Programowanie Obiektowe
Kolekcje, wyjątki i generyki
Zadanie oceniane nr 2
4-05-2022

Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (add + commit + push) w katalogu o nazwie w stylu: zadanie_oceniane_2. Fakt wgrania plików do swojego repozytorium można sprawdzić samodzielnie logując się (via www) na swoje konto i sprawdzając czy pojawiły się tam wszystkie zmiany.

"Pralnia"

Parę lat temu nie odebrałem na czas pewnego elementu garderoby który oddałem do pralni. Po kilku tygodniach gdy się po niego zgłosiłem okazało się z wspomniana szata zniknęła. Rozpoczęto zakrojone na szeroką skalę poszukiwania które zapewne trwają po dziś dzień. Na pamiątkę tego wydarzenia zasymulujemy sobie dzisiaj proces czyszczenia odzieży. Tym razem żaden obiekt nie zaginie w czeluściach pralnianych kubełków, bo będziemy pamiętać o equalsach i hashcode`ach... które oczywiście wstawione będą tylko tam gdzie jest to niezbędne.

Prace do wykonania:

1. **Stworzyć klasy i ich hierarchie odzwierciedlające wspomniane poniżej elementy (nazewnictwo dobrać wedle uznania)**
   - Koszula (czystość tak/nie***, rozmiar: 35-40, kieszeń*, guziki**)
   - Kufajka (czystość tak/nie*** – domyśnie nie, rozmiar: 35-45, kieszeń*)
   - Płaszcz (czystość tak/nie*** – domyśnie nie, kieszeń*, waga: 5-10)

     * Kieszeń przechowuje elementy takie jak:
     - Guzik(id)
     - Kartka z adresem (ulica – wstawiana z zewnątrz, nr domu 1-222)
     - Granat (odbezpieczony – tak/nie – jeden na 10 nie odbezpieczony)

     Mogą się one powtarzać, mają swój indeks i będą często usuwane.

     Każdy element garderoby mający kieszeń, wypełnia ją conajwyżej 1-3 elementami, których prawdpodobieństwo wystąpienia jest takie samo. Adres jakikolwiek String.

     Zakładamy że kartka z adresem jest równa innej kartce jeśli przechowywana w niej nazwa ulicy i nr domu się zgadzają. Natomiast to czy dwa guziki są sobie równe, rozpoznajemy po id. Pozostałe elementy są "fizyczne". 1 element = 1 miejsce w pamięci. Dwa obiekty (instancje klasy) reprezentują zawsze dwa różne elementy.

     ** Guziki
     Kolekcja guzików w której nie mogą się powtarzać ale pamiętana jest kolejność ich dodawania. Będzie ich 6. Kolekcję wypełniamy przy tworzeniu koszuli.

     *** czystość – domyśnie nie

   - Pralnia
     Posiada kolekcje zawierające referencje na:
     - elementy garderoby przyjęte do prania (kolejność nieistotna, bez powtórzeń)
     - wyprane elementy (kolejność wstawiania istotna, bez powtórzeń)
     - coś co przechowuje relacje na lini element garderoby (klucz) z kolekcją rzeczy wyjętych z kieszeni (wartość)

     Należy w dobrze znany sposób wystawić na zewnątrz metody przeznaczone do obsługi pralni.

     - ✔ putToWash (Element garderoby)
       Wstawia element do kolekcji zawierającej elementy przyjęte do prania
     - ✔ washAll()
       - ➔ przegląda szaty przeznaczone do prania pod kątem zawartości kieszeni i opróżnia je
       - ➔ znalezione fanty usuwa z miejsca odnalezienia i dodaje do kolekcji trzymanej dla elementu garderoby z którego kieszeni były one wyjęte
       - ➔ w przypadku gdy w kieszeni znaleziony zostanie odbezpieczony granat, wtedy jest to traktowane jak niedopuszczalna sytuacja, która zagraża bezpieczeństwu i zdecydowanie wymaga usprawiedliwionego przerwania tego procesu.

- ➔ Ustawia czystość na true
- ➔ Wrzuca do wypranych elementów

- ✔ pickUpWashedClothes()
  Zwraca wyprane elementy garderoby.
- ✔ getPocketStuffByClothes(element garderoby)
  Dla przekazanego elementu garderoby zwraca kolekcję elementów wyjętych z kieszeni.

- Klient
  Ma dostęp do pralni (taki żeby widział metody które są mu potrzebne i żadnych innych)

  Posiada metodę doJob(), która tworzy po jednym elemencie garderoby każdego typu, przekazuje je do pralni po kolei, pierze wszystkie, odbiera ubrania i zawartość kieszeni. Nigdzie ich nie przechowuje.

- Świat
  Posiada metodę goLive(), która tworzy pralnię, dodaje 10-ciu klientów do indeksowalnej kolekcji bez konieczności szybkiego usuwania, listuje ją uruchamiając metodę doJob() na każdym z nich.

2. **Uczynić pralnię parametryzowalną**
Wstawić dodatkowe pole: certificate do Pralni, którego typu nie musimy znać na etapie implementacji tego przybytku. Właściwy typ (trzeba go stworzyć) będzie przekazany dopiero podczas tworzenia instancji pralni oraz przez konstruktor. Jedyna logika biznesowa to jakikolwiek ciąg znaków zwracany w ramach reprezentacji tekstowej tego obiektu (np. "hdkr4826df94kf9444jdfllhh"). Utworzyć metodę retrieveCertificate, która zwraca certyfikat po uprzednim wypisaniu go na konsoli.

3. **Zademonstrować działanie**