Testovaní softwaru

Cílem cvičení je procvičit si na příkladu, jak přetavit specifikace ve formě textového popisu do testů, které ověří shodu implementace se specifikacemi. (Jde především o princip vytváření testů; zde použitá realizace testů je zvolena z důvodu jednoduchosti.)

Rosemary's Grocery Store

Dostali jste se do týmu, který vyvíjí počítačovou hru. Rosemary je obchodnice, která v této hře vede obchůdek se smíšeným zbožím (jehož implementaci máte ve správě). Obchod prodává především potraviny (tzv. normal items), ale Rosemary umí sehnat a nabídnout také jisté speciální předměty. Rosemary je pečlivá a vede si evidenci o stavu zboží, které má na skladě. Na konci každého dne stav všeho zboží aktualizuje.

Zboží

Každé zboží nebo předmět, které má Rosemary na skladě, budeme reprezentovat instancí třídy `Item` :

```
class Item:

    def __init__(self, name, days_left, quality):
        self.name = name
        self.days_left = days_left
        self.quality = quality
```

Každé zboží či předmět má tedy

- název (`name` , řetězec),
- dobu trvanlivosti (`days_left` , celé číslo), tedy zbývající počet dní (následujících po aktuálním dni), do kdy je nutné zboží prodat, než se začne výrazně kazit, a
- aktuální kvalitu (`quality` , celé číslo).

Speciální předměty

Rosemary může mít na skladě i speciální předměty, které se určují podle specifického jména. Mezi tyto speciální předměty patří

- zrající sýr (`"Aged Brie"`),
- diamant (`"Diamond"`) a
- vstupenky (`"Tickets"`) na nějaké hodně žádané představení.

Tyto předměty jsou zvláštní v tom, že se chovají (stárnou) jinak než normální zboží.

Aktualizace kvality

Ve hře se na konci každého dne zavolá pro každé zboží na skladě funkce `update()` . Ta aktualizuje (obvykle sníží) hodnoty `days_left` a `quality` :

```
>>> bread = Item('Bread', days_left=3, quality=5)
>>> update(bread)
>>> print(bread.days_left)
2
>>> print(bread.quality)
4
```

Poznámka 1: Funkce `update()` je modifikátorem, tj. nic nevrací, ale přímo na místě modifikuje instanci třídy `Item` , kterou jí předáme jako argument.

Poznámka 2: Z hlediska objektového návrhu by bylo lepší, kdyby `update()` byla metodou ve třídě `Item` a kdybychom mohli od třídy `Item` odvodit další třídy, např. BrieItem, DiamondItem, apod., které by věděly, jak sami sebe aktualizovat. Celý zbytek hry ale využívá pouze `Item` ve formě uvedené výše; zde popsané změny by si vyžádaly dalekosáhlé úpravy v kódu celé hry, a proto je zatím nemáme právo udělat.

Pravidla aktualizace

Funkce `update()` předpokládá, že jako vstup obdrží instanci třídy `Item` naplněnou platnými hodnotami, a musí dodržet následující pravidla:

- Název zboží nebo předmětu se nikdy nemění.
- Kvalita zboží nikdy není menší než 0 a větší než 50 s jedinou výjimkou (Diamanty, viz níže).
- Na konci každého dne, tj. při zavolání `update()` , se kvalita i trvanlivost zboží sníží o 1 (není-li uvedeno jinak).
- Zboží s dosaženou nebo překročenou dobou trvanlivosti (`days_left <= 0`) ztrácí kvalitu 2x rychleji.

- Diamanty neztrácejí ani kvalitu, ani trvanlivost. Kvalita diamantů je vždy 100 (výjimka z pravidla, že kvalita nemůže být vyšší než 50).
- Zrající sýr s ubíhajícím časem kvalitu neztrácí, ale získává (kvalita se zvyšuje o 1), nezávisle na `days_left`.
- Vstupenky s ubíhajícím časem také získávají na kvalitě:
  - +1, pokud do akce zbývá víc než 10 dní,
  - +2, pokud do akce zbývá 6-10 dní,
  - +3, pokud do akce zbývá 1-5 dní.
  - Ale v den konání akce se vstupenky stanou bezcennými (kvalita klesne na 0).

