CECS 275 Spring 2022 Project:

Cash Register

By

Dylan Dang (026158052)

&

Dong Woo Shin (026715414)

M/W Lecture 5:00 PM, LAB 6:00 PM

May 5, 2022

# Introduction:

A. Design a program in Java that outputs a receipt of purchase at a market or a convenience store.

- Each receipt must contain the following information (**not an exhaustive list please see a sample receipt below for more information**):

1. The market's name, address, phone number, and fax
2. The date and time of purchase
3. The method of payment:
   a. Card – card type (e.g. visa, master, ), display card number (e.g. XXXXXXXXXXXX1234), entry method (e.g. slides or chip), and whether the card is approved or not (e.g. APPROVED or DENIED).
   b. Cash – cash amount
4. List of items purchased which includes the item's name, quantity, and total
5. The subtotal (amount before tax)
6. Tax percent and amount
7. The balance due (total amount including tax)
8. The amount of change
9. The total number of items
10. The barcode of the receipt

B. Other Requirements:

-Must include at least 5 classes. (Suggestion: CashRegister, CreditCard, Inventory, Barcode, Address)

-The receipt must be formatted nicely.

-DO NOT randomly put items on the receipt by using cout only (this will result in a zero for the project).

-The inventory must be updated accordingly with the item scanned. For example, if there are 10 bottles of water and a customer buys one, the inventory should be updated to 9 bottles of water since another customer might try to find the exact item.

C. Sample run of this program (This is a just an example. Feel free to change the prompt properly according to your own machine):

```
Please scan your item (Press F to finish): 123456
Please scan your item (Press F to finish): 456783
Please scan your items (Press F to finish): 1234567
Would you like to pay with cash or card? card
Please swipe or slide in your card: 1234567891234
Receipt Printing…
(make sure you show a receipt after)

Please scan your item (Press F to finish): 123456
Please scan your item (Press F to finish): 456783
Please scan your items (Press F to finish): 1234567
Would you like to pay with cash or card? cash
Please insert cash: 20
Please insert cash: 40
(… until it's enough or over the amount to be paid)
Receipt Printing…
(make sure you show a receipt after)
```

For this project, my partner and I were tasked to create a cash register program in C++. The cash register should be able to receive the items the customer wants to purchase, ask and receive payment via cash or credit, manage the inventory of items, and print out a receipt of the items purchased. We also had to make sure that the customers were also paying the correct amount and also make sure that customers weren't buying something out of stock.

To do this, we had to create 5 classes: CashRegister, Inventory, Basket, Address, and Barcode. The CashRegister class controls the price, tax rate, sales tax, total cost, etc. The Inventory class sets the cost of each item, and manages the quantity of each item. The Basket class manages the item quantity inside the basket and also keeps track of the cost of items inside the basket. The Address class is in charge of getting the store/convenience store's name, street, state, zip code, and phone number and then outputting that information onto the receipt. The Barcode class was used to create the barcode that is printed on the receipt.

## Program Analysis and Algorithm Design:

– Describe any variables involved in program:

For the CashRegister class, we used variables such as quantity, cost, taxRate, unitPrice, salesTax, taxTotal, subTotal, and total to keep track of all prices. This was very important because without them, it would be very difficult for the customer to know how much they need to pay.

For the Basket class, we used variable numItems to keep track of the number of items the customer has placed in their basket. With the tracked quantity, we are able to accurately calculate the price.

For the Address class, we used variables store_name, street, state, zipcode, and phone_number to hold information about our store. We want to make sure that these variables are private to ensure no tampering could ever happen. The information of the store is the most important thing as it appears at the top of our receipt for every and all purchases.

– Describe any functions used in the program:

The Address class holds the name of the store, address and phone numbers. The main function of this class is the toSting function which prints the store information on the receipt.

From the CashRegister class, we have isQuantity, updateRegister, updateUnits, and displayReceipt. isQuantity makes sure that the customer can only buy items that are in stock. The updateRegister function keeps a running total of taxes and prices while items are added to the basket. The updateUnits function makes sure to subtract the amount of items the customer has taken out from the total quantity. In displayReceipts it prints the bottom of the receipt which displays the sales tax, total tax, subtotal, and total price.

Inventory class holds all the Item's Name, prices and quantity. This class is connected with the basket class because when the user puts the items in their basket, the CashRegister updates the quantity of inventory.

The Barcode class is an independent class. The printBarcode function prints a barcode using unicode. To use unicode in our receipt, we need to include io.h and allow the use of unicode(in line 33, from Barcode.h). Every iteration of the barcode is never the same as we used the rand function to produce a random number. After the number is generated, we mod it by 10 and insert it into the switch case. The barcode will be printed by segments and which differs depending on the case.

