// 3.9: Modified Account Class

```
//HEADER FUNCTION
#ifndef ACCOUNT_H_INCLUDED
#define ACCOUNT_H_INCLUDED

#include <string>

class Account {
public:
   //Account constructor with two parameters
   Account(std::string accountName, int initialBalance)
      :name{accountName} {//assign accountName to data member name

      //validate that the initialBalnxe is greater than 0; if not,
      //data member balance keeps its default initial value of 0
      if(initialBalance > 0) {
         balance = initialBalance;
      }
   }

   //function that deposits (adds) only a valid amount to the balance
   void deposit(int depositAmount) { //if deposit amount is valid
      if (depositAmount > 0) {
         balance = balance + depositAmount;
      }
   }

   //function that returns the account balance
   int getBalance() const {
      return balance;
   }

   //function that sets the name
   void setName(std::string accountName) {
      name = accountName;
   }

   //function that returns the name
   std::string getName() const {
      return name;
   }
```

```
   //function that withdraws (subtracts) only a valid amount from the balance
   void withdraw(int withdrawalAmount){
      if (withdrawalAmount <= balance) {
         balance = balance - withdrawalAmount;
      } else if (withdrawalAmount > balance) {
         std::cout << " exceeded account balance: withdrawal void. Please try again.";
      }
   }

private:
   std::string name; //account name data member
   int balance{0}; // data member with default initial value
}; //end class account

#endif // ACCOUNT_H_INCLUDED

//.CPP FUNCTION


#include <iostream>
#include "Account.h"

using namespace std;

int main()
{
   Account account1{"Jane Green", 50};
   Account account2{"John Blue", -7};

   //display initial balance of each object
   cout << "Account one: " << account1.getName() << " balance is $"
      << account1.getBalance();
   cout << "\nAccount two: " << account2.getName() << " balance is $"
      << account2.getBalance();

   cout << "\n\nEnter deposit amount for Account one: "; //prompt
   int depositAmount;
   cin >> depositAmount; //obtain user input
   cout << "adding " << depositAmount << " to Account one balance";
   account1.deposit(depositAmount); //add to account's balance

   //display new balances
```

```
    cout << "\n\nAccount one: " << account1.getName() << " balance is $"
       << account1.getBalance();
    cout << "\nAccount two: " << account2.getName() << " balance is $"
       << account2.getBalance();

    cout << "\n\nEnter deposit amount for Account two: "; //prompt
    cin >> depositAmount; //obtain user input
    cout << "adding " << depositAmount << " to Account two balance";
    account2.deposit(depositAmount); //add to account's balance

    //display new balances
    cout << "\n\nAccount one: " << account1.getName() << " balance is $"
       << account1.getBalance();
    cout << "\nAccount two: " << account2.getName() << " balance is $"
       << account2.getBalance();

cout << "\n\nEnter withdrawal amount for Account one: "; //prompt
    int withdrawalAmount;
    cin >> withdrawalAmount; //obtain user input
    cout << "taking " << withdrawalAmount << " from Account one balance";
    account1.withdraw(withdrawalAmount); //add to account's balance

    //display new balances
    cout << "\n\nAccount one: " << account1.getName() << " balance is $"
       << account1.getBalance();
    cout << "\nAccount two: " << account2.getName() << " balance is $"
       << account2.getBalance();

    cout << "\n\nEnter withdrawal amount for Account two: "; //prompt
    cin >> withdrawalAmount; //obtain user input
    cout << "taking " << withdrawalAmount << " from Account two balance";
    account2.withdraw(withdrawalAmount); //add to account's balance

    //display new balances
    cout << "\n\nAccount one: " << account1.getName() << " balance is $"
       << account1.getBalance();
    cout << "\nAccount two: " << account2.getName() << " balance is $"
       << account2.getBalance();

}
```

//3.10: Invoice Class

/* Create a class called INVOICE that a hardware store might use to
represent an invoice for an item sold at the store. An INVOICE should
include four data members: a part number (type STRING), a part description
(type STRING), a quantity of the item being purchased (type INT) and a
price per item (type INT). Your class should have a constructor that
initializes the four data members. Provide a SET and GET function for each
data member. In addition, provide a member function named GETINVOICEAMAOUNT
that calculates the invoice amount (i.e., multiples the quantity by the price
per item), the return the amount as an INT value. If the quantity is not
positive, it should be set to 0. if the price per item is not positive,
it should be set to 0. Write a test program that demonstrates class
INVOICE'S capabilities. */

