Przykładowy egzamin - Zestaw E17

Ostatnia aktualizacja pliku: 22.01.2024 07:34.

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. (13pkt max.)

A. Klasa `BusStation` w pakiecie `transport` z prywatnymi polami:

- `name`: typu `String`, reprezentujący nazwę dworca autobusowego.
- `city`: typu `String`, reprezentujący miasto, w którym znajduje się dworzec.
- `buses`: typu `ArrayList<String>`, lista przechowująca nazwy autobusów.

B. Metody w klasie `BusStation`:

- Metoda `addBus(String bus)`: dodaje autobus do listy `buses`.
- Metoda `removeBus(String bus)`: usuwa autobus z listy `buses`.
- Konstruktory, gettery, settery, `toString()`, `equals()` i `hashCode()`.
- Pamiętaj o odpowiedniej kopii dla pola będącego listą tablicową.

C. Klasa `IntercityBusStation`, dziedzicząca po `BusStation` w tym samym pakiecie, z dodatkowym prywatnym polem `numberOfPlatforms`: typu `int`, reprezentujący liczbę peronów na dworcu.

D. Metody w klasie `IntercityBusStation`:

- Konstruktory, gettery i settery dla `numberOfPlatforms`.
- Nadpisane metody `toString()`, `equals()` i `hashCode()`.

E. Napisz klasę testującą `TestBusStation` w tym samym pakiecie:

- W metodzie `main` utwórz obiekty klasy `BusStation` i `IntercityBusStation`.
- Testuj działanie metod dodawania i usuwania autobusów.
- Wyświetl informacje o obu dworcach, aby sprawdzić poprawność działania metod.

Zadanie 2. (13pkt max.)

- Wykonaj poniższe czynności w pakiecie `university`.
- Napisz rekord `Student` z polami `id` (typu `int`), `name` (typu `String`) oraz `averageGrade` (typu `double`). Zaimplementuj dwie klasy implementujące generyczny interfejs `Comparator`: `AverageGradeComparator` do porównywania obiektów po polu `averageGrade` (od najwyższej do najniższej średniej ocen) oraz `IdComparator` do porównywania obiektów po polu `id` (od najniższego do najwyższego identyfikatora). Stwórz listę tablicową 5 obiektów typu `Student` i posortuj ją zgodnie z oboma kryteriami (najpierw po średniej ocen, a następnie przy równości po identyfikatorze).

Zadanie 3. (12pkt max.)

- Poniższe czynności wykonaj w pakiecie `finding`.
- Utwórz statyczną metodę generyczną `findFirstNonNull`. Metoda ta przyjmuje tablicę obiektów tego samego typu generycznego `T` i zwraca pierwszy element z listy, który nie jest `null`. Jeśli wszystkie elementy są null, metoda zwraca `null`. Stwórz przypadek testowy.

Zadanie 4. (12pkt max.)

W pakiecie `algorithm`, zaimplementuj statyczną metodę `mapToString(HashMap<K, V> map)`, która zwraca `String` reprezentujący wszystkie pary klucz-wartość w podanej mapie w formacie `"klucz: wartość"`. Każda para powinna być oddzielona przecinkiem i spacją. Metoda ta powinna być odpowiednia dla map przechowujących dowolny typ kluczy i wartości. Stwórz przypadek testowy na bazie klasy klucza `Person` z polem `name`. Przyjmij, że dwie osoby są równe jeśli mają te same imię.