W ramach zadania przyjmujemy, że wszystkie kolekcje zdefiniowane w treści przechowują obiekty klasy Hero. Twoje zadanie polega na zaprojektowaniu architektury obiektowej pozwalającej na łatwe przeglądanie elementów kolekcji.

Klasa Hero reprezentuje postacie z pewnej gry RPG. Zawiera ona podstawowe dane dotyczące postaci jak „klasę postaci” (np. wojownik, mag), poziom doświadczenia czy aktualny stan zdrowia oraz punktów many.

Różne obszary w grze przechowują informacje na temat swoich postaci w różnego typu kolekcjach:

- Klasa CustomLinkedList reprezentuje listę wiązaną z ogonem i głową. Przyjmij, że przeglądanie kolekcji odbywa się w kolejności odwrotnej do naturalnej.
- Klasa ReversedArrayList listę implementowaną na tablicy. Lista ta zwraca obiekty w odwrotnej kolejności (indeks 0 oznacza ostatni element, 1 przedostatni itd.) w przypadku losowego dostępu, przyjmij, że w przypadku przeglądania kolekcji lista zwraca elementy w kolejności naturalnej (w kolejności dodawania).

Kolekcje postaci zawartych w poszczególnych strefach na mapie gry pobierane są przez przygotowany kod, który inicjalizuje obiekty odpowiadające kolekcjom postaci przebywających w tych obszarach. W ramach rozwiązania nie wolno zmieniać kodu inicjalizacji tych kolekcji.

Twoje zadanie:

1. Wyświetl wszystkie postacie zawarte w poszczególnych obszarach.
2. Przygotuj listę możliwych par postaci do turnieju PvP. Kolejność wylosowanych parach jest istotna, tzn. para (hero1, hero2) nie jest tym samym co para (hero2, hero1). W ramach tego punktu możesz założyć, że gracz może dostać wolny los (automatycznie awansować) jeżeli wylosuje sam siebie, dlatego takie pary też są istotne (przyjmij, że istnieje możliwość wygenerowania dwóch par (hero1, hero1) w ramach całego losowania).
3. Przygotuj rozwiązanie, które pozwoli w łatwy sposób przeglądać postacie, których „klasa postaci” (atrybut heroClass) ma z góry określoną wartość. Załóż, że rozwiązanie powinno być łatwo skalowalne, tzn może być wykorzystywane dla większej liczby rodzajów kolekcji, oraz funkcji (nie tylko te z pktu 1 i 2). Rozwiązanie powinno uwzględniać, że liczba postaci w grze może być bardzo duża, natomiast operacje wykonywane na odfiltrowanych listach wykonywane są względnie rzadko.
4. Wykorzystaj rozwiązanie z pktu 3 do powtórzenia zadań z pktu 1 i 2 dla odfiltrowanych kolekcji.

Uwaga: w ramach rozwiązania przyjmujemy, że nigdy nie może dojść do sytuacji, że kolekcja jest modyfikowana w trakcie przeglądania.