# Program Code:

[main.cpp]

```cpp
/*
 * This C++ program's purpose is to:
 * - output a receipt of purchase at a marcket or a convenience store
 * - create cash register that includes at least 5 different classes
 * - manage an inventory system
 * CECS 275 - Spring 2022
 * @author Dylan Dang
 * @author Dongwoo Shin
 * @version 1.3.0
 */

#include <iomanip>
#include <iostream>
#include <string>
#include <array>

#include "Basket.h"
#include "CashRegister.h"
#include "Inventory.h"
#include "Barcode.h"
#include "Address.h"

// global variable
const int NUM_ITEMS = 6;

// function prototypes
void displayItems(std::array<Inventory, NUM_ITEMS>);
void shoppingBasket(std::array<Inventory, NUM_ITEMS> &, std::array <Basket, NUM_ITEMS> &, CashRegister &);
void displayBasket(const std::array <Basket, NUM_ITEMS>);

int main()
{
    CashRegister sales;
    Address addy("Walmart", "8885 N Florida Ave", "Tampla FL", "33604", "(983) 932-0562");
    Barcode bc;

    // creating inventory item objects
    std::array<Inventory, NUM_ITEMS> items { Inventory("Apples", 0.95, 10),  // name, price, quantity
                                             Inventory("Marker", 1.75, 15),
                                             Inventory("Drills", 20.99, 10),
                                             Inventory("Shirts", 7.95, 20),
                                             Inventory("Shampoo", 24.97, 15),
                                             Inventory("Pencils", 2.50, 25) };

    // basket array to hold items
    std::array <Basket, NUM_ITEMS> content{ };

    // keeps track of customer response
    char yes_no = ' ';
    string cash_or_credit = " ";

    // holds customer credit card info
    string credit_number = " ";

    // keeps track of cash needed to pay for item
    double cash = 0;
    double cash_count = 0;

    std::cout << "\tWELCOME TO WALMART\n\n";

```

```
    do {
        shoppingBasket(items, content, sales);

        std::cout << "\nDo you wish to buy another item? (y/n) ";
        std::cin >> yes_no;
        std::cout << "\n";

        // makes sure customer answers with a valid responses
        while (toupper(yes_no) != 'Y' && toupper(yes_no) != 'N') {
            std::cout << "\nDo you wish to buy another item? (y/n) ";
            std::cin >> yes_no;
        }

        // if customer wants to purchase another item
        if (toupper(yes_no) == 'N')
        {
            // asks if customer wants to pay with cash or credit
            std::cout << "Would you like to pay with cash or credit? " << endl;
            std::cin >> cash_or_credit;

            // makes sure customer answers with a valid response
            while (cash_or_credit != "cash" && cash_or_credit != "credit") {
                std::cout << "Invalid payment option. " << endl << "cash or credit? ";
                std::cin >> cash_or_credit;
            }

            // if customer wants to pay with cash
            if (cash_or_credit == "cash") {

                // show customer the total price including tax to pay
                std::cout << "\nTotal Due (including sales tax): $" << sales.recTotal() << endl << endl;
                std::cout << "Please insert cash. " << endl;
                std::cin >> cash;
                cash_count += cash;

                // makes sure the customer pays the correct amount
                while (cash_count < (sales.recTotal()-0.01)) {
                    std::cout << "Please insert more cash. " << endl;
                    std::cin >> cash;
                    cash_count += cash;
                }
                std::cout << "PRINTING RECEIPT..." << endl;

            // if customer wants to pay with credit
            } else if (cash_or_credit == "credit") {
                std::cout << "\nTotal Due (including sales tax): $" << sales.recTotal() << endl << endl;
                std::cout << "Please enter credit card number. " << endl;
                std::cin >> credit_number;

                // makes sure the customer inputs valid credit card number, output denied
                while (credit_number.length() != 16) {
                    std::cout << "CARD DENIED! INVALID CREDIT CARD NUMBER! " << "MUST BE 16 DIGITS." << endl;
                    std::cin >> credit_number;
                }
                // output approved if card number is valid
                if (credit_number.length() == 16) {
                    std::cout << "CARD APPROVED!\n" << "PRINTING RECEIPT..." << endl;
                }
            }
```

```cpp
            // printing store address, items purchased, receipt, and barcode
            std::cout << endl;
            addy.toString();
            displayBasket(content);
            sales.displayReceipt();
            std::cout << endl;
            bc.printBarcode();


        }
    } while (toupper(yes_no) != 'N');

    std::cin.get();
    std::cin.ignore();

    return 0;
}

/**
 * Displays the item names, units on hand, and item cost
 * @param pointer to an Inventory object
 */
void displayItems(std::array<Inventory, NUM_ITEMS> item) {
    std::cout << std::fixed << std::showpoint << std::setprecision(2);
    std::cout << "Item ID\t\t" << "Description\t\t" << "Inventory\t" << "Cost\n\n";

    // iterate through items in array and display them
    for (int i = 0; i < NUM_ITEMS; i++) {
        if (item[i].getUnits() > 0) {
            std::cout << (i + 1) << "\t\t"
                      << item[i].getDescription() << "\t\t\t"
                      << item[i].getUnits() << "\t\t"
                      << item[i].getCost() << "\n";
        } else {
            // once customer bought up the entire stock
            // output "Out of stock" instead of 0
            std::cout << (i + 1) << "\t\t"
                      << item[i].getDescription() << "\t\t\t"
                      << "Out of stock\t"
                      << item[i].getCost() << "\n";
        }
    }
}

