Поправителен изпит по ООП

04.09.2018

Вендинг машина

При изпълнение на задачата използвайте базовият проект, който се намира в `exams/vending-exam`.

В тази задача трябва да се имплементират класовете за вендинг машини и рецепти за продукти, продавани от вендинг машините.

Основните класове са:

- `org.elsys.vending.Recipe` - клас за рецепта, която се произвежда от вендинг машината. Всяка рецепта съдържа колекция от съставки и количества, които се влагат при производство на рецептата. Например - рецептата за “Кафе еспресо” съдържа съставката кафе и количеството кафе, което е необходимо за производство на “Кафе еспресо”. Рецептата за “Кафе еспресо със захар” съдържа две съставки - кафе и захар и количествата от всяка една от съставките.
- `org.elsys.vending.EspressoVendingMachine` - клас за вендинг машина, която произвежда топли напитки - например “Кафе еспресо”, “Капучино”, “Мляко с кафе” и т.н.
- `org.elsys.vending.SmartVendingMachine` - вендинг машина, която следи за наличните количества от съставките, които може да използва, и при намаляване на наличните количества под определен праг сигнализира екипа за поддръжка на машината.

Оценяване

- 60т - Среден 3
- 80т - Добър 4
- 100т - Мн. добър 5
- 120т - Отличен 6

Задачи

1. `Recipe`, `EspressoVendingMachine` и `SmartVendingMachine`

Имплементирайте методите на трите класа. За да сте сигурни, че методите са имплементирани коректно, трябва да изпълните `JUnit` тестовете, които се намират в папката `tests`.

1.1 `Recipe`(10т)

1. (2т) Трябва да има конструктор с 2 аргумента - име на рецептата и цена.
2. (3т) Конструкторът трябва да проверява цената за коректност и ако тя не е коректна да генерира изключение (`RuntimeException`).
3. (2т) Методът `addIngredient` трябва да проверява дали съставката вече не е включена в рецептата и ако съставката вече е включена в рецептата да генерира изключение.
4. (3т) Методът `getIngredients` трябва да връща всички съставки и техните количества като `java.util.Map`. Методът трябва да връща копие на данните в рецептата.

1.2 `EspressoVendingMachine` (25т)

Всяка `EspressoVendingMachine` има набор от контейнери за ингредиенти, които могат да се смесват за получаване на крайните продукти от рецептите.

В по-простите модели (какъвто е и `EspressoVendingMachine`) всички контейнери за съставки имат една и съща вместимост от 5 кг.

1. (5т) При създаването на обект от `EspressoVendingMachine` се предава колекция от имената на съставките, които може да се използват при смесването на крайните продукти. Конструкторът създава обект с необходимите за всички съставки контейнери. Всяка вендинг машина помни натрупаният от продажбите оборот. Всички контейнери в `EspressonVendingMachine` са с еднакъв капацитет от 5 кг.
2. (3т) `public void resupplyContainer(String ingredient)` - напълване на контейнера за дадена съставка до максималният предвиден обем на контейнера. Ако вендинг машината няма контейнер за предадената съставка метода генерира `RuntimeException`.
3. (2т) `public void resupply()` - презареждане на цялата машина и нулиране на натрупаният оборот. (Очакването е този метод да използва `resupplyContainer(String ingredient)`).

4. (2т) `public boolean hasEnoughIngredientSupply(String ingredient, int amount)` - проверява дали наличното количество на дадена съставка е по-голямо или равно от указаната в метода стойност. Ако вендинг машината няма контейнер за предадената съставка метода генерира `RuntimeException`.
5. (2т) `public Collection<String> getIngredientContainers()` - връща колекция от всички видове конфигурирани контейнери за съставка във вендинг машината.
6. (2т) `public int getIngredientContainerCapacity(String ingredient)` - връща капацитета на даден контейнер във вендинг машината. Ако вендинг машината няма контейнер за предадената съставка метода генерира `RuntimeException`.
7. (2т) `public int getIngredientSupply(String ingredient)` - връща наличното количество на дадена съставка. Ако вендинг машината няма контейнер за предадената съставка метода генерира `RuntimeException`.
8. (2т) `public void useIngredient(String ingredient, int amount)` - консумира дадено количество от дадена съставка. Наличното количеството на съставката от съответния вид трябва да намалее. Ако вендинг машината няма контейнер за предадената съставка метода генерира RuntimeException. Ако наличното количество на предадената съставка е по-малко от поисканото метода генерира изключение `RuntimeException`.
9. (5т) `public void brewRecipe(Recipe recipe)` - изпълнява предадената рецепта. Ако вендинг машината не разполага с необходимото количество на някой от ингредиентите, то метода генерира изключение.

