# Przykładowy egzamin - Zestaw E13

Ostatnia aktualizacja pliku: 07.01.2024 23:13.

Imię i nazwisko, numer albumu .....

**Informacje wstępne**

- Punktacja: 46-50 pkt - bdb(5,0); 41-45 pkt - db+(4,5); 36-40 pkt - db(4,0); 31-35 pkt - dst+(3,5); 26-30 pkt - dst(3,0); 0-25 pkt - ndst (2,0).
- **Egzamin należy wykonać na komputerach zamontowanych na stałe w pracowniach.**
- Student przesyłając rozwiązania oświadcza, że rozwiązał je samodzielnie.
- W trakcie egzaminu 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 egzaminu przez wszystkich.
- Należy zwracać uwagę na właściwe umieszczenie kodu (luzem lub w pakiecie). Kod musi się kompilować, aby był sprawdzany. Kod zakomentowany nie będzie 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 (o ile polecenie tego nie zabrania), ale nie należy tego nadużywać.
- 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.
- Jeśli w poleceniu nie są sprecyzowane modyfikatory dostępu, należy dostępować zgodnie z zasadami hermetyzacji.
- Jeśli w poleceniu pojawia się informacja o konieczności zachowania formatowania napisów (np. wielkość znaków, znaki interpunkcyjne), to należy to bezwzględnie wykonać.
- **W rozwiązaniach należy uwzględniać dobre praktyki omawiane na wykładzie, 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 egzamin. Rozwiązania niespakowane jako zip nie będą sprawdzane. Archiwum powinno być bez hasła.
- **Nazwa archiwum powinna być wg schematu NUMERZESTAWU_NUMERALBUMU.zip gdzie numer zestawu znajduje się na górze kartki z poleceniami. np. A23_123456.zip.**
- Zawartość pendrive będzie pusta. Umieszczenie poleceń na pendrive powinno odbyć się w czasie egzaminie. Rozwiązania po czasie mogą nie być sprawdzane.
- Podpunkty będą oceniane kaskadowo oraz wykładniczo — wykonanie ich bez wykonania wcześniejszych podpunktów może oznaczać zero punktów. Koniec polecenia ma największą wagę w ocenię danego zadania.
- 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).
- Po kartkach z poleceniami można pisać i traktować jako brudnopis.

**Zadanie 1. (12pkt 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 tablicę, aby sprawdzić, czy sortowanie przebiegło prawidłowo i czy kolejność utworów jest zgodna z założeniami.

**Zadanie 2. (13pkt max.)**

- Poniższe czynności wykonaj w pakiecie `notmod`.
- Stwórz finalną klasę `InmutablePoint` z prywatnymi finalnymi polami: `x`, `y`, `z` (współrzędne punktu).
- Dodaj konstruktor parametryczny do inicjalizacji wszystkich pól.
- Dodaj publiczne metody `getX`, `getY`, `getZ` do pobierania wartości pól, ale nie dodawaj żadnych metod umożliwiających ich modyfikację.
- Zaimplementuj metody `toString`, `equals` i `hashCode`.

**Zadanie 3. (12pkt max.)**

- Poniższe czynności wykonaj w pakiecie `utilities`.
- Stwórz metodę generyczną `findMin`, która przyjmuje trzy parametry typu generycznego `T` i zwraca najmniejszy z nich. Typ `T` powinien implementować interfejs `Comparable<T>`. Metoda powinna porównywać trzy argumenty i zwracać ten o najmniejszej wartości.

**Zadanie 4. (13pkt max.)**

- Poniższe czynności wykonaj w pakiecie `algorithm`.
- Napisz klasę generyczną `ItemManager<T>`, która będzie zarządzać elementami typu `T`. Klasa powinna zawierać prywatną listę elementów oraz metody `void addItem(T item)`, dodającą nowy element do listy, `T getItem(int index)`, zwracającą element na danej pozycji oraz `int getItemCount()`, zwracającą liczbę przechowywanych elementów.