```

```cpp
/**
 * Asks the customer to enter the item ID and quantity he/she wishes to purchase
 * This information is processed by functions of the sales and basket classes
 * @param reference to an array of item objects
 * @param reference to an array of basket objects
 * @param reference to a sales object
 */
void shoppingBasket(std::array<Inventory, NUM_ITEMS> &item, std::array <Basket, NUM_ITEMS> &basket, CashRegister &sales) {
    int iQty = 0;
    int iID = 0;

    displayItems(item);

    std::cout << "\nWhich item do you wish to buy? (1 - 6) ";
    std::cin >> iID;

    while (iID <= 0 || iID > NUM_ITEMS) {
        std::cout << "\nWhich item do you wish to buy? (1 through " << NUM_ITEMS << ") ";
        std::cin >> iID;
    }

    // gets the amount the customer wants to purchase and subtract from total quantity in inventory
    std::cout << "How many items do you wish to buy? ";
    std::cin >> iQty;

    while (sales.isQuantity(item[iID - 1], iQty) == false) {
        std::cout << "How many items do you wish to buy? ";
        std::cin >> iQty;
    }

    // update units on hand and prices after customer takes items from inventory
    sales.setCost(item[iID-1]);
    sales.updateUnits(item[iID - 1]);
    sales.updateRegister();

    // update basket with new items
    basket[iID - 1].setItemInfo(item[iID-1]);
    basket[iID - 1].setNumItems(sales.getQuantity());
}

/**
 * Outputs to the screen the item name, quantity and cost of tiems currently in the basket
 * @param array of basket objects
 */
void displayBasket(const std::array <Basket, NUM_ITEMS> content) {

    // formatting the output of what the customer has in their basket
    std::cout << std::fixed << std::showpoint << std::setprecision(2);

    for (int i = 0; i < NUM_ITEMS; i++) {
        if (content[i].getQuantity() > 0) {
            std::cout << "ITEM NAME: " << std::setw(28) << content[i].getDescription() << "\n"
                      << "QUANTITY:  " << std::setw(28) << content[i].getQuantity() << "\n"
                      << "COST: "      << std::setw(26)
                      << "$"           << std::setw(7) << (content[i].getCost() * content[i].getQuantity()) << "\n\n";
        }
    }
    std::cout << "-------------------------------------\n\n";
}
```

