs)

|  |
| --- |
| store |
| -foodItem: string |
| -drinkItem: string |
| -foodItemPrice: double |
| - orders: vector <string> |
| -drinkItemPrice: double |
| +Store() |
| +Store(food: string, drink:string) |
| +displayStore() : void |
| +selectStore() : void |
| +displayDrinks(d: vector <string>) : void |
| +drinkOrders: vector <string> |

|  |
| --- |
| pizza |
| -pizza: string |
| -pdrink: string |
| +pizza() |
| +pizza(pizza: string, price: double) |
| +pizzas: vector <string> |
| +porders: vector <pizza> |
| +getPizza(): string |
| +getPdrink(): string |
| +pprice: double |
| +pdrinkPrice: double |

|  |
| --- |
| +getPrice(): double |
| +selectPizza(): void |
| +displayVectorP(&: vector <pizza>) void |
| +calcPrice(): double |
| +selectDrink(): void |
| +displayOrder(): void |
| +displayDrinksP(p: vector <string>): void |
| receiptOutput(): void |

|  |
| --- |
| taco |
| -taco: string |
| -tdrink: string |
| +taco() |
| +taco(taco: string, price: double) |
| +tacos: vector <string> |
| +torders: vector <taco> |
| getTaco() : string |
| getTdrink() : string |
| +tprice: double |
| +tdrinkPrice: double |

|  |
| --- |
| +getPrice(): double |
| +selectTaco(): void |
| +displayVectorT(&: vector <taco>) void |
| +calcPrice(): double |
| +selectDrink(): void |
| +displayOrder(): void |
| +displayDrinksT(p: vector <string>): void |
| receiptOutput(): void |

|  |
| --- |
| icecream |
| -icecream: string |
| -idrink: string |
| +icecream() |
| +icecream(icecream: string, price: double) |
| +icecreamlist: vector <string> |
| +iceorders: vector <icecream |
| getIceCream() : string |
| getIdrink() : string |
| +iprice: double |
| +idrinkPrice: double |

|  |
| --- |
| +getPrice(): double |
| +selectIceCream(): void |
| +displayVectorI&: vector <icecream>) void |
| +calcPrice(): double |
| +selectDrink(): void |
| +displayOrder(): void |
| +displayDrinksI(p: vector <string>): void |
| receiptOutput(): void |

|  |
| --- |
| chinese |
| -chinese\_ : string |
| -cdrink\_: string |
| +chinese() |
| +chinese(chinese: string, price: double) |
| +chineseFood: vector <string> |
| +corders: vector <taco> |
| getChinese() : string |
| getCdrink() : string |
| +cprice: double |
| +cdrinkPrice: double |

|  |
| --- |
| +getPrice(): double |
| +selectChinese(): void |
| +displayVectorC(&: vector <chinese>) void |
| +calcPrice(): double |
| +selectDrink(): void |
| +displayOrder(): void |
| +displayDrinksC(p: vector <string>): void |
| receiptOutput(): void |

|  |
| --- |
| burger |
| -burger: string |
| -bdrink: string |
| +burger() |
| +burger(burger: string, price: double) |
| +burgers: vector <string> |
| +orders: vector <burger> |
| getBurger() : string |
| getBdrink() : string |
| +price: double |
| +drinkPrice: double |

|  |
| --- |
| +getPrice(): double |
| +selectBurger(): void |
| +displayVector(&: vector <burger>) void |
| +calcPrice(): double |
| +selectDrink(): void |
| +displayOrder(): void |
| +displayDrinks(p: vector <string>): void |
| receiptOutput(): void |