PPJ

ćwiczenie 12

17 grudnia 2023

I. Przygotujmy dwie klasy: Product reprezentującą produkt oraz ShoppingCart reprezentującą koszyk zakupowy. Każdy produkt może być przypisany do konkretnego koszyka. Twoim zadaniem jest:

1. Utwórzyć klasę Product posiadającą pola:
   - name (typu String) - przechowujące nazwę produktu,
   - price (typu double) - przechowujące cenę produktu,
   - shoppingCart (typu ShoppingCart) - przechowujące informację o koszyku, do którego przypisany jest produkt.
2. Zaimplementuj w klasie Product metodę addToShoppingCart, która umożliwia przypisanie danego produktu do konkretnego koszyka
3. Utwórz klasę ShoppingCart posiadającą pola:
   - customerName (typu String) - przechowujące nazwę klienta,
   - products (typu Product[]) - przechowujące tablicę produktów,
   - productCount (typu int) - przechowujące liczbę produktów w koszyku.
4. Zaimplementuj w klasie ShoppingCart metodę addProduct, która pozwoli na dodanie produktu do koszyka. Dodatkowo, zaimplementuj metodę displayProducts, która wyświetli nazwy i ceny wszystkich produktów w koszyku.
5. W metodzie main utwórz obiekty klas ShoppingCart i Product, dodaj produkty do koszyka, a następnie wyświetl zawartość koszyka.

II. Przyjmij że jeden balon wykonany z lateksu o wielkości 9 cali wypełniony helem o pojemności $0,007m^3$ jest w stanie udźwignąć 6 gramowy ciężar. Przygotuj klasę `Balloon` zawierającą:

- domyślny konstruktor - który wylosuje ilość helu jakim wypełniony będzie balon przyjmując że ilość helu może się wahać od $0,005m^3$ do $0,009m^3$;
- metoda `getLoad` - wyliczy udźwig balonu, na podstawie założonych danych.

Utwórz klasę `Donkey` definiującą prywatne pola: `double mass` (wyrażoną w kilogramach) i jednowymiarową tablicę typu `Balloon`. Następnie przygotuj metody:

- `void addBaloon(Baloon)`, która przechowa dodany balon w tablicy;
- `boolean isFlying()`, która zwróci wartość `true` gdy jest wystarczająco balonów aby podnieść ciężar osła i `false` w przeciwnym przypadku.

Napisz program, który oderwie od ziemi osła i gdy się to już stanie osioł radośnie wykrzyknie "Ja latam!!!"

III. Zadbaj aby tablica balonów zdefiniowana w poprzednim zadaniu dopasowywała swój rozmiar do ilości balonów.

IV. Utwórz klasę `Word` zawierającą pole tablicę typu `char[]` i pole typu `int`. Bezparametrowy konstruktor tworzący wyraz zainicjuje pola odpowiednio: tablicą o rozmiarze 100 elementów i wartością 0. Metoda `addChar(char)` pozwoli na dopisywanie kolejnych znaków do ciągu. Metoda `show()` wyświetli ciąg, a metoda `length()` zwróci ilość znaków w ciągu.

V. Zadeklaruj klasę `Dzem` zawierającą prywatne pola:

- `String smak`,
- `double waga`

Utwórz konstruktory przyjmujące jako parametry zmienne:

- `(String smak, double waga)`,
- `(double waga)`,
- `(String smak)`.

Dostarczone parametry konstruktorów powinny zainicjalizować pola klasy. W przypadku gdy konstruktor nie dostarcza niezbędnego parametru należy przyjąć, że pole `smak` musi przyjąć wartość `‘‘No name’’`, natomiast `waga`: `100.0`.

Przedstaw wykorzystanie wszystkich konstruktorów tworząc obiekty reprezentujące trzy rodzaje dżemów.

Uwaga: Tylko konstruktor z dwoma parametrami może przypisać wartości polom klasy, skorzystaj z słowa kluczowego `this`.

VI. Przygotuj klasę `Ball` z polem obiektowym `double radius` i polem klasowym `int ballCounter`. Następnie przygotuj metody klasowe:

- `makeBall` – tworzącą i zwracającą obiekt klasy Ball o losowej wielkości pomiędzy 10 a 20;
- `showCouter` – wyświetlającą informację ile obiektów klasy `Ball` zostało utworzonych.

Przedstaw wykorzystanie powyższych metod tworząc określoną ilość obiektów `Ball` i zweryfikuj czy twoje estymacje są zgodne z wskazaniami zmiennej ballCounter.

VII. Przygotuj klasę `Osoba` definiującą pola:

- `String` imie,
- `int` rokUrodzenia.

Klasa będzie również definiować:

- dwuargumentowy konstruktor, inicjujący pola klasy;
- jednoargumentowy konstruktor, przyjmujący jako parametr `String` imie, natomiast jako pole `rokUrodzenia` przypisujący wartość 1990;
- metodę `zwrocImie()` zwracającą wartość pola `imie`;
- metodę `zwrocWiek()` zwracającą wiek osoby;
- statyczną metodę `zwrocStarszaOsobe` przyjmującą w liście argumentów dwa obiekty klasy `Osoba` i zwracającą starszą osobę;
- statyczną metodę `zwrocNajstarszaOsobe` przyjmującą jako argument tablicę obiektów klasy `Osoba` i zwracającą najstarszą osobę.

VIII. Utwórz klasę `Prostokąt` i dziedziczącą po niej klasę `Prostopadloscian`. Operację powtórz dla klas `Trojkat` opisującą figurę trójkąta równobocznego i dziedziczące po niej `Ostroslup` i `Graniastoslup`. Zadbaj aby:

- klasy zawierały wszystkie niezbędne pola;
- wszystkie pola byly poprzedzone specyfikatorem `private`;
- konstruktory inicjowały wartości pól zarówno na podstawie dostarczonych wartości liczbowych jak i obiektu klasy bazowej;
- wszystkie klasy figur posiadały metodę wyświetlającą pole powierzchni;
- wszystkie klasy brył posiadały metody wyświetlające pole powierzchni oraz objętość bryły;

IX. Utwórz klasę `Osoba` z prywatnym polem `String imie` i dziedziczącą po niej klasę `Spawacz` z polam `stazpracy`. Utwórz w obu klasach metodę `String wyswietl()`, która wykorzystując słowo kluczowe `super` zwróci ciąg znaków zawierający wszystkie informacje zawarte w obiekcie tej klasy.