I. Stwórz klasę Monster. Klasa powinna posiadać następujące pola:

- name – opisującą nazwę danego potwora
- health – opisującą poziom życia danego potwora
- strength – opisującą poziom siły danego potwora

Utwórz obiekt klasy Monster, a następnie wypisz jego parametry.

II. Dodaj do klasy Monster konstruktor, którego zadaniem będzie przypisanie wartości do pól klasy. Dodaj także bezargumentowy konstruktor, który będzie uzupełniał pola domyślnymi wartościami. Następnie stwórz dwa obiekty klasy Monster wykorzystując różne konstruktory, po czym wypisz informacje na temat obiektów.

III. Dodaj do klasy Monster następujące metody:

- getHealth() – zwracającą informację o poziomie życia danego stwora
- takeDamage(int) – metodę przyjmującą jako argument wartość, która powinna zostać odjęta od poziomu życia
- getPower() – zwracającą informację o sile (strength) danego potwora
- isAlive() – zwracającą informację (boolean) czy poziom życia danego potwora jest dodatni
- show() – wypisującą na ekranie informacje o danym obiekcie

Następnie przetestuj działanie metod na dwóch różnych obiektach.

IV. Stwórz program, którego zadaniem będzie wygenerowanie dwóch obiektów klasy Monster. Wartości podane do konstruktora powinny zostać wylosowane. Następnie dodaj statyczną metodę arena(Monster, Monster), która:

- będzie przyjmowała dwa obiekty klasy monster jako argumenty
- następnie będzie przeprowadzała symulację walki pomiędzy dwoma Monsterami
- powinna zwrócić obiekt typu Monster, który daną walkę wygra

Symulacja walki powinna odbywać się w pętli dopóki nie zostanie wyłoniony zwycięzca. W każdej rundzie walki zadajemy obrażenia przeciwnemu obiektowi równe sile przeciwnika. Po każdej rundzie należy sprawdzić czy oba obiekty typu Monster mają poziom życia powyżej zera, w przeciwnym przypadku metoda powinna zwrócić obiekt zwycięzcy.

V. Napisz program którego zadaniem będzie przeprowadzenie turnieju. Wygeneruj 12 obiektów typu Monster, a następnie przechowaj je w tablicy. Następnie przeprowadź symulację walk pomiędzy monsterami (każdy z każdym), pamiętając ilość walk wygranych dla danego obiektu. Po zakończeniu symulacji wypisz posortowane (ze względu na ilość wygranych) informacje o uczestnikach turnieju.

VI. Wykonaj poniższe kroki:

- Utwórz klasę Sala, posiadającą pola: int numer, int pojemnosc, Student[] studenci.
- Utwórz klasę Uczelnia, z polami: String nazwa, int iloscStudentow, Sala[] listaSal.
- Dodaj metodę statyczną dodajSale dodającą nową salę do tablic sal uczelni.
- Utwórz klasę Student, posiadającą pola: int id, String imie, String nazwisko. Id powinno być unikalne i przydzielane automatycznie - nie podawane z konstruktora.

- Należy dodać metodę nie statyczną, dodającą studenta do pierwszej wolnej sali na uczelni.

Dodatkowe wyzwania:

- Stwórz dodatkowe pole w klasie Sala i dodaj 2 metody modyfikujące/sprawdzające wartość pola.
- Stwórz dodatkowe pole w klasie Uczelnia dodaj 2 metody modyfikujące/sprawdzające wartość pola.
- Stwórz dodatkowe pole w klasie Student i dodaj 2 metody modyfikujące/sprawdzające wartość pola.