**СУ „Св. Климент Охридски“, ФМИ**

Специалност „Софтуерно Инженерство“

**Обектно-ориентирано програмиране, 2020-2021 г.**

**Задача за домашно № 1**

**Срок: 28.03.2021 г. 23:59**

## **Важна информация**

### **Оценяване на домашното**

* Част от точките за това домашно ще бъдат давани след покриването на автоматични тестове - за **коректно реализирана функционалност (30%)** и **правилно управление на динамичната памет (20%)**
* За да получите тези точки, предадените от вас решения трябва да отговарят на следните критерии
  + Да съдържат указаните методи и имена на класове (ще ви бъде даден шаблон, върху който да работите) - позволено е да добавяте нови методи и класове, но **не е позволено** да променяте даденото от нас.
  + Предавайте единствено файлове съдържащи код - архиви съдържащи .sln файлове или каквито и да е други файлове, които не са .cpp или .hpp ще получават 0 точки на автоматичните тестове.
  + **Не предавайте** архиви от тип .rar - **ще се приемат архиви от тип .zip.** При получен архив от тип .rar (или друг тип, които не може да бъде разархивиран от системата за тестване), отново получавате 0 точки на автоматичните тестове.
  + Именувайте архива си по следния начин - SI\_R\_HW1\_<курс>\_<група>\_<факултетен номер>. Архиви, които не спазват тази конвенция ще получат 0 точки на автоматичните тестове. (Пример: SI\_R\_HW1\_1\_1\_12345.zip)
  + Не променяйте имената на файлове, които получавате
  + Спазвайте следната структура на архива:

**SI\_R\_HW1\_<курс>\_<група>\_<факултетен номер>.zip**

├── **1**

│ ├── drink.hpp

│ ├── VendingMachine.hpp

| ├── \*.cpp

| └── \*.hpp

└── **2**

├── Error.hpp

├── sample\_unit\_tests.cpp

├── TestCase.hpp

├── TestSuite.hpp

├── \*.cpp

└── \*.hpp

* Ако решението ви не се компилира, получавате 0 точки на автоматичните тестове
* Останалите **50%** ще бъдат давани след преглеждане на решенията ви от асистент
* Спазвайте практиките за обектно-ориентирано програмиране, коментирани на упражнения и лекции.

## **Задача 1 (4 точки)** Линк към шаблона:[тук](https://github.com/lyubolp/OOP-SI-2021/tree/main/HW1/task1)

### **Инструкции**

1. *В тази задача е* ***забранено*** *използването на всички библиотеки от STL*
2. *Не променяйте предоставяните публични интерфейси (методи и полета) на класовете, тъй като тези методи ще се използват в автоматични тестове и ако имат променена сигнатура тестовете няма да компилират и ще получите 0 точки. От вас се очаква да имплементирате дадените методи.*
3. За да компилира кодът ви трябва всички методи да имат имплементация, дори да връщат грешен отговор.
4. Позволено е да добавяте други методи/класове, за да реализирате задачата. Тях няма да ги тестваме.

### **Условие**

Дадена е програма за управление на автомат за напитки.

Напитката се характеризира по следния начин:

* Име (низ, с динамичен размер)
* Количество, в литри (*подсказка: може да имаме и бутилка от 500мл*)
* Калории
* Цена

Автомата за напитки трябва да поддържа следните функционалности:

* Добавяне на напитка
  + Ако напитката не съществува (не съществува напитка със същото име), метода да връща true
  + Ако напитката съществува, метода да връща false и да не добавя напитката
* Закупуване на напитка
  + Потребителя подава на машината дадена сума пари и име на напитката.
    - Ако напитката съществува и сумата е достатъчна , напитката се премахва от машината и парите се добавят към машината. Методът връща стойност 0
    - Ако парите не са достатъчни, но напитката съществува - напитката не се премахва, но парите се добавят към машината. (Приемаме, че ако искаме да закупим напитката, трябва да дадем достатъчен брой пари - получаването на напитка се случва само след подадена достатъчна сума) Методът връща стойност 1.
    - Ако напитката не съществува - парите не се добавят към машината. Методът връща стойност 2.
* Проверка на получените пари от продажби (приемаме, че нашата машина не връща пари - ако дадена напитка струва 1.60, но ние сме дали 1.80, в машината остават 1.80)

## **Задача 2: Система за автоматични тестове (6 точки)**

## Линк към шаблона:[тук](https://github.com/lyubolp/OOP-SI-2021/tree/main/HW1/task2)

### **Инструкции**

1. *В тази задача е позволено използването на всички библиотеки от STL с ИЗКЛЮЧЕНИЕ НА класа Error, в който са забранени std::vector и std::string.*
2. *Не променяйте предоставяните публични интерфейси (методи и полета) на класовете, тъй като тези методи ще се използват в автоматични тестове и ако имат променена сигнатура тестовете няма да компилират и ще получите 0 точки. От вас се очаква да имплементирате дадените методи.*
3. За да компилира кодът ви трябва всички методи да имат имплементация, дори да връщат грешен отговор.
4. Позволено е да добавяте други методи/класове, за да реализирате задачата. Тях няма да ги тестваме.

### **Условие**

Асистентите по ООП са решили да тестват автоматично задачите на студентите тази година

и се нуждаят от вашата помощ да напишат системата си. Предоставили са ви основни

изисквания към системата, и основния интерфейс, който искат да ползват, като имплементацията

е ваше задължение.

Следват основните инструкции към задачата, като подробности може да намерите в кода.

#### Тестов пакет (TestSuite)

Всяка задача по ООП ще си има собствен набор от тестове наричан Тестов Пакет.

Тестовият пакет трябва да има:

* Име (може да се променя)
* Множество тестови сценарии
* Възможност за добавяне на нов сценарии към множеството
* Възможност за филтриране на преминаващи/непреминаващи сценарии
* Възможност за филтриране на сценариите по тип грешка
* Възможност за премахване на всички тестови сценарии с даден тип грешка

#### Тестов сценарий (TestCase)

Тестов сценарии представлява отделен тест за точно една функционалност. Всеки сценарий

трябва да има:

* Име
* Възможност да се провери дали теста е бил успешен
* Възможност да се провери дали има грешка
* Възможност да се провери типа на грешката
* Възможност да се провери съобщението на грешката

#### Грешка (Error)

❗❗❗ **За този клас е забранено използването на класовете std::vector и std::string**

Всеки тест може да е успешен, а може и да не е. Когато един тест не е минал успешно е

необходимо да имаме допълнителна информация какво се е счупило. За тази цел тестовите

сценарии могат да имат грешка, която трябва да има:

* Тип (Празна/Никаква грешка, Грешка при компилация, Предупреждение, Неуспех при сравнение)
* Съобщение
* Възможност да се проверява типа на грешката
* Възможност да се проверява дали грешката има съобщение
* Възможност да се проверява съобщението на грешката ако има такова
* Възможност да се създава нова грешка от всеки един тип