Object Oriented Programming Project

Documentation

Team 4:

Layan Alfaghawe: 20220255

Leen Iyad: 20220888

Hussam Rasheed: 20220426

Essam Khader: 20220923

Table of Contents

Work Division…………………………………………………………………………….3

Classes……………………………………………………………………………….……4

Date Class………………………………………………………………………………………...…4

Address Class…………………………….………………………………….……………….…..…5

Customer Class………………………………………………………………………...…………...6

Medication Class…………………………………………………………………………………...7

OffTheShelf Class………………………..………………………………………………………...9

Prescription Class…………………………………………………………………………………10

Pharmacy Class……………………..……………………………………………………………..11

User Interface……………………………………………………………….…………..16

Menus……………………………………………………………………………………..….….…17

Set and Resize Pharmacies…………………………………………………………………..........17

Edit Pharmacies……………………………………………………………………………...……18

Displays……………………………………………………………………………….……….…...19

int main()……………………………………………………………………………..……….…...20

|  |  |
| --- | --- |
| **Work Division** | |
| **Phase 1** | |
| Layan | purchase function and user interface |
| Leen | customer class |
| Hussam | functions addCust, addMed, displayAv |
| Essam | medication class |
| **Phase 2** | |
| Layan | user interface and edits to previous classes |
| Leen | date class |
| Hussam | address class |
| Essam | pharmacy class |
| **Phase 3** | |
| Layan | user interface and managing dynamic arrays |
| Leen | off the shelf class |
| Hussam | prescription class |
| Essam | Managing dynamic arrays |

Classes:

Date Class

Private data attributes

* day of type integer
* month of type integer
* year of type integer

Public member functions

Constructors

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters:
* An integer d, which is defaulted to 1
* An integer m, which is defaulted to 1
* An integer y, which is defaulted to 2000

We call setters to make sure that all the default values are validated.

* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference, “src”.

Setters

* A setter for day, setDay, which is a member function of type void that takes an integer ‘d’ as a parameter. It checks to make sure that ‘d’ is between 1 and 31 and sets day to it.
* A setter for month, setMonth, which is a member function of type void that takes an integer ‘m’ as a parameter. It checks to make sure that ‘m’ is between 1 and 12 and sets month to it.
* A setter for year, setYear, which is a member function of type void that takes an integer ‘y’ as a parameter. It checks to make sure that ‘y’ is greater than 2000 and sets year to it.

Getters

* A getter for day, getDay, which is a member function of type integer that returns the day.
* A getter for month, getMonth, which is a member function of type integer that returns the month.
* A getter for year, getYear, which is a member function of type integer that returns the year.

Print

* A print member function of type void that prints the date.

Address Class

Private data members

* email: an email of type string
* city: a city name of type string
* mobile\_no: a mobile number of type string
* street\_name: a street name of type string

Public member functions

Constructors

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters:
* A string e, which is defaulted to “No Email”
* A string c, which is defaulted to “No City Name”
* A string m, which is defaulted to “No Mobile Number”
* A string s, which is defaulted to “No Street Name”
* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference “src”.

Setters

* A setter for email, setEmail, which is a member function of type void that takes a string ‘e’ as a parameter. It checks to make sure ‘e’ is not empty and sets email to it.
* A setter for city, setCity, which is a member function of type void that takes a string ‘c’ as a parameter. It checks to make sure ‘c’ is not empty and sets city to it.
* A setter for mobile\_no, setMobileNo, which is a member function of type void that takes a string ‘n’ as a parameter. It checks to make sure ‘n’ is not empty and sets mobile\_no to it.
* A setter for street\_name, setStreet, which is a member function of type void that takes a string ‘s’ as a parameter. It checks to make sure ‘s’ is not empty and sets street\_name to it.

Getters

* A getter for email, getEmail, which is a member function of type string that returns the email.
* A getter for city, getCity, which is a member function of type string that returns the city.
* A getter for mobile\_no, getMobile, which is a member function of type string that returns the mobile number.
* A getter for street\_name, getStreet, which is a member function of type string that returns the street name.

Print

* A print member function that prints the address details.