Váš úkol

Funkce `update()` už je implementována. Vaším úkolem je sestavit sadů testů pro tuto funkci, které ověří, že se implementace funkce chová podle specifikací. Konkrétně je vaším úkolem vytvořit modul `test_rosemary.py` s co nejúplnější sadou testů, který nahrajete do BRUTE, kde vám oznámíme, jak dobrou/úplnou sadu testů máte.

Testujte jen výše uvedené specifikace. Funkce `update()` předpokládá, že bude vždy volána s 1 argumentem, kterým bude instance třídy `Item`, a tato instance bude obsahovat platná data. Není úkolem funkce `update()` testovat chybný vstup, a proto ani vy byste neměli testovat, jak se funkce zachová při chybném vstupu.

Při testování softwaru je ideální **v každém testu testovat pouze jedinou věc** tak, aby při selhání testu bylo ihned zřejmé, co je špatně. (Kdyby měl test mnoho možností, proč selhat, netušili byste, která z nich za selhání může.) Je proto zcela obvyklé mít hodně testů a mít je krátké!

Obsah modulu s testy

Specifikace:

- Všechny testy budou v jednom modulu (`test_rosemary.py`).
- Modul na začátku importuje třídu `Item` a funkci `update` z modulu `rosemary.py`. Tento modul nemáte k dispozici, v BRUTE jej připojíme k vašemu odevzdanému modulu s testy. (Nic vám nebrání zkusit si současně s psaním testů vytvořit vlastní implementaci funkce `update` v modulu `rosemary.py`! Tento modul ale neodevzdávejte!)
- Není třeba používat jiné moduly kromě modulu `rosemary.py`. Moduly `inspect` a `importlib` budou zakázány.
- Každý test bude samostatná funkce, jejíž jméno bude začínat na `test_`.
- Každá taková funkce bude vracet buď
  - `True`, když se funkce `update` bude chovat podle testované specifikace, nebo
  - `False`, pokud detekujete odchylku od specifikací.

- **Testové funkce by měly být krátké!** Funkce delší než 10 řádků budeme ignorovat!
- Testové funkce nebudou přijímat žádné argumenty - náš odevzdavací skript jim žádné argumenty nepředá a bude vypisovat zvláštní chybu

```
Traceback (most recent call last):
  File "check_test.py", line 128, in run_test
    result = getattr(module, fn.fn_name)()
```

- Testové funkce mohou běžet maximálně 1 vteřinu.
- Doporučení: nazývejte testy popisnými jmény, aby bylo zřejmé, co testují. I přesto se jména pokuste udržet krátká!

Vzor souboru `test_rosemary.py` :

```
from rosemary import Item, update

def test_normal_item_decreases_days_left():
    # Prepare for the test
    item = Item('Bread', days_left=3, quality=5)
    # Call to the tested function
    update(item)
    # Check the specification
    return item.days_left == 2

def test_...
    ...

def test_...
    ...
```

Poznámka: Zde se zcela obejdeme bez jakéhokoli testovacího frameworku, dokonce i bez modulu `testing.py` , který jsme si vytvářeli na přednášce.

Jak budeme vaše testy testovat?

Jak už bylo řečeno, v BRUTE k vašemu testovému modulu `test_rosemary.py` připojíme naši implementaci funkce `update()` v modulu `rosemary.py` . Máme k dispozici nejen (snad) správnou implementaci, ale i mnoho chybných s mnoha různými chybami. Vaši sadu testů vyzkoušíme na všech implementacích funkce `update()` . Cílem je, aby

- žádný test nehlásil chybu (nevracel `False`) pro správnou implementaci a zároveň
- pro každou chybnou implementaci alespoň jeden z testů zahlásil chybu (vrátil `False`).

Domácí úkol

Zůstává příprava z minula!

courses/b4b33rph/cviceni/program_po_tydnech/tyden_07_testovani.txt · Last modified: 2021/11/18 10:16 by xposik