Unit testing

Oscar Bedolla

Github url:

```
// CheckoutTest.cpp : This file contains the 'main' function. Program execution begins
and ends there.
//

#include "pch.h"
#include <iostream>
#include <gtest/gtest.h>
#include "Checkout.h"

using namespace std;

class CheckoutTests : public ::testing::Test {
public:

protected:
	Checkout checkOut;
};

TEST_F(CheckoutTests, CanCalculateTotal) {
	checkOut.addItemPrice("a", 1);
	checkOut.addItem("a");
	int total = checkOut.calculateTotal();
	ASSERT_EQ(1, total);
}

TEST_F(CheckoutTests, CanGetTotalForMultipleItems) {
	checkOut.addItemPrice("a", 1);
	checkOut.addItemPrice("b", 2);
	checkOut.addItem("a");
	checkOut.addItem("b");
	int total = checkOut.calculateTotal();
	ASSERT_EQ(3, total);
}

TEST_F(CheckoutTests, CanAddDiscount) {
	checkOut.addDiscount("a", 3, 2);
}

TEST_F(CheckoutTests, CanCalculateTotalWithDiscount) {
	checkOut.addItemPrice("a", 1);
	checkOut.addDiscount("a", 3, 2);
	checkOut.addItem("a");
	checkOut.addItem("a");
	checkOut.addItem("a");
	int total = checkOut.calculateTotal();
	ASSERT_EQ(2, total);
}
```

```
TEST_F(CheckoutTests, itemeWithNoPriceThrowsException) {
	ASSERT_THROW(checkOut.addItem("a"), std::invalid_argument);
}

// Run program: Ctrl + F5 or Debug > Start Without Debugging menu
// Debug program: F5 or Debug > Start Debugging menu
```

```
//checkout.h file
#pragma once

#include <string>

#ifndef CHECKOUT_H_
#define CHECKOUT_H_
#include <string>
#include <map>
class Checkout
{
public:
	Checkout();
	virtual ~Checkout();

	void addItemPrice(std::string item, int price);
	void addItem(std::string item);
	void addDiscount(std::string item, int nbrOfItems, int discountPrice);
	int calculateTotal();

protected:
	struct Discount {
		int nbrOfItems;
		int discountPrice;

	};
	std::map<std::string, int> prices;
	std::map<std::string, Discount> discounts;
	std::map<std::string, int> items;
	int total;

	void calculateItem(std::string item, int itemCnt);
	void calculateDiscount(std::string item, int itemCnt, Discount discount);
};

#endif
```

```
//checkout.cpp code
#include "pch.h"
#include "Checkout.h"


Checkout::Checkout():total(0)
{
```

```
}


Checkout::~Checkout()
{
}

void Checkout::addItemPrice(std::string item, int price ) {
	prices[item]=price;
}

void Checkout::addItem(std::string item) {
	std::map<std::string, int>::iterator priceIter = prices.find(item);
	if (priceIter == prices.end()) {
		throw std::invalid_argument("Invalid item. No price");
	}

	items[item]++;
}
void Checkout::addDiscount(std::string item, int nbrOfItems, int discountPrice) {
	Discount discount;
	discount.nbrOfItems = nbrOfItems;
	discount.discountPrice = discountPrice;
	discounts[item] = discount;
	;
}

int Checkout::calculateTotal(){
	total = 0;

	for(std::map<std::string, int>::iterator itemIter=items.begin();
		itemIter != items.end(); itemIter++){
		std::string item=itemIter->first;
			int itemCnt=itemIter->second;
			calculateItem(item, itemCnt);
	}
return total;
}

void Checkout::calculateItem(std::string item, int itemCnt) {
	std::map<std::string, Discount>::iterator discountIter;
	discountIter = discounts.find(item);
	if (discountIter != discounts.end()) {
		Discount discount = discountIter->second;
		calculateDiscount(item, itemCnt, discount);
	}
	else {
		total += itemCnt * prices[item];
	}
}

void Checkout::calculateDiscount(std::string item, int itemCnt, Discount discount) {
	if (itemCnt >= discount.nbrOfItems) {
		int nbrOfDiscounts = itemCnt / discount.nbrOfItems;
		total += nbrOfDiscounts * discount.discountPrice;
		int remainingItems = itemCnt % discount.nbrOfItems;
		total += remainingItems * prices[item];
	}
```

```
        else {
                total += itemCnt * prices[item];
        }
}
```