Customer Class

Private data members

* customer\_ID: a customer ID of type integer
* customer\_name: a customer name of type string
* address: an address of type address

Public member functions

Constructors

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters.
* An integer i, which is defaulted to 0
* A string n, which is defaulted to “No Name”
* A string e, which is defaulted to “No Email”
* A string c, which is defaulted to "No City Name”
* A string m, which is defaulted to "No Mobile Number”
* A string s, which is defaulted to "No Street Name”
* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference “src”.

Setters

* A setter for customer\_ID, setID, which is a member function of type void that checks if the passed integer is less than 0, and if it is, will ask the user to input a new ID.
* A setter for customer\_name, setName, which is a member function of type void that checks if the passed string isn’t empty and set it to name data member.
* A setter for address, setAddress, which is a member function of type void that uses the address’s setters to set the object.

Getters

* A getter for customer\_ID, getID, which is a member function of type integer that returns customer\_ID.
* A getter for customer\_name, getName, which is a member function of type string that returns customer\_name.
* A getter for address, getAddress, which is a member function of type address that returns address.

Print

* A print member function of type void that prints the customer’s information.

Medication Class

Private data members

* medication\_ID: a medication ID of type integer
* medication\_name: a medication name of type string
* description: a description of type string
* price: a price of type double
* quantity\_in\_stock: quantity of medication in stock of type integer
* expiry\_date: an expiration date of type date
* barcode: a barcode of type string

Public member functions

Constructors

Public member functions :

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters.
* A string i, which is defaulted to 0
* A string m, which is defaulted to “No Name”
* A string des, which is defaulted to "No Description”
* A double p, which is defaulted to 0
* An integer q, which is defaulted to 0
* A string b, which is defaulted to “No Barcode”
* An integer day, which is defaulted to 1
* An integer month, which is defaulted to 1
* An integer year, which is defaulted 2000
* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference “src”.

Setters

* A setter for medication\_ID, setID, which is a member function of type void that checks if the passed integer is less than 0, and if it is, will ask the user to input a new ID.
* A setter for medication\_name, setMedName, which is a member function of type void that checks if the passed string isn’t empty and set it to name data member.
* A setter for description, setDes, which is a member function of type void that checks if the passed string isn’t empty and set it to name data member.
* A setter for price, setPrice, which is a member function of type void that takes a double x as a parameter. It checks to make sure that x is positive and sets the price to it if so.
* A setter for quantity\_in\_stock, setQuantity, which is a member function of type void that takes an integer x as a parameter. It checks to make sure that x is positive or 0 and sets the quantity\_in\_stock to it if so.
* A setter for expiry\_date, setDate, which is a member function of type void that uses the date’s setters to set the date.
* A setter for barcode, setBarcode, which is a member function of type void that checks if the passed string isn’t empty and set it to name data member.

Getters

* A getter for medication\_ID, getID, which is a member function of type integer that returns the ID.
* A getter for medication\_name, getMedName, which is a member function of type string that returns medication\_name.
* A getter for description, getDes, which is a member function of type string that returns the description.
* A getter function for price, getPrice, which is a member function of type double that returns the price.
* A getter for quantity\_in\_stock, getQuantity, which is a member function of type integer that returns quantity\_in\_stock.
* A getter for date, getDate, which is a member function of type date that returns the date.
* A getter for barcode, getBarcode, which is a member function of type string that returns the barcode.

print

* A print member function of type void that prints the medication’s information.

OffTheShelf Class

Private data members

* BOGOF: a boolean that represents the buy 1 get 1 free offer
* offer\_ends: the date when the offer ends of type date
* today: today’s date of type date

Public member functions

Constructor

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters.
* A boolean, b which is defaulted to false
* An integer d, which is defaulted to 1
* An integer m, which is defaulted to 1
* An integer y, which is defaulted to 2000
* All the variables from the medication class, which are sent to the medication’s constructor
* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference “src”.

