Przykładowe kolokwium #2 - Zestaw Z13

Ostatnia aktualizacja pliku: 09.01.2024 18:25.

Imię i nazwisko, numer albumu .....

Informacje wstępne

- Łącznie do zdobycia max **60** punktów. Próg zaliczenia: 25 pkt (bez innych punktów).
- **Kolokwium należy wykonać na komputerach zamontowanych na stałe w pracowniach.**
- Student przesyłając rozwiązania oświadcza, że rozwiązał je samodzielnie.
- W trakcie kolokwium nie można korzystać z żadnych materiałów pomocniczych w żadnej formie. Wszelkie kody powinny być napisane manualnie bez wspomagania się dodatkami automatycznie generującymi kod (np. Copilot, chat GPT itp.).
- Publikowanie poleceń i rozwiązań w internecie jest zabronione do czasu napisania kolokwium przez wszystkie grupy ćw.
- Należy zwracać uwagę na właściwe umieszczenie kodu (luzem lub w pakiecie).
- Kod musi się kompilować, aby był sprawdzany.
- Należy oddzielać klasę z definicjami od klasy testującej (z main) zgodnie z poleceniami.
- Jeśli w poleceniu nie jest podany typ zmiennej, można go wybrać dowolnie.
- Jeśli w danej metodzie nie ma sprecyzowanej „walidacji", to można ją pominąć.
- Metody nie powinny wykonywać nadmiarowych, nielogicznych czynności.
- Poza zmiennymi/polami w klasie wymienionym w polecaniach zabronione jest tworzenie innych pól w klasie. Stworzenie dodatkowych metod jest dopuszczalne, ale nie należy tego nadużywać.
- Jeśli w poleceniu nie są sprecyzowane modyfikatory dostępu, należy dostępować zgodnie z zasadami hermetyzacji.
- **W rozwiązaniach należy uwzględniać dobre praktyki omawiane na wykładzie i ćwiczeniach, o ile polecenie nie mówi coś innego.**
- Rozwiązania (projekt z IntelliJ) należy w całości spakować jako archiwum zip. Następnie ustawić nazwę. Rozwiązania należy umieścić na pendrive przekazanym przez prowadzącego kolokwium.
- **Nazwa archiwum powinna być wg schematu NUMERZESTAWU_NUMERALBUMU.zip gdzie numer zestawu znajduje się na górze kartki z poleceniami. np. A23_123456.zip.**
- Archiwum powinno być bez hasła.
- Kod zakomentowany nie będzie sprawdzany.
- Zawartość pendrive będzie pusta. Udostępniony będzie tylko w celu zgrania rozwiązań. Umieszczenie poleceń na pendrive powinno odbyć się w czasie kolokwium. Rozwiązania po czasie mogą nie być sprawdzane.
- Jeśli w poleceniu pojawia się informacja o konieczności zachowania formatowania napisów (np. wielkość znaków, znaki interpunkcyjne), to należy to bezwzględnie wykonać.
- Podpunkty będą oceniane kaskadowo — wykonanie ich bez wykonania wcześniejszych podpunktów może oznaczać zero punktów.
- O ile nie zaznaczono w poleceniu inaczej, każdą z metod należy wywołać co najmniej jeden raz (może być bardzo trywialnie). Warto zwrócić uwagę, że samo tworzenie obiektów w każdym zdefiniowanym samodzielnie typie nie jest wymagane (chyba że polecenie tego wymaga).
- Należy zachowywać kolejność argumentów w konstruktorach i metodach. Należy dążyć do tego, że nazwy argumentów metod powinny pokrywać się z nazwami pól w klasie, gdzie to ma sens.
- Warto zwracać uwagę na typ zwracany metod — jeśli metoda ma „coś" zwrócić, będzie to wskazane w poleceniu.
- Po kartkach z poleceniami można pisać i traktować jako brudnopis.

Zadanie 1. (15pkt max.)

A. Wykonaj poniższe czynności:

- Stwórz klasę `Song` w pakiecie `music`, która powinna zawierać trzy pola:
  - `title`: typu `String`, reprezentującego tytuł piosenki.
  - `artist`: typu `String`, reprezentującego artystę wykonującego piosenkę.
  - `duration`: typu `int`, reprezentującego czas trwania piosenki w sekundach.
- Zaimplementuj dwie klasy porównujące, które implementują generyczny interfejs `Comparator<Song>`:
  - `DurationComparator`: porównuje obiekty klasy `Song` według czasu trwania (`duration`), od najkrótszego do najdłuższego utworu.
  - `ArtistTitleComparator`: porównuje obiekty klasy `Song` najpierw według artysty (`artist`), a w przypadku równości - według tytułu (`title`). W obu przypadkach porządek powinien być odwrotny do naturalnego.

B. Wykonaj poniższe czynności:

- W klasie `TestSong` w tym samym pakiecie w metodzie `main`:
  - Utwórz i posortuj tablicę obiektów `Song` najpierw według długości utworu (używając `DurationComparator`).
  - W przypadku, gdy dwa utwory mają tę samą długość, zastosuj `ArtistTitleComparator`, aby ustalić kolejność.
  - Po zakończeniu sortowania wyświetl posortowaną listę, aby sprawdzić, czy sortowanie przebiegło prawidłowo i czy kolejność utworów jest zgodna z założeniami.

Zadanie 2. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `sports`.
- Stwórz klasę `Stadium` (Stadion) z polami: `name` (nazwa), `location` (lokalizacja), `capacity` (pojemność), `homeTeam` (drużyna gospodarzy). Dodaj konstruktor parametryczny, gettery, settery oraz metody `toString`, `equals` i `hashCode`.
- Stwórz klasę `AthleteProfile` (Profil Sportowca). Klasa `AthleteProfile` powinna mieć pola: `athleteName` (nazwa sportowca), `sport` (dyscyplina sportowa), `stadium` typu `Stadium` (stadion). Dodaj konstruktor parametryczny, gettery, settery oraz metody `toString`, `equals` i `hashCode`.

Zadanie 3. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `optimize`.
- Napisz metodę generyczną `findMax`, która przyjmuje trzy parametry typu generycznego `T` i zwraca największy z nich. Zakładamy, że typ `T` implementuje interfejs `Comparable<T>`. Metoda powinna porównywać trzy argumenty i zwracać ten o największej wartości.

Zadanie 4. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `printing`.
- Stwórz metodę statyczną generyczną w klasie `Utilities` o nazwie `printEverySecond`. Metoda ta powinna przyjmować jako argument obiekt implementujący interfejs `Collection<E>`, gdzie `E` jest typem generycznym.
- Zadaniem metody `printEverySecond` jest wyświetlenie na konsoli co drugiego elementu z przekazanej kolekcji. Należy pominąć pierwszy element, a następnie wyświetlić drugi, pominąć trzeci, wyświetlić czwarty, i tak dalej.

- W klasie `TestUtilities`, w metodzie `main`, utwórz kilka kolekcji różnych typów (na przykład `List<String>`, `Set<Integer>`), wypełnij je elementami, a następnie użyj metody `printEverySecond` do wyświetlenia co drugiego elementu każdej z kolekcji.