[CashRegister.h]

```cpp
/*
 * CashRegister class header & implementation
 * CECS 275 - Spring 2022
 * @author Dylan Dang
 * @author Dongwoo Shin
 * @version 1.0.5
 */

#ifndef CASH_REGISTER_H
#define CASH_REGISTER_H

#include <iomanip>
#include <iostream>
#include <string>

#include "CashRegister.h"
#include "Inventory.h"

class CashRegister {
    private:
        int    quantity;            // Item quantity to be bought
        double cost;                // Cost of the item
        double taxRate;             // Tax rate (6%)
        double unitPrice;           // The unit price of an item
        double salesTax;            // The sales tax (6%)
        double taxTotal;            // The tax total (sales tax)
        double subTotal;            // The subtotal (cost)
        double total;               // The grand-total

    public:
        /**
         * Constructs an inventory with tax rate
         * @param rate the tax rate
         */
        CashRegister(double rate = 0.06) {
            taxRate = rate;
            unitPrice = 0.0;
            salesTax = 0.0;
            subTotal = 0.0;
            taxTotal = 0.0;
            total = 0.0;
        }

        // Accessors
        void setCost(const Inventory item);
        bool isQuantity(Inventory item, int iQty);
        void updateRegister();
        void updateUnits(Inventory &item);
        void displayReceipt();

        /**
         *  Gets the item cost
         *  @return the item cost
         */
        double getCost() const {
            return cost;
        }
        /**
         *  Gets the item unit price
         *  @return getCost() the item cost
         */
        double getUnitPrice() const {
            return (getCost());
        }
```

```
65          /**
66           *  Gets the sale tax
67           *  @return (getCost() * taxRate) the sale tax
68           */
69          double getSalesTax() const {
70              return (getCost() * taxRate);
71          }
72          /**
73           *  Gets the total tax
74           *  @return getSalesTax() the total tax
75           */
76          double getTaxTotal() const {
77              return (getSalesTax());
78          }
79          /**
80           *  Gets the sub total
81           *  @return getCost() the sub total
82           */
83          double getSubTotal() const {
84              return (getCost());
85          }
86          /**
87           *  Gets the total cost
88           *  @return (getSubToal() * getSalesTax) the total cost
89           */
90          double getTotal() const {
91              return (getSubTotal() + getSalesTax());
92          }
93          /**
94           *  Gets the item quantity
95           *  @return quantity the item quantity
96           */
97          int getQuantity() const {
98              return quantity;
99          }
100         /**
101          *  Gets the item unit price for receipt
102          *  @return unitPrice the item unit price
103          */
104         double recUnitPrice() const {
105             return unitPrice;
106         }
107         /**
108          *  Gets the sales tax for receipt
109          *  @return salesTax the sales tax
110          */
111         double recSalesTax() const {
112             return salesTax;
113         }
114         /**
115          *  Gets the total tax for receipt
116          *  @return taxTotal the total tax
117          */
118         double recTaxTotal() const {
119             return taxTotal;
120         }
121         /**
122          *  Gets the subtotal for receipt
123          *  @return subTotal the subtotal
124          */
125         double recSubTotal() const {
126             return subTotal;
127         }
128         /**
129          *  Gets the total cost for receipt
130          *  @return total the total cost
131          */
132         double recTotal() const {
133             return total;
134         }
135     };
```

```cpp

/**
 * Determines whether the value passed to it is valid
 * If it is valid, the value is assigned to the quantity member and returns true
 * If number of available units is 0, quantity is assigned 0, output out of stock message, returns true
 * If value entered is greater than quantity available, ouput item and quantity available, retuns false
 * @param Inventory object the item
 * @param integer value the amount the customer wants to purchase
 * @return true if value is valid, false if value is invalid
 */
bool CashRegister::isQuantity(Inventory item, int iQty) {
    if (item.getUnits() > 0 && iQty <= item.getUnits()) { // make sure value inputted is valid
        quantity = iQty; // sets quantity to inputted value
        return true;
    } else if (item.getUnits() == 0) {
        quantity = 0;
        std::cout << "\n" << item.getDescription() << " Out Of Stock\n"; // output out of stock if quantity is 0
        return true;
    } else {
        std::cout << "\nItem: " << item.getDescription() << "\n"; // if invalid output name and quantity
        std::cout << "Availabe quantity: " << item.getUnits() << "\n";
        return false;
    }
    return false;
}
/**
 * Sets the cost for an item
 * The value to be assigned to the cost is passed to it by the item object
 * @param Inventory object the item
 */
void CashRegister::setCost(const Inventory item) {
    cost = item.getCost();
}
/**
 * Sets the number of units on hand stored in the item object
 * The new value is calculated by subtracting the quantity of items
 * bought from the units on hand stored in the item object
 * @param reference to an Inventory object the item
 */
void CashRegister::updateUnits(Inventory &item) {
    item.setUnits(item.getUnits() - getQuantity());
}
```

```cpp
/**
 * Keeps a running total of taxes and prices while items are added to the basket
 * All values are multiplied by the quantity being purchased
 */
void CashRegister::updateRegister() {
    unitPrice += getUnitPrice() * getQuantity();
    salesTax  += getSalesTax()  * getQuantity();
    subTotal  += getSubTotal()  * getQuantity();
    taxTotal  += getTaxTotal()  * getQuantity();
    total     += getTotal()     * getQuantity();
}
/**
 * Outputs to screen with the unit price, sales tax, total tax,
 * subtotal, total cost
 */
void CashRegister::displayReceipt() { // formatting prices for receipt
    std::cout << std::fixed << std::showpoint << std::setprecision(2);

    std::cout << std::left << "Sales-Tax (6%): " << std::setw(18)
              << std::right << "$ " << std::setw(6)
              << recSalesTax() << "\n";

    std::cout << std::left << "Tax-Total: " << std::setw(23)
              << std::right << "$ " << std::setw(6)
              << recTaxTotal() << "\n\n";

    std::cout << std::left << "Sub-Total: " << std::setw(23)
              << std::right << "$ " << std::setw(6)
              << recSubTotal() << "\n";

    std::cout << std::left << "Purchase Price: " << std::setw(18)
              << std::right << "$ " << std::setw(6)
              << recTotal() << "\n";

    std::cout << "\n\t      THANK YOU FOR  \n"
              << "\t  SHOPPING AT WALMART!" << endl;
}
#endif
```