Setters

* A setter for BOGOF, setBOGOF, which is a member function of type void that takes a boolean s as a parameter and sets BOGOF to it. It then adds 3 months to the offer\_ends date if BOGOF is true, and 2 years if it is false.
* A setter for offer\_ends, setOfferEnds, which is a member function of type void that takes 3 integer parameters and uses the date setters to set it.

Getters

* A getter for BOGOF, getBOGOF, which is a member function of type Boolean that returns BOGOF.
* A getter for offer\_ends, getOfferEnds, which is a member function of type date that returns offer\_ends.
* A getter for today, getToday, which is a member function of type date that returns today.

Print

* A print member function of type void that prints the off the shelf medication’s information, using the medication’s print function as well.

Prescription Class

Private data members

* FDA\_number: an FDA number of type integer
* approval\_date: a date that represents the approval date

Public member functions

Constructors

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters.
* An integer, fda which is defaulted to 0
* An integer d, which is defaulted to 1
* An integer m, which is defaulted to 1
* An integer y, which is defaulted to 2000
* All the variables from the medication class, which are sent to the medication’s constructor
* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference “src”.

Setters

* A setter for FDA\_number, setFDA, which is a member function of type void that takes an integer ‘n’ as a parameter. It checks to make sure that ‘x’ is positive or 0 and sets the FDA\_number to it.
* A setter for approval\_date, setAppDate, which is a member function of type void that takes 3 integer parameters and uses the date setters to set it.

Getters

* A getter for FDA\_number, getFDA, which is a member function of type integer that returns the FDA\_number.
* A getter for approval\_date, getAppDate, which is a member function of type date that returns the approval\_date.

Print

* A print member function of type void that prints the prescription medication’s information, using the medication’s print function as well.

Pharmacy Class

Private data members

* pharmacy\_id: a pharmacy ID of type integer
* pharmacy\_name: a pharmacy name of type string
* meds: a pointer to an array of medications
* shelf: a pointer to an array of off the shelf medications
* pres: a pointer to an array of prescription medications
* ‘m’, ‘sm’, ‘pm’, ‘c’: integers that count the current number of medications, off the shelf medications, prescription medications and customers respectively
* msize, smsize, pmsize, csize: integers that show the array size of meds, shelf, pres and cust respectively

Public member functions

Constructors

* A defaulted constructor with default parameter values that it will assign to the data members if they are not provided. This constructor has the following parameters.
* An integer i, which is defaulted to 0
* A string n, which is defaulted to “No Name”
* All the pointers are set to NULL
* All the counters are set to 0
* All the array sizes are set to 0
* A copy constructor that is used to create a new object by copying the values from an existing object passed as a constant reference “src”.

Setters

* A setter for pharmacy\_ID, setID, which is a member function of type void that checks if the passed integer is less than 0, and if it is, will ask the user to input a new ID.
* A setter for pharmacy\_name, setName, which is a member function of type void that checks if the passed string isn’t empty and set it to name data member.
* A setter for meds, setMedP, which is a member function of type void that receives a medication pointer and sets meds to it.
* A setter for shelf, setShelfP, which is a member function of type void that receives a medication pointer and sets shelf to it.
* A setter for pres, setPresP, which is a member function of type void that receives a medication pointer and sets pres to it.
* A setter for cust, setCustP, which is a member function of type void that receives a medication pointer and sets cust to it.
* A setter for all the other integers, setInts, which is a member function of type void that receives 8 integers and sets the array counters and sizes to them.

Getters

* A getter for pharmacy\_ID, getID, which is a member function of type integer that returns pharmacy\_ID.
* A getter for pharmacy\_name, getName, which is a member function of type string that returns pharmacy\_name.
* A getter for meds, getMeds, which is a member function that returns meds as a medication pointer.
* A getter for shelf, getShelf, which is a member function that returns shelf as an offtheshelf pointer.
* A getter for pres, getPres, which is a member function that returns pres as a prescription pointer.
* A getter for cust, getCust, which is a member function that returns cust as a customer pointer.
* A getter for msize, getMsize, which is a member function of type int which returns msize.
* A getter for smsize, getSMsize, which is a member function of type int which returns smsize.
* A getter for pmsize, getPMsize, which is a member function of type int which returns pmsize.
* A getter for csize, getCsize, which is a member function of type int which returns csize.

Copy Functions

* copyMeds: which is a void member function that takes a medication array and its size. It checks to see if meds points to NULL, if id does not, then the array it is pointing to is deleted at first. msize is set to the size, and meds points to a dynamically created array. Then the values are copied from the received array to the dynamic one.
* copyShelf: which is a void member function that takes an offtheshelf array and its size. It checks to see if shelf points to NULL, if it does not, then the array it is pointing to is deleted at first. smsize is set to the size, and shelf points to a dynamically created array. Then the values are copied from the received array to the dynamic one.
* copyPres: which is a void member function that takes a prescription array and its size. It checks to see if pres points to NULL, if it does not, then the array it is pointing to is deleted at first. pmsize is set to the size, and pres points to a dynamically created array. Then the values are copied from the received array to the dynamic one.
* copyCust: which is a void member function that takes a customer array and its size. It checks to see if cust points to NULL, if it does not, then the array it is pointing to is deleted at first. csize is set to the size, and cust points to a dynamically created array. Then the values are copied from the received array to the dynamic one.

Resize Medication

* addMed: which is a void member function that adds a medication to the mth element in the meds array. And adds 1 to m each time to increase the counter of medications.
* resizeMeds: which is a void member function that takes an integer ‘n’ as a parameter.
* If meds is pointing to NULL, a medication array is created dynamically and meds points to it. The size of the array is ‘n’ and then addMed is called ‘n’ times to add ‘n’ medications to the array.
* If meds is not pointing to NULL, a temp array is created dynamically with the size of msize+n.
* The elements of the array are all copied from meds to the temp, and then the smaller array is deleted. meds points to the other array and temp points to NULL.
* Then ‘n’ medications are added to the array.
* removeMed: which is a void member function that takes an integer ‘id’ as a parameter.
* If meds is not pointing to NULL, then we search the medication array for ‘id’, if found, a flag, index, is set to the index of that id.
* If not, index remains at -1, and the user is notified that the id was not found.
* In the case the id is found, msize and m are both decremented by 1 and a temp medication array is created.
* We copy the elements from meds to temp, from index 0 to index, and then we copy the elements of meds[i+1] into temp[i].
* The larger array is deleted, meds points to the other array and temp points to NULL. And the user is notified of the success of the operation.

Resize OffTheShelf

* addShelf: which is a void member function that adds an offtheshelf medication to the smth element in the shelf array. And adds 1 to sm each time to increase the counter of offtheshelf medications.
* resizeshelf: which is a void member function that takes an integer ‘n’ as a parameter.
* If shelf is pointing to NULL, an offtheshelf array is created dynamically and shelf points to it. The size of the array is ‘n’ and then addShelf is called ‘n’ times to add ‘n’ offtheshelf medications to the array.
* If shelf is not pointing to NULL, a temp array is created dynamically with the size of smsize+n.
* The elements of the array are all copied from shelf to the temp, and then the smaller array is deleted. shelf points to the other array and temp points to NULL.
* Then ‘n’ offtheshelf medications are added to the array.
* removeShelf: which is a void member function that takes an integer ‘id’ as a parameter.
* If shelf is not pointing to NULL, then we search the shelf array for ‘id’, if found, a flag, index, is set to the index of that id.
* If not, index remains at -1, and the user is notified that the id was not found.
* In the case the id is found, smsize and sm are both decremented by 1 and a temp offtheshelf medication array is created.
* We copy the elements from shelf to temp, from index 0 to index, and then we copy the elements of shelf[i+1] into temp[i].
* The larger array is deleted, shelf points to the other array and temp points to NULL. And the user is notified of the success of the operation.

