PV178 – Domácí úkol #2

Vending Machine

Vaším úkolem bude **implementace řídící logiky automatu** na výdej nápojů/snacků. Obsluha automatu je inspirována skutečným výdejním automatem, pro provedení objednávky je tedy zapotřebí **nejdříve vhodit mince** (automat si je bude v součtu pamatovat jako vložený kredit), **pak zvolit produkt** na příslušných souřadnicích a **následně provést potvrzení** objednávky.
Pro zjednodušení zadání předpokládejme, že po vhození mincí už nelze objednávku zrušit, ale je třeba si vybrat produkt a objednávku uskutečnit (automat tedy vrací případný zbytek pouze při realizaci objednávky). Pro snadnou interakci bude program disponovat sadou před definovaných příkazů, skrze které uživatel uděluje pokyny řídící jednotce automatu (jednotlivé příkazy viz níže). Samotná implementace pak bude využívat návrhový vzor State Pattern o kterém se můžete více dozvědět například zde, případně si projít ukázkovou implementaci.

1. Ve složce **Exceptions** vytvořte následující položky:

    a. Vyjímku s názvem **StockUnavailableException**, která bude vyvolána v případě, že došel daný typ zboží (podrobnosti o jejím vyvolání jsou uvedeny níže)

2. Ve složce **Model** vytvořte následující položky:

    a. strukturu **Coordinates,** která bude reprezentovat souřadnice produktu v rámci automatu.
        i. **Souřadnice jsou tvořeny dvěma indexy**, prvním z nich je standardně **index řádku** (klasicky indexován od 0), který je **typu int** a druhým je **index sloupce**, pro který se používají velká tiskací písmena (indexován od ‘A’) a je tedy **typu char**.
        ii. Uvedené indexy, respektive jejich hodnoty pak bude struktura Coordinates přijímat jako parametry konstruktoru (v uvedeném pořadí).
        iii. Struktura bude mimo jiné definovat **defaultní hodnotu pro souřadnice** s názvem **Empty**, která bude deklarovaná jako statická a bude mít hodnotu 0 pro **index řádku** a hodnotu ’\0’ (výchozí hodnota typu char) pro index sloupce.
        iv. **Souřadnice tedy budou mít tyto veřejné členy:**
            - *… int RowIndex ...* (index řádku)
            - *… char ColumnIndex ...* (index sloupce)
            - *… static ... Coordinates Empty ...* (defaultní souřadnice)
        v. Coordinates budou mít, mimo jiné, také **přepsanou metodu ToString()**, která bude vypisovat souřadnice v následujícím formátu: *"[{RowIndex};{ColumnIndex}]"*

    b. třídu Product, která představuje produkt nabízený automatem.
        i. **Produkt bude mít následující veřejné členy:**
            - *… string Name …* (název produktu)
            - *… decimal Price …* (cena produktu)
        ii. Hodnoty těchto členů pak bude třída Product v uvedeném pořadí přijímat v konstruktoru.

iii. Product bude mít, mimo jiné, také **přepsanou metodu ToString()**, která bude vypisovat souřadnice v následujícím formátu:
"{Name} for {Price},- CZK"

c. třídu Stock, která představuje zboží nabízené automatem. Vztah mezi zbožím a produktem je definován kompozicí, tedy Stock obsahuje instanci třídy Product.
   i. **Stock bude mít následující veřejné členy:**
      - *… Product Product …* (produkt, který obsahuje Stock)
      - *… int Quantity …* (mnozstvi daného produktu)
      - *… DispatchStock()* metoda která dekrementuje property Quantity a to pouze v případě, že její hodnota je větší než 0, jinak vyhodí StockUnavailableException s následující zprávou:
        "No units of {nazev_produktu} available."
   ii. Součástí třídy Stock bude také **konstruktor s parametry** nastavující hodnoty členů Product a Quantity a to opět v uvedeném pořadí.
   iii. Stock bude mít, mimo jiné, také **přepsanou metodu ToString()**, která bude vypisovat souřadnice v následujícím formátu:
   "{Product} (available: {Quantity}x)"

3. Ve složce **Machine/ControlUnit** implementujte rozhraní IControlUnit třídou **ControlUnit**:

   a. Implementovaná třída ControlUnit, má za úkol si udržovat a spravovat data ohledně veškerém obsaženém zboží a jeho umístění (souřadnicích) v rámci automatu. K tomuto účelu si tedy nejprve **zvolte vhodnou datovou strukturu**.

   b. **Konstruktor** této třídy bude mít předpis (můžete vždy předpokládat neprázdná pole): *public ControlUnit(int[] rowIdentifiers, char[] columnIdentifiers)* kde pole *rowIdentifiers* bude vždy obsahovat sekvenci po sobě jdoucích čísel indexovaných od nuly a pole *columnIdentifiers* pak bude vždy obsahovat po sobě jdoucí velká písmena, která budou začínat vždy od ‘A’.

   c. Property State a metodě SwitchToState bude dále věnována pozornost v bodě 4.

   d. Implementujte zbylé metody předepsané rozhraním IControlUnit (kde naleznete i jejich podrobné popisy). Pokud je některé z těchto metod předán nevalidní parametr typu Coordinates, dojde k **vyhození vhodného typu výjimky** dle konkrétních hodnot (parametr může být prázdný, indexy souřadnic mohou být mimo rozsah, atd.) Při implementaci se snažte zabránit duplikátnímu kódu pomocí vhodné dekompozice na dílčí metody.

4. Nyní přejděte do složky **Machine/States**, jak již z názvu podsložky vyplývá, zde budete vytvářet jednotlivé stavy v rámci kterých se řídící jednotka musí nacházet. Ještě než začnete se samotnou implementací stavů, **ujistěte se, že** dobře **chápete jak** samotný **State pattern funguje**, vaše implementace by rozhodně neměla začínat psaním množství podmínek.

   a. V této úloze budeme pro jednoduchost předpokládat **3 základní stavy**:
      i. **InsertCoinState** což je iniciální stav automatu, v kterém je hodnota kreditu (property pro uložení kreditu je zmíněna dále) rovna nule, po vhození libovolné kladné nenulové částky pak automat přejde do stavu:

ii. **SelectCoordinatesState** kde opět jak název napovídá se od uživatele očekává zvolení souřadnic produktu respektive zboží. Uživatel musí vybrat právě ty souřadnice, kde hodnota zboží není null, jiná omezení ve fázi výběru nejsou.
iii. **ConfirmOrderState**, který následuje po předchozím stavu je pak charakterizován kladným, nenulovým kreditem a validními (splňujícími uvedené podmínky) souřadnicemi produktu. V této části uživatel typicky potvrzuje svou objednávku, může však také zvýšit hodnotu kreditu, případně měnit souřadnice požadovaného zboží. Po výdeji zboží (vždy proběhne výdej pouze jednoho kusu) se automat uvede opět do iniciálního stavu.

b. Všechny **výše uvedené stavy budou reprezentovány třídami s identickými názvy**, které budou implementovat rozhraní IState. Každý stav si tedy udržuje aktuální kredit a vybrané souřadnice, dále pak také odkaz na řídící jednotku která poskytuje funkcionalitu pro správu zboží v rámci automatu.

c. Každý stav bude mít následující dva konstruktory:
i. ...*State(IControlUnit unit)* konstruktor přijímající referenci na řídící jednotku který je vhodný pro prvotní instanciaci stavu
ii. ...*State(IState state, IControlUnit unit)* konstruktor přijímající navíc ještě referenci na předchozí stav, která si uchovává aktuální hodnoty kreditu a zvolených souřadnic, tato varianta konstruktoru je vhodná pro instanciaci nového stavu z předchozího (při změně stavu)

d. Nyní se na chvíli vraťte do složky **Machine/ControlUnit** k implementaci třídy ControlUnit. Property **State** předepsaná příslušným rozhraním bude uchovávat aktuální stav řídící jednotky, metoda **SwitchToState** provede změnu stavu řídící jednotky (viz sekce c) část ii)).

e. V **Machine/States** pak pro každý stav implementujte předepsané metody (popisy jsou opět součástí rozhraní). Při implementaci zohledněte stavové hlášky, detailně popsané v níže uvedené sekci f). Opět se vyhněte duplicitnímu kódu (využijte toho, že je povoleno přidávat další třídy, může být například vhodné vymyslet vhodnou hierarchii pro stavy řídící jednotky).

f. Pro poskytnutí zpětné vazby uživateli, budou jednotlivé metody implementované v předcházející sekci d) při svém volání **vypisovat na konzoly následující zprávy:**
i. ***RaiseCredit()*** v případě validního parametru (value je větší jak nula) se vypíše: *"Credit: {aktuální_hodnota_kreditu},- CZK"* jinak dojde k vyhození vhodného typu výjimky
ii. SelectProduct()
- Pokud ještě **není vložen kredit**, dojde k v každém případě k výpisu *"Insert coin first."* (nehledě na správnost souřadnic), pokud je již hodnota kreditu větší jak nula, platí následující body:
- Jestliže jsou **zadány platné souřadnice a nachází se na nich instance třídy Stock, která nemá hodnotu null**, vypíše se zpráva: *"Row: {index_radku} and column: {index_sloupce} are now selected."*
- **V případě** platných souřadnic na kterých je **Stock s hodnotou *null*** dojde k výpisu: *"There is no stock available at {zadane_souradnice}"*, v tomto případě také nedojde k nastavení zadaných souřadnic.