[Inventory.h]

```cpp
/*
 * Inventory class header & implementation
 * CECS 275 - Spring 2022
 * @author Dylan Dang
 * @author Dongwoo Shin
 * @version 1.2.0
 */

#ifndef INVENTORY_H
#define INVENTORY_H

#include <string>
using namespace std;

class Inventory {
    private:
        string description;        // The item name
        double    cost;            // Cost of an item
        int       units;           // Number of units on hand

    public:
        // Default constructor
        Inventory() {
            description = " ";
            cost = 0.0;
            units = 0;
        }
        /**
         * Constructs an inventory with item description, cost, units
         * @param desc the item description
         * @param c the item cost
         * @param u the units at hand
         */
        Inventory(string desc, double c, int u) {
            description = desc;
            cost = c;
            units = u;
        }
        /**
         * Sets the description of item
         * @param desc the description
         */
        void setDescription(string desc) {
            description = desc;
        }
        /**
         * Sets the cost of item
         * @param c the cost
         */
        void setCost(double c) {
            cost = c;
        }
        /**
         * Sets the units on hand
```

```
     * Sets the units on hand
     * @param u the units
     */
    void setUnits(int u) {
        units = u;
    }
    /**
     * Gets the description of item
     * @return description the item description
     */
    string getDescription() const {
        return description;
    }
    /**
     * Gets the cost of item
     * @return cost the item cost
     */
    double getCost() const {
        return cost;
    }
    /**
     * Gets the units on hand
     * @return units the units on hand
     */
    int getUnits() {
        return units;
    }
};
#endif
```

[Barcode.h]

```cpp
/*
 * Barcode class header & implementation
 * CECS 275 - Spring 2022
 * @author Dylan Dang
 * @author Dongwoo Shin
 * @version 1.1.0
 */

#ifndef BARCODE_H
#define BARCODE_H

#include <iostream>
#include <io.h>
#include <stdlib.h>     /* srand, rand */
#include <time.h>       /* time */

using namespace std;

class Barcode {
private:
    int returnval, str2int;
    wchar_t* c;
    wchar_t strU[100] = L"";

public:
    Barcode() {}
    /**
     * Prints the barcode using unicode
     * Unicode pattern is random
     */
    void printBarcode() {
        // allow the use of Unicode
        _setmode(_fileno(stdout), 0x00020000);

        // make use of the computer's internal clock to generate random values
        srand (time(NULL));

        for (unsigned int i = 0; i < 12; i++) {
            // get random number
            str2int = rand() % 10 ;

            // output different unicode values up to 12 times depending on random value
            switch (str2int) {
            case 0:
            c = wcscat(strU,L"\x2502\x2588\x2502");
            break;
            case 1:
            c = wcscat(strU,L"\x2588\x2502\x2588 ");
            break;
            case 2:
            c = wcscat(strU,L"\x2502\x2588\x2502");
            break;
            case 3:
            c = wcscat(strU,L"\x2588\x2588\x2502");
```

```
54                  c = wcscat(strU,L"\x2588\x2588\x2502");
55                  break;
56                  case 4:
57                  c = wcscat(strU,L"\x2502\x2588 ");
58                  break;
59                  case 5:
60                  c = wcscat(strU,L"\x2588 \x2502");
61                  break;
62                  case 6:
63                  c = wcscat(strU,L"\x2502\x2502\x2502");
64                  break;
65                  case 7:
66                  c = wcscat(strU,L"\x2502 \x2588");
67                  break;
68                  case 8:
69                  c = wcscat(strU,L"\x2502 \x2502\x2588 ");
70                  break;
71                  case 9:
72                  c = wcscat(strU,L"\x2588\x2502\x2588 ");
73                  break;
74                  }
75              }
76
77              wcout << c <<endl;
78              wcout << c <<endl;
79              wcout << c <<endl;
80          }
81      };
82      #endif
```

[Basket.h]

```cpp
/*
 * ShoppingBasket class header & implementation
 * CECS 275 - Spring 2022
 * @author Dylan Dang
 * @author Dongwoo Shin
 * @version 1.0.2
 */

#ifndef SHOPPING_BASKET_H
#define SHOPPING_BASKET_H

#include <string>

#include "Inventory.h"

class Basket {
    private:
        std::string description;            // item description
        int numItems;                       // number of items in the basket
        double cost;                        // cost of items in the basket

    public:
        // Default constructor
        Basket() {
            description = " ";
            numItems = 0;
            cost = 0.0;
        }
        /**
         * Constructs a basket with an item quantity.
         * @param iQty the item quantity
         */
        Basket(int iQty) {
            numItems = iQty;
        }
        /**
         * Sets values to description and cost
         * @param Inventory object the item
         */
        void setItemInfo(const Inventory item) {
            description  = item.getDescription();
            cost         = item.getCost();
        }
        /**
         * Sets number of items
         * @param iQty the item quanitity
         */
        void setNumItems(int iQty) {
            numItems += iQty;
        }
        /**
         * Gets the item quantity
         * @return numItems the number of items
         */
```

```
        int getQuantity() const {
            return numItems;
        }
        /**
         * Gets the item description
         * @return description the item description
         */
        std::string getDescription() const {
            return description;
        }
        /**
         * Gets the item cost
         * @return cost the item cost
         */
        double getCost() const {
            return cost;
        }
    };
    #endif
```

[Address.h]

```cpp
/*
 * Address class header & implementation
 * CECS 275 - Spring 2022
 * @author Dylan Dang
 * @author Dongwoo Shin
 * @version 1.1.0
 */

#ifndef ADDRESS_H
#define ADDRESS_H

#include <iostream>
#include <sstream>
#include <iomanip>

using namespace std;

class Address {
    private:
        string store_name;      // the store's name
        string street;          // the street where the store is located
        string state;           // the state where the store is located
        string zipcode;         // the zipcode where the store is located
        string phone_number;    // the store's phone number

    public:
        // Default constructor
        Address() {
            store_name = " ";
            street = " ";
            state = " ";
            zipcode = " ";
            phone_number = " ";
        }
        /**
         * Constructs an address with a store name, street, state, zipcode, phone number.
         *
         * @param store_name the store name
         * @param street the street
         * @param state the last state
         * @param zipcode the annual zipcode
         * @param phone_number the phone_number
         */
        Address(const string &store_name, const string &street, const string &state, const string &zipcode, const string &phone_number)
                : store_name(store_name), street(street), state(state), zipcode(zipcode), phone_number(phone_number) {}

        /**
         *  Gets the store name
         *  @return store_name the store name
         */
        const string &getStoreName() const {
            return store_name;
        }
        /**
         *  Sets the store name
```

```
 *  @param sn the store name
 */
void setStoreName(const string &sn) {
    store_name = sn;
}
/**
 *  Gets the street
 *  @return street the street
 */
const string &getStreetName() const {
    return street;
}
/**
 *  Sets the street
 *  @param s the street
 */
void setStreetName(const string &s) {
    street = s;
}
/**
 *  Gets the state
 *  @return state the state
 */
const string &getStateName() const {
    return state;
}
/**
 *  Sets the state
 *  @param st the state
 */
void setStateName(const string &st) {
    state = st;
}
/**
 *  Gets the zipcode
 *  @return zipcode the zipcode
 */
const string &getZipcode() const {
    return zipcode;
}
/**
 *  Sets the zipcode
 *  @param z the zipcode
 */
void setZipcode(const string &z) {
    zipcode = z;
}
```

```
        /**
         *  Gets the phone number
         *  @return phone_number the phone number
         */
        const string &getPhoneNumber() const {
            return phone_number;
        }
        /**
         *  Sets the phone number
         *  @return pn the phone number
         */
        void setPhoneNumber(const string &pn) {
            phone_number = pn;
        }
        /**
         * Prints the store name, street, state, zipcode, phone number
         */
        void toString() {
            ostringstream os;
            os << setw(23) << store_name << endl
            << setw(26) << phone_number << endl
            << setw(20) << street
            << ", " << state
            << ", " << zipcode << endl
            << "--------------------------------------\n" << endl;
            cout << os.str();
        }
};
#endif
```

## Sample Run:

[Start]

```
                    WELCOME TO WALMART

Item ID         Description             Inventory       Cost

1               Apples                  10              0.95
2               Marker                  15              1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6)
```

[Out of Stock]

```
                    WELCOME TO WALMART

Item ID         Description             Inventory       Cost

1               Apples                  10              0.95
2               Marker                  15              1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6) 1
How many items do you wish to buy? 10

Do you wish to buy another item? (y/n) y

Item ID         Description             Inventory       Cost

1               Apples                  Out of stock    0.95
2               Marker                  15              1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6)
```

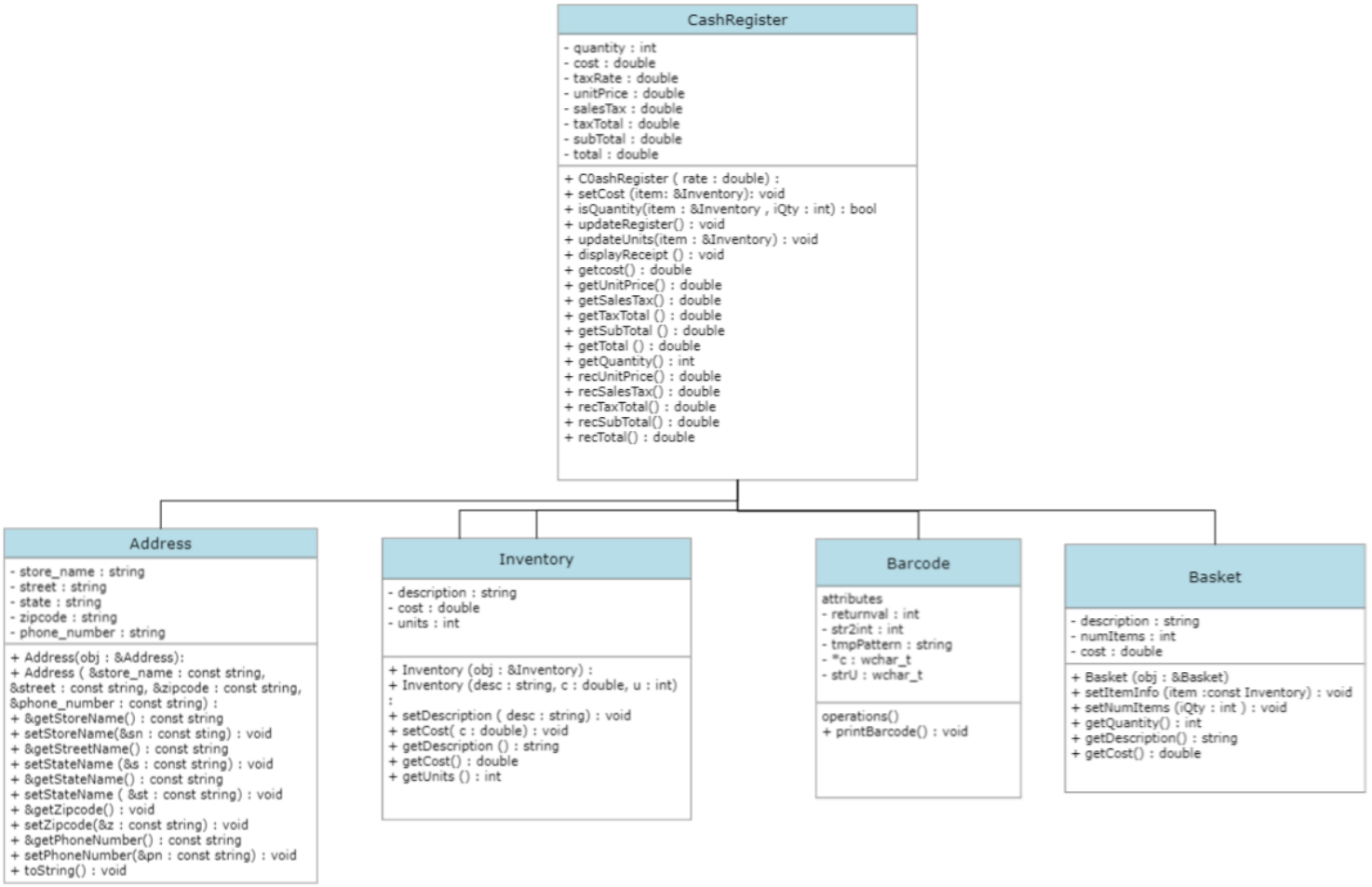
[Pay with cash + insert cash until item(s) is paid]

```
                    WELCOME TO WALMART

Item ID         Description             Inventory       Cost

1               Apples                  10              0.95
2               Marker                  15              1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6) 1
How many items do you wish to buy? 10

Do you wish to buy another item? (y/n) y

Item ID         Description             Inventory       Cost

1               Apples                  Out of stock    0.95
2               Marker                  15              1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6) 5
How many items do you wish to buy? 5

Do you wish to buy another item? (y/n) n

Would you like to pay with cash or credit?
cash

Total Due (including sales tax): $142.41

Please insert cash.
100
Please insert more cash.
40
Please insert more cash.
2.41
PRINTING RECEIPT...

                Walmart
            (983) 932-0562
  8885 N Florida Ave, Tampla FL, 33604
--------------------------------------

ITEM NAME:                      Apples
QUANTITY:                           10
COST:                         $   9.50

ITEM NAME:                     Shampoo
QUANTITY:                            5
COST:                         $ 124.85

--------------------------------------

Sales-Tax (6%):               $   8.06
Tax-Total:                    $   8.06

Sub-Total:                    $ 134.35
Purchase Price:               $ 142.41

              THANK YOU FOR
          SHOPPING AT WALMART!
```

[Pay with credit + invalid credit card]

```
                    WELCOME TO WALMART

Item ID         Description             Inventory       Cost

1               Apples                  10              0.95
2               Marker                  15              1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6) 2
How many items do you wish to buy? 7

Do you wish to buy another item? (y/n) y

Item ID         Description             Inventory       Cost

1               Apples                  10              0.95
2               Marker                  8               1.75
3               Drills                  10              20.99
4               Shirts                  20              7.95
5               Shampoo                 15              24.97
6               Pencils                 25              2.50

Which item do you wish to buy? (1 - 6) 6
How many items do you wish to buy? 3

Do you wish to buy another item? (y/n) n

Would you like to pay with cash or credit?
credit

Total Due (including sales tax): $20.93

Please enter credit card number.
128492
CARD DENIED! INVALID CREDIT CARD NUMBER! MUST BE 16 DIGITS.
4820129359212895
CARD APPROVED!
PRINTING RECEIPT...

                Walmart
             (983) 932-0562
  8885 N Florida Ave, Tampla FL, 33604
--------------------------------------

ITEM NAME:                      Marker
QUANTITY:                            7
COST:                         $  12.25

ITEM NAME:                     Pencils
QUANTITY:                            3
COST:                         $   7.50

--------------------------------------

Sales-Tax (6%):               $   1.19
Tax-Total:                    $   1.19

Sub-Total:                    $  19.75
Purchase Price:               $  20.93

              THANK YOU FOR
           SHOPPING AT WALMART!
```

# UML Diagram:


CashRegister
- quantity : int
- cost : double
- taxRate : double
- unitPrice : double
- salesTax : double
- taxTotal : double
- subTotal : double
- total : double
+ C0ashRegister ( rate : double) :
+ setCost (item: &Inventory): void
+ isQuantity(item : &Inventory , iQty : int) : bool
+ updateRegister() : void
+ updateUnits(item : &Inventory) : void
+ displayReceipt () : void
+ getcost() : double
+ getUnitPrice() : double
+ getSalesTax() : double
+ getTaxTotal () : double
+ getSubTotal () : double
+ getTotal () : double
+ getQuantity() : int
+ recUnitPrice() : double
+ recSalesTax() : double
+ recTaxTotal() : double
+ recSubTotal() : double
+ recTotal() : double
Address
- store_name : string
- street : string
- state : string
- zipcode : string
- phone_number : string
+ Address(obj : &Address):
+ Address ( &store_name : const string, &street : const string, &zipcode : const string, &phone_number : const string) :
+ &getStoreName() : const string
+ setStoreName(&sn : const sting) : void
+ &getStreetName() : const string
+ setStateName (&s : const string) : void
+ &getStateName() : const string
+ setStateName ( &st : const string) : void
+ &getZipcode() : void
+ setZipcode(&z : const string) : void
+ &getPhoneNumber() : const string
+ setPhoneNumber(&pn : const string) : void
+ toString() : void
Inventory
- description : string
- cost : double
- units : int
+ Inventory (obj : &Inventory) :
+ Inventory (desc : string, c : double, u : int) :
+ setDescription ( desc : string) : void
+ setCost( c : double) : void
+ getDescription () : string
+ getCost() : double
+ getUnits () : int
Barcode
attributes
- returnval : int
- str2int : int
- tmpPattern : string
- *c : wchar_t
- strU : wchar_t
operations()
+ printBarcode() : void
Basket
- description : string
- numItems : int
- cost : double
+ Basket (obj : &Basket)
+ setItemInfo (item :const Inventory) : void
+ setNumItems (iQty : int ) : void
+ getQuantity() : int
+ getDescription() : string
+ getCost() : double