//This is the main.cpp function.

```
#include <iostream>
#include <string>
#include "Invoice.h"

using namespace std;

int main()
{
   //sample items that could be purchased... I do not know a lot about hardware parts haha
   Invoice purchase1{"125469", "Hammer.", 6 }; //First item is a silver hammer bought for $6
each.
   Invoice purchase2{"83958A", "Lightbulb.", 2}; //Second is a 60-watt lightbulb bought for $2
each. */

   cout << "ITEMS PURCHASED:";

   cout << purchase1.getName();
   cout << "Quantity: " ; //Prompt
   int quant;
   cin >> quant >> endl; //obtain user input
   cout << "Price: " << purchase1.getInvoiceAmount << endl;

   cout << purchase2.getName();
   cout << "Quantity: "; //Prompt
   int quant;
```

```
    cin >> quant >> endl; // obtain user input
    cout << "Price: " <<  purchase2.getInvoiceAmount << endl;
}



// This is the invoice.h function.

#ifndef INVOICE_H_INCLUDED
#define INVOICE_H_INCLUDED

#include <string>

class Invoice {
public:
    //Invoice constructor with four parameters
    Invoice(std::string num, std::string desc, int price){

    //Function that sets the name (num and desc)
    void setName() {
        if (num != desc) {
            name = num + desc;
        }
    }

    std::string getName() const {
        return name;
    }

    //Function that returns the price of an item.
    int getPrice() const {
        return price;
    }

    //Function that returns invoice amount.
    void setInvoiceAmount(int price, int quant) {
        invoiceAmount = price * quant;
        if (quant < 0) {
            quant = 0;
        }
        if (price < 0) {
            price = 0;
        }
```

```
    }

    int getInvoiceAmount() const{
        return invoiceAmount;
        }
    }

private:
    int num{0};
    int quant{0}; //initial quantity of an item is 0.
    int price{0}; // initial price of an item is $0.
    int balance{0}; //initial balance of a customer's receipt is 0.
    std::string name;
    int invoiceAmount{0};

/*
            VARIABLES
            num = part number (name)
            desc = description of part
            quant = quantity of item
            price = price of item
            balance = total cost of all products.
            invoiceAmount = price * quant
*/

#endif // INVOICE_H_INCLUDED
};
```

//3.12: Date class

/* Create a class called DATE that includes three pieces of information: a month, a day, and a year.
All INT TYPES. Your class should have a constructor with three parameters that uses the parameters to initialize the three data members.
For the purpose of this exercise, assume that the values provided for the year and the day are correct,
but ensure that the month value is in the range 1 - 12; if it isn't, set the month to 1.
Provide a SET and GET function for each data member.
Provide a member function DISPLAYDATE that displays the month, day, and year separated by forward slashes.

Write a test program the demonstrates DATE'S capabilities. */

//This is the main .cpp file.

```
#include <iostream>
#include <string>
#include "Date.h"

using namespace std;

int main() {
    Date date1{12, 1, 2016};
    Date date2{2, 16, 2017};

    cout << "Date 1 is: " << date1.getDisplayDate << endl;
    cout << "Date 2 is: " << date2.getDisplayDate << endl;
}
```

// This is the .h header file.

```
#ifndef DATE_H_INCLUDED
#define DATE_H_INCLUDED
#include <string>

class Date {

public:
   Date(int month, int day, int year){
   //Function that sets and returns the month
   void setMonth(int month) {
      if (month < 1 || month > 12) {
         month = 1;
      }
   }

   int getMonth() const{
      return month;
   }

   //Function that returns the day
   int getDay() {
      return day;
   }
```

```
    //Function that returns the year
    int getYear() {
        return year;
    }

    //Function that helps concatenate the month, day, and year.
    std::string getSlash() {
        return "/";
    }

    //Section that displays the complete date
    void setDisplayDate(int day, int month, int year) {
        displayDate = std::to_string(month) + getSlash + std::to_string(day) + getSlash +
std::to_string(year);
    }

    int getDisplayDate(){
        return displayDate;
        }
    }
private:
    int month{0};
    int year{0};
    int day{0};
    int displayDate{0};
};

#endif // DATE_H_INCLUDED
```

//3.12

//Create a class called date that displays a date.

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

using namespace std;
```

```
class Date {
public:
    Date(int d, int m, int y)
    : day{d}, month{m}, year{y} {

    if (month > 12) {
        m = month;
        m = 1;
    }
    }

    int getDay() const {
    return day;
    }

    void setDay(int d){
    d = day;
    }

    int getMonth() const{
    return month;
    }

    void setMonth(int m) const{
    m = month;
    }

    int getYear() const{
    return year;
    }

    void setYear(int y) const{
    y = year;
    }

    string displayDate() {
    string calendar{to_string(month) + "/" + to_string(day) + "/" +  to_string(year)};
    return calendar;
}

private:
    int day;
    int month;
```

```
    int year;
};

int main()
{

    Date thisChristmas{25, 12, 2018};
    Date myBirthday{16, 02, 2017};

    //Date thisChristmas.getDay();
    //Date thisChristmas.getMonth();
    //Date thisChristmas.getYear();
    cout << thisChristmas.displayDate() << endl;
    cout << myBirthday.displayDate() << endl;
}
```