1.3 `SmartVendingMachine` (30т)

`SmartVendingMachine` има възможност да изпраща съобщения на екипа по поддръжка при намаляване на количеството на дадена съставка. За тази цел `SmartVendingMachine` има имплементиран метод `public void notifySupport()`.

1. (5т) Направете класа `SmartVendingMachine` наследник на класа `EspressoVendingMachine`.
2. (5т) Предефинирайте метода `public void brewRecipe(Recipe recipe)` така, че при изпълнение на дадена рецепта да проверява наличността на всички съставки и ако някоя от съставките е по-малко от 20% от капацитета на контейнера, да извика метода `notifySupport()`.
3. (10т) Научете метода `public void brewRecipe(Recipe recipe)` да изпраща съобщение само веднъж, а не при всяко извикване.
4. (10т) Изпратете повторно съобщение до поддръжката, когато някой от ингредиентите стане по-малко от 10% от капацитета на контейнера. Това съобщение също трябва да се изпрати само веднъж.

2. main (19т)

Дефинирайте main метод в класа `org.elsys.vending.MainClass`, който прави следното:

1. (5т) прочита ред от стандартния вход, който съдържа описание на `EspressoVendingMachine` и създава такава вендинг машина. Имената на съставките за които машината има контейнери са разделени със запетайки: Например:

   ```
   coffee, milk, sugar
   ```

2. (5т) Чете редове от стандартният вход докато не прочете END. Всеки прочетен ред е рецепта, която трябва да изпълни от вендинг машината. Елементите на рецептата са разделени с точка и запетая. Първият елемент е името на рецептата, вторият елемент е цената на рецептата, а следващите елементи са съставките и техните количества. Името на съставката и количеството и са отделени едно от друго със запетая. Например:

   ```
   Coffee with milk; 2.0; coffee, 5; milk, 10
   Coffee with milk and sugar; 2.0; coffee, 5; milk, 10; sugar, 5
   END
   ```

3. (5т) Изпълнението на рецептите трябва да е оградено в `try/catch` блок и да обработва изключенията, които може да се генерират.
4. (4т) След приключване на четенето и изпълнението на рецепти трябва да отпечатате натрупаният от машината оборот.

3. Поддръжка за вендинг машина с контейнери с различен капацитет - йерархия от вендинг машини(32т)

1. (8т) Дефинирайте абстрактен клас за вендинг машина `AbstractVendingMachine`. Направете `EspressoVendingMachine` наследник на `AbstractVendingMachine`. Разпределете методите и полетата на `EspressoVendingMachine` адекватно между двата класа, така че да отговарят на основните принципи на обектно ориентираното програмиране..
2. (8т) Дефинирайте клас `FlexibleVendingMachine`, която поддържа контейнери с различен капацитет.
3. (8т) Дефинирайте клас `SmartFlexibleVendingMachine`, който наследява FlexibleVendingMachine и поддържа функционалността за изпращане на съобщенията дефинирана в `SmartVendingMachine`. При дефиниране на `SmartFlexibleVendingMachine` обърнете специално внимание на йерархията на класове и избягвайте дублирането на код.
4. (8т) Дефинирайте jUnit тестове за разработените класове.

4. Използване на Java Streams API (32т)

1. (8т) В базовият абстрактен клас `AbstractVendingMachine` добавете функционалност за съхранение и търсене на рецепти по тяхното име. Методите за търсене трябва да използват Java Streams API.
2. (8т) Добавете метод `getRecipesByIngredient(String ingredient)`, който намира всички рецепти, в които се използва предадената съставка. Методът трябва да използват Java Streams API.
3. (8т) Добавете метод `getRecipesByPrice(double fromPrice, double toPrice)`, който търси рецепти в ценовият диапазон между fromPrice и toPrice.
4. (8т) Добавете метод `getRecipesSortedByPrice()`, който връща колекция от всички рецепти, подредени по тяхната цена от най-малката към най-голямата.