Przykładowy egzamin - Zestaw E14

Ostatnia aktualizacja pliku: 14.01.2024 19:24.

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. Utwórz rekord `StudentRecord` w pakiecie `student`, który powinien zawierać trzy pola:

- `name`: typu `String`, reprezentującego imię studenta.
- `id`: typu `String`, reprezentującego identyfikator studenta.
- `gpa`: typu `double`, reprezentującego średnią ocen studenta.

B. Dodaj do rekordu `StudentRecord`:

- Kompaktowy konstruktor, który weryfikuje, czy średnia ocen (`gpa`) jest w przedziale od 0.0 do 4.0. Jeśli `gpa` jest poza tym zakresem, konstruktor powinien rzucać wyjątek `IllegalArgumentException`.
- Metodę `isHonorStudent`, która zwraca `true`, jeśli średnia ocen (`gpa`) studenta jest wyższa lub równa 3.5, i `false` w przeciwnym przypadku.
- Metodę `printDetails`, która wyświetla informacje o studencie, włączając w to jego imię, identyfikator i średnią ocen.

C. W pakiecie `student`, utwórz klasę testową `TestStudentRecord` z metodą `main`, w której:

- Utwórz dwa obiekty typu `StudentRecord` z różnymi danymi.
- Wywołaj metodę `printDetails` na każdym z obiektów, aby wyświetlić ich szczegóły.
- Sprawdź, którzy studenci są studentami z wyróżnieniem, używając metody `isHonorStudent`.

Zadanie 2. (13pkt max.)

- Poniższe czynności wykonaj w pakiecie `university`.
- Zdefiniuj rekord `Student`, który zawiera pola: `name` (typu `String`), `averageGrade` (typu `double`) i `yearOfStudy` (typu `int`).
- Zaimplementuj interfejs `Comparable<Student>` w taki sposób, aby instancje rekordu `Student` były sortowane najpierw malejąco według średniej ocen (`averageGrade`), a w przypadku tej samej średniej ocen, rosnąco według roku studiów (`yearOfStudy`).
- Stwórz tablicę 4 instancji rekordu `Student` i posortuj ją zgodnie ze zdefiniowanym kryterium.

Zadanie 3. (12pkt max.)

- Poniższe czynności wykonaj w pakiecie `utilities`.
- Stwórz statyczną metodę generyczną `countLessThanOrEqual`, która przyjmuje dwa argumenty:
  - Tablicę elementów typu `T`.
  - Dowolny obiekt typu `T`.

Typ `T` musi implementować interfejs `Comparable<T>`. Metoda powinna zwracać liczbę elementów w tablicy, które są mniejsze lub równe drugiemu argumentowi metody.

Dodatkowo, utwórz klasę `Person` z polami `name` (typu `String`) i `age` (typu `int`). Klasa `Person` powinna implementować interfejs `Comparable<Person>` w oparciu o pole `age`. Przetestuj metodę `countLessThanOrEqual` na tablicy obiektów `Person`.

Zadanie 4. (13pkt max.)

- Poniższe czynności wykonaj w pakiecie `algorithm`.
- Napisz statyczną metodę generyczną `countUniqueKeys`, która przyjmuje jako argument `HashMap<K, V>` i zwraca liczbę unikalnych kluczy w tej mapie. Metoda powinna być zdolna do obsługi `HashMap` z różnymi typami kluczy. Stwóz przypadek testowy dla metody.