Resize Prescription

* addPres: which is a void member function that adds a prescription medication to the pmth element in the pres array. And adds 1 to pm each time to increase the counter of prescription medications.
* resizePres: which is a void member function that takes an integer ‘n’ as a parameter.
* If pres is pointing to NULL, a prescription medication array is created dynamically and pres points to it. The size of the array is ‘n’ and then addPres is called ‘n’ times to add ‘n’ prescription medications to the array.
* If pres is not pointing to NULL, a temp array is created dynamically with the size of pmsize+n.
* The elements of the array are all copied from pres to the temp, and then the smaller array is deleted. pres points to the other array and temp points to NULL.
* Then ‘n’ prescription medications are added to the array.
* removePres: which is a void member function that takes an integer ‘id’ as a parameter.
* If pres is not pointing to NULL, then we search the prescription medication array for ‘id’, if found, a flag, index, is set to the index of that id.
* If not, index remains at -1, and the user is notified that the id was not found.
* In the case the id is found, pmsize and pm are both decremented by 1 and a temp prescription medication array is created.
* We copy the elements from pres to temp, from index 0 to index, and then we copy the elements of pres[i+1] into temp[i].
* The larger array is deleted, pres points to the other array and temp points to NULL. And the user is notified of the success of the operation.

Resize Customer

* addCust: which is a void member function that adds a customer to the cth element in the cust array. And adds 1 to c each time to increase the counter of customer.
* resizeCust: which is a void member function that takes an integer ‘n’ as a parameter.
* If cust is pointing to NULL, a customer array is created dynamically and cust points to it. The size of the array is ‘n’ and then addMed is called ‘n’ times to add ‘n’ customers to the array.
* If cust is not pointing to NULL, a temp array is created dynamically with the size of csize+n.
* The elements of the array are all copied from cust to the temp, and then the smaller array is deleted. cust points to the other array and temp points to NULL.
* Then ‘n’ customers are added to the array.
* removeCust: which is a void member function that takes an integer ‘id’ as a parameter.
* If cust is not pointing to NULL, then we search the customer array for ‘id’, if found, a flag, index, is set to the index of that id.
* If not, index remains at -1, and the user is notified that the id was not found.
* In the case the id is found, csize and c are both decremented by 1 and a temp customer array is created.
* We copy the elements from cust to temp, from index 0 to index, and then we copy the elements of cust[i+1] into temp[i].
* The larger array is deleted, cust points to the other array and temp points to NULL. And the user is notified of the success of the operation.

Array Counters

* A getter for m, numMeds, which is a member function of type integer that returns m.
* A getter for sm, numShelf, which is a member function of type integer that returns sm.
* A getter for pm, numPres, which is a member function of type integer that returns pm.
* A getter for c, numCust, which is a member function of type integer that returns c.

Print

* print: which is a member function of type void that prints the pharmacy’ ID, name and number of medications, offtheshelf medications, prescription medications and customers
* printMeds: which is a member function of type void that prints all the available medications if m is not 0.
* printShelf: which is a member function of type void that prints all the available medications if sm is not 0.
* printPres: which is a member function of type void that prints all the available medications if pm is not 0.
* printCust: which is a member function of type void that prints all the available medications if c is not 0.

Total value of medications

* totalValM: which is a member function of type double that calculates the sum of every medication in the array multiplied by its quantity.
* totalValS: which is a member function of type double that calculates the sum of every offtheshelf medication in the array multiplied by its quantity.
* totalValP: which is a member function of type double that calculates the sum of every prescription medication in the array multiplied by its quantity.

Destructor

* Which checks if any of the array pointers are pointing to NULL, if they are not, the arrays stored in the heap are deleted.

User Interface

The user interface of the system is the source file containing the main function which runs the code, it was created as follows:

There is a single global variable, ‘p’, which is used to count the number of pharmacies currently in the array. There is also a pharmacy pointer which is set to NULL, this is used to create a dynamic array of pharmacies.

Menus

printMenu1()

* This function simply prints the menu showcasing the possible operations that can be performed by the system.

printMenu2(int id)

* This function simply prints the edit menu for the pharmacy chosen by the user, using the id integer it receives.

printMenu3(int id)

* This function simply prints the display menu for the pharmacy chosen by the user, using the id integer it receives.

printMenu4()

* This function simply prints the resize menu for the pharmacies.

medType()

* This function has a menu showcasing the 3 types of medications and takes input from the user of the type they want. It returns the choice as an integer.

Set and Resize Pharmacies

setPharmacies(int &size)

* This function asks the user to input the number of pharmacies they want, and if the phar pointer is not pointing to NULL, will set p to 0 and will delete the array phar is pointing to.
* phar is then made to point to a dynamically created array of the size the user entered, and the addPharmacy() function is called that many times.

addPharDyn(int &size)

* this function will ask the user to input the number of pharmacies they wish to add and will then increase the size by that amount.
* A temp pointer is created and points to a dynamically created array of the new size, then the elements of the smaller array are copied to the temp array.
* The phar array is now deleted, and phar points to the new array while temp points to NULL. The addPharmacy function is called ‘n’ times, the number entered by the user.

addPharmacy()

* This function asks the user to input the name of the pharmacy, and it sets both the name and ID next. The ID is reliant on p, which it then adds every time the function is called.

removePharmacy(int &size)

* This function will check the number of pharmacies and return if there are none, if not, it will ask the user to input a pharmacy id to remove.
* If phar is not pointing to NULL, the id entered is checked against the ids in the array and if found, the index will be put in an integer called index.
* If the index is still at -1 after checking the array, the user is notified that the id was not found, else, p and the size are both decremented by 1.
* A temp is set to point to a new dynamic pharmacy array of the new size, and the elements are copied from phar to the temp from element 0 to index. Then from element index to the end, temp[i] will contain phar[i+1].
* After copying, the phar array is deleted, phar points to the new array and temp points to NULL. The user is notified that the operation was successful.

resizePhar(int &size)

* This function calls the function printMenu4, and asks the user to choose where to add a pharmacy or remove one.

Edit Pharmacies

editPharmacy()

* This function asks the user to input an id, and searches the pharmacy array for that id.
* The user is then given a choice to add or remove a medication or customer.

addMed(int id)

* The function calls medType() in order to specify the medication type to add.
* Then the correct add member function is chosen according to the user’s response.

addShelf(int id)

* Asks the user to input the number of offtheshelf medication they wish to add and calls the resizeShelf() member function.

addPres(int id)

* Asks the user to input the number of prescription medication they wish to add and calls the resizePres() member function.

addCust(int id)

* Asks the user to input the number of customers they wish to add and calls the resizeCust() member function.

removeMed(int id)

* The function calls medType() in order to specify the medication type to remove.
* If there are no medications in the pharmacy, the user is notified of the inability to complete the operation.
* If not, the user is asked to input the id of the medication they wish to remove.

removeShelf(int id)

* This function checks if there are any offtheshelf medications in the pharmacy, if not, the user is notified of the inability to complete the operation.
* Else, the user is asked to input the id of the offtheshelf medication they wish to remove.

removePres(int id)

* This function checks if there are any prescription medications in the pharmacy, if not, the user is notified of the inability to complete the operation.
* Else, the user is asked to input the id of the prescription medication they wish to remove.

removeCust(int id)

* This function checks if there are any customers in the pharmacy, if not, the user is notified of the inability to complete the operation.
* Else, the user is asked to input the id of the customers they wish to remove.

Displays

displayPharmacies()

* This function prints the information of all the pharmacies in the phar array.

pharmDetails(int &size)

* This function asks the user to input an id to find, and if found stores its index in a variable.
* If the id is not found, the user is prompted to enter a new id to search for.
* The user is then asked to chose whether to display the medications or the customers.

displayMeds(int id)

* This function calls medType() to in order to specify the type of medication to print.
* The medications of that type are all printed afterwards.

displayCusts(int id)

* This function prints all the customers in the pharmacy.

int main()

* The main declares three integers, choice, end and size, initializing end to 1 and size to 0.
* What follows is the while loop that controls the program, which first asks the user to input the number of an operation and uses printMenu1() to show the available operations.
* It uses that choice to call a function and perform the operation.
* Within this loop, there is another while loop that checks the choice input by the user to ensure it is only a number from 1 to 5.
* If the choice is valid, it is then checked by an if statement to determine the operation needed.
* The outer while loop depends on the end variable, so at the end of the loop the user is asked to input 0 to exit and 1 to continue, the value will be stored in end, and controls the while loop.