- Pokud jsou **zadány neplatné souřadnice** pak dojde k vyhození vhodného typu výjímky (jak bylo uvedeno v části **3. d)**)

iii. TryDeliverProduct()

- Opět platí, že pokud ještě **není vložen kredit**, dojde k v každém případě k výpisu *"Insert coin first."* (nehledě na to zda jsou či nejsou zadány souřadnice), pokud je již hodnota kreditu větší jak nula, platí následující body:
- Pokud ještě **nejsou vybrány validní souřadnice Stocku**, vypíše se zpráva: "No coordinates were given.", ve zbývajících případech platí:
- Pokud **hodnota kreditu není dostatečná pro nákup** daného zboži, dostane uživatel na výstup: *"Not enough credit."*
- V případě že během transakce dojde k **vyvolání** vámi implementované vyjímky **StockUnavailableException**, tuto výjimku v této metodě zachyťte a na výstup vypište hodnotu její property Message, v tomto okamžiku nastavte vybrané souřadnice na prázdnou (Coordinates.Empty) hodnotu a přepněte řídící jednotku do stavu SelectCoordinatesState
- Pokud **vydání zboží nic nebrání**, proveďte výdej zboží a uveďte automat opět do iniciálního stavu, uživatel dostane následující zprávu: "Delivered {vybrany_produkt}, returned {zbytek_kreditu},- CZK."

Uvedená specifikace pro výpis zpráv do konzole obsahuje velké množství podmínek, při implementaci výše uvedených metod se však **vyhněte přílišnému větvení kódu**, místo toho využijte výše popsané a případné vlastní přidané stavy automatu (jedna z výhod State Patternu).

5. Na závěr ve složce **Machine** proveďte implementaci třídy **VendingMachine** podle rozhraní IVendingMachine (opět obsahuje popisy jednotlivých metod), tato třída představuje kontrakt mezi uživatelem a řídící jednotkou automatu (její metody se volají po zadání příslušných příkazů uživatelem do konzole). Kromě popisu metod v rámci rozhraní IVendingMachine **bude třída VendingMachine mít:**

   a. **Konstruktor s parametrem typu IControlUnit**, kde třída dostane referenci na svou řídící jednotku, kterou si pochopitelně uchová v property ControlUnit

   b. Metody *InsertCoin(...), SelectCoordinates(...)* a *ConfirmOrder()* budou **pouze využívat funkce poskytované řídící jednotkou, případně jejím aktuálním stavem** (implementace bude tedy typicky jednořádková).

   c. Metodu GetCurrentOffer() **implementovanou následovně**:
      i. Pro získání všech položek bude využita již implementovaná metoda v rámci řídící jednotky
      ii. Návratová hodnota typu IEnumerable<string> bude mít každou položku ve formátu: *"{Coordinates} {Stock}"*, kde Coordinates jsou souřadnice pro dané zboží

   d. O inicializaci dat a třídu Program se **nemusíte nijak starat** (po dokončení implementace jednoduše do spuštěné konzole vepište příkaz *"help"* pro vypsání všech dostupných příkazů).

V případě jakýchkoliv nejasností v zadání můžete **využít přiložené unit testy**, případně **napsat na *409727@mail.muni.cz* nebo do příslušného vlákna v rámci předmětového fóra.**

Unit testy přiložené k zadání **jsou záměrně nekompletní**, k testování tedy využijte i naimplementovanou obsluhu konzole třídou Program.

Při vypracování domácí úlohy **dodržujte jmenné konvence, principy zapouzdření a názvy a formáty výpisů předepsané zadáním** (jinak nebudou procházet unit testy).

Je také **zakázáno jakkoliv modifikovat jak předpřipravené zadání, tak přiložené unit testy** (při kontrole budou stejně nahrazeny kompletní sadou testů).

V případě zjištění provedení nepovolených změn bude odevzdané vypracování ohodnoceno nulovým počtem bodů.