Przykładowy egzamin - Zestaw E12

Ostatnia aktualizacja pliku: 19.01.2024 06:37.

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. Wykonaj poniższe czynności:

- Stwórz klasę `Athlete` w pakiecie `sports`, która powinna zawierać trzy pola:
  - `name`: typu `String`, reprezentującego imię i nazwisko sportowca.
  - `nationality`: typu `String`, reprezentującego narodowość sportowca.
  - `records`: tablica typu `double`, reprezentująca rekordy sportowca w różnych dyscyplinach (na przykład czas w biegach, odległość w skokach).
- Zaimplementuj dwie klasy porównujące, które implementują generyczny interfejs `Comparator<Athlete>`:
  - `RecordComparator`: porównuje obiekty klasy `Athlete` według ich najlepszego rekordu (`records`), od najniższego do najwyższego.
  - `NationalityNameComparator`: porównuje obiekty klasy `Athlete` najpierw według narodowości (`nationality`), a w przypadku równości - według imienia i nazwiska (`name`). W obu przypadkach porządek powinien być odwrotny do naturalnego.

B. Wykonaj poniższe czynności:

- W klasie `TestAthlete` w tym samym pakiecie w metodzie `main`:
  - Utwórz i posortuj tablicę obiektów `Athlete` najpierw według ich najlepszego rekordu (używając `RecordComparator`).
  - W przypadku, gdy dwaj sportowcy mają ten sam najlepszy rekord, zastosuj `NationalityNameComparator`, aby ustalić kolejność.
  - Po zakończeniu sortowania wyświetl posortowaną listę, aby sprawdzić, czy sortowanie przebiegło prawidłowo i czy kolejność sportowców jest zgodna z założeniami.

Zadanie 2. (13pkt max.)

- Poniższe czynności wykonaj w pakiecie `memory`.
- Stwórz interfejs `MemoryManager` z:
  - Metodą abstrakcyjną `allocateMemory(int size)`.
  - Metodą domyślną `freeMemory()` wyświetlającą informację "Memory Freed".
  - Metodą statyczną `getMemoryType()` zwracającą `String` "Memory Type".
- Stwórz klasy `RAMManager` i `DiskManager`, które implementują `MemoryManager`. `allocateMemory(int size)` w `RAMManager` powinno wyświetlać "Allocating RAM Memory", a w `DiskManager` - "Allocating Disk Space". Stwórz klasę testującą `MemoryTester`. