**CITB408 Програмиране с Java**

Пролетен семестър 2019/2020

**Задание за разработване на проект - Вариант 2**

**Елена Шаварска, F91033**

**Документация по разбработката на проекта**

1. **Клас Product**Клас Продукт, която ще се продава в магазин – съхранява идентификационен номер, име, единична цена на стоката и срок на годност;   
   Идентификационният номер и срока на годност са константи, за да не могат да се променят след като веднъж се зададат.  
   Срокът на годност е реализиран чрез java.util.Calendar.  
   Реализирани са get методи и toString метод.
2. **Клас Cashier**Клас Касиер – съхранява информация за идентификационен номер и име на касиера;  
   Идентификационният номер е константа, за да не може да се променят след като веднъж е зададен.  
   Реализирани са get методи и toString метод.
3. **Клас Cart**В клас Количка имаме хешмап с ключ от тип Продукт и стойност от тип double, която показва какво количество от продукта сме добавили в количката. Променливата е от тип double, за да може да се използва за продукти, които се мерят на килограм – например плодове и зеленчуци.  
   Имаме също променлива от тип магазин – в кой магазин се продава продукта. Чрез нея достъпваме информацията какво количество от продукта е налично в магазина.  
   Реализиран е метод за добавяне на продукти в количката, който се извиква с променлива от тип Продукт и double количество. Хвърля *InvalidQuantityException* ако количеството е по-малко или равно на 0. Ако количетвото на продукта е повече от наличното количество в магазина се хвърля *InsufficientQuantityException*.  
   В противен случай се проверява дали продуктът вече не е бил добавян в кошницата – ако да, количеството просто се увеличава, иначе просто се добавя като нов продукт.

Реализирани са get методи и toString метод.

1. **Клас CashRegister**Клас Каса съхранява информация за това кой касиер работи на касата в момента, общия оборот на касата, в кой магазин се намира, както и касовата бележка и количката на сегашния клиент.  
   Реализирани са следните функции:
   * **public void newClient(Calendar date, Cart cart)–** задава датата на покупката, както и количката с продукти на клиента. Датата е реализирана чрез java.util.Calendar.  
     Създава нов обект от клас Касова бележка, задава го като текуща касова бележка за касата и увеличава общия брой касови бележки в магазина.
   * **public void addProducts() –** Разделя изпълнението на програмата на отделни нишки, за да може всяка каса да добавя продукти паралелно. Обхожда хешмап с продуктите от кошницата, като добавя всеки един чрез метода *addProduct().* Изписва информация за това кой касиер работи на касата в момента и кой продукт се добавя. Хваща InsufficientQuantityException и InvalidQuantityException, които се хвърлят от *addProduct()*.
   * **private void addProduct(Product product, double quantity)** – Проверява дали количеството, което трябва да се добави от дадения продукт е по-голямо от 0 и дали има достатъчна наличност в магазина, в противен случай хвърля съответно *InvalidQuantityException* и *InsufficientQuantityException.* Добавя продукта със съответното количество към касовата бележка и го изважда от магазина чрез метода *shop.sellProduct(product,quantity).*
   * **public void payment() –** Извиква методите *print()* и *saveToFile()* за касовата бележка. Добавя оборота от покупката (касовата бележка) към оборота на касата.
2. **Клас Receipt**

Клас Касова бележка съдържа пореден номер на бележката, касиер, който издава касовата бележка, дата и час на издаване на касовата бележка, списък със стоки, които се включват в касовата бележка включително цената и количеството им – ArrayList от тип ReceiptEntry, обект от клас Магазин – от кой магазин е бележката, както и общата цена на всички продукти. Имаме и една статична променлива *numberOfReceipts,* която брои колко издадени бележки има до момента. По нея се задава поредния номер на всяка нова бележка.

Реализирани са следните методи:

* **public void addProduct(Product product, double quantity) –** проверява дали даденият продукт вече се съдържа в списъка с продукти. Ако да – увеличава количеството и общата цена, ако ли не – добавя го като нов елемент от списъка. Метода хваща изключение от тип *InvalidQuantityException,* което се хвърля от метода *add(quantity)* на класа *ReceiptEntry.*
* **public void print()** – Принтира касовата бележка форматирано на конзолата.
* **public void saveToFile()** – Запазва касовата бележка във файл със име Receipt\_ + поредния номер на бележката.
* **public double getProfit() –** методът взима файла на текущата бележка и извлича от него информацията след “Total:”, която връща като оборот.
* Реализирани са get методи и toString метод.

1. **Клас ReceiptEntry**

Класът съдържа Продукт, неговото количество и цената за съответното количество от този продукт. Използва се за улеснение в списъка от продукти на касовата бележка. Има конструктор, функция за добавяне на количество към даден продукт, get методи и toString метод. Функцията за увеличаване на количеството хвърля *InvalidQuantityException,* ако желаното за добавяне количество е по-малко или равно на 0.

1. **Клас Shop**Съдържа име на магазина, хешмап с наличните продукти (ключ – продукта, стойност – налично количество), лист с касиерите и лист с касите в магазина. Има също променлива, която пази информацията за броя касови бележки, издадени в магазина.
2. **Клас Util**Класът реализира четенето от касова бележка от файл, защото тази функция не пасва към останалите класове. Класът е абстрактен, защото не ни трябва обект от него. При подаване на String с името на файла, ред по ред той се изписва на конзолата.
   * **public void productDelivery(Product product, double quantity) –** метод за доставка на нови продукти. Ако количеството, което се доставя, не е по-голямо от нула, се хвърля *InvalidQuantityException.* В противен случай се проверява дали вече има налични бройки от продукта и количеството трябва само да се увеличи, или той никога не е бил доставян и трябва да се добави.
   * **addCashRegister и addCashier** добавят съответно нова каса и нов касиер към магазина.
   * **public double profit() –** минава през всички каси в магазина и събира оборота им, за да върне общия оборот на магазина.
   * **public void checkAllProducts()** – принтира всички налични продукти и тяхното количество на конзолата.
   * **public synchronized void sellProduct(Product product, double quantity)**  - Хвърля *InvalidQuantityException,* ако количеството, което трябва да продадем е по-малко или равно на 0. Хвърля *InsufficientQuantityException,* ако няма достатъчно наличност от продукта, който трябва да бъде продаден. Проверява също дали продукта изобщо се продава в магазина. Ако всички изисквания са покрити, намалява количеството на продадения продукт в магазина. Методът е синхронизиран, за да може отделните нишки на касите да не черпят едновременно от един ресурс (количество продукт).
   * Имаме метод за увеличаване общия брой на касовите бележки, get методи и toString метод.
3. **InsufficientQuantityException**

Дава информация от кой продукт колко не достига.

1. **InvalidQuantityException**

Дава информация, че количеството продукт не може да бъде по-малко или равно на 0.