Programowanie Obiektowe

Laboratorium 6 – Lista A (2 kwietnia 2022)

mgr inż. Damian Mroziński

Poniższe zadania należy zaimplementować jako elementy wybranego przez siebie "systemu". Może to być zarówno sklep spożywczy, komis samochodowy czy dziekanat przyjmujący studentów na studia.

Stwórz **klasę główną** *Sklep/Dziekanat/Magazyn*, która będzie miała w sobie maina.

Zadanie 6.0(4.0):
Celem będzie stworzenie kilku rodzajów konstruktorów, aby zaobserwować ich użycie w praktyce.
Stwórz **klasę poboczną**, której obiekty będziesz przechowywał w swojej **klasie głównej**, np *Warzywa/Studenci/Narzędzia.* Zaimplementuj w niej konstuktory. Wszystkie z poniższych metod poza swoim zastosowaniem powinny logować swoje wywołania w konsoli.

- konstruktor domyślny, ustawiający wartości na błędne (na przykład "ERROR" dla Stringów, -1 dla wartości liczbowych)
- konstruktor z parametrami (liczba i zastosowanie zależy od przyjętych założeń)
- konstuktor kopiujący
- metodę która wyświetli wszystkie pola.

Zaobserwuj różnicę w nastepujących użyciach (w poniższym kodzie klasą poboczną jest *Item*)

```
1    // obiekty stworzone "osobno"
2    Item itemBlad = //konstruktor domyślny
3    Item itemPomidor = // konstruktor z parametrami
4    Item itemPomidorGalazka = // konstruktor kopiujacy
5
6    // uzycie tablicy
7    Item []tab = new Item[3];
8      // co sie stanie gdy sprobujesz wypisac elementy od razu po tutaj, po zdefiniowaniu tablicy?
9    Arrays.fill( // uzupełnienie tab za pomoca metody fill
10
11   // uzycie kolekcji
12   ArrayList<Item> list = new ArrayList<Item>();
13   list.add( // uzupełnienie kolekcji o obiekt stworzony za pomoca konstruktora domyślnego
14   list.add( // uzupełnienie kolekcji o obiekt stworzony za pomoca konstruktora z parametrami
15   list.add( // uzupełnienie kolekcji o obiekt stworzony za pomoca konstruktora kopiujacego
```

Dla każdego obiektu stworzonego osobno, bądź tych umieszczonych w tablicach/kolekcjach wywołaj metodę wyświetlające je na ekranie.

Kod napisany w ramach poprzedniego zadania pomoże Ci w sprawdzeniu kolejnych zadań.

Zadanie 6.1(3.0):
Celem będzie zrozumienie różnicy między polami i metodami normalnymi i statycznymi, dzięki czemu błędy typu "Cannot make a static reference to the non-static method" przestaną być straszne.

Dodaj do **klasy pobocznej** cztery nowe pola.

```
16   public static int countStatic_ = 0;
17   public int countNonStatic_ = 0;
18
19   public static int idStatic_ = 0;
20   public int idNonStatic_ = 0;
```

A następnie wykorzystaj obie pary pól w konstruktorach. Celem pól *count** jest przechowywanie liczby wszystkich stworzonych obiektów klasy pochodnej w trakcie trwania całego programu. Celem pól *id** jest przypisanie do każdego z obiektów unikalnego numeru ID. Zaaktualizuj klasę wypisującą obiekt na ekran.
Wybierz które pola spełniają swoją funkcję, a te niespełniające wykomentuj. Jeśli pozostawiłeś jakieś pole statyczne, wypisz je na ekran w **klasie głównej** BEZ użycia obiektu **klasy pobocznej**.

Zadanie 6.2(2.0):
Celem zadania jest stworzenie ***klasy dodatkowej*** *która będzie w kompozycji z* ***klasą poboczną****. Kompozycja, czyli stworzenie zależności gdzie jedna klasa nie może istnieć bez drugiej.*

Stwórz **klasę dodatkową**, która następnie będzie polem **klasy pobocznej** (na przykład data przydatności do spożycia, która będzie nierozłączna z przedmiotami przechowywanymi w sklepie). Dorób konstuktor/konstruktory/metody według uznania. Zaktualizuj **klasę poboczną** oraz **klasę główną** tak, aby finalnie **klasa poboczna** wypisywała również informację o **klasie dodatkowej**.

Zadanie 6.3(1.0):

Zaktualizuj cały napisany do tej pory kod tak, aby:

- dostęp do pól **klasy pobocznej** miała jedynie ona - **klasa główna** nie może ich odczytać ani modyfikować.
- **klasa główna** mogła jedynie wyświetlać pola statyczne, nie może ich modyfikować
- **klasa poboczna** nie była widocza z perspektywy **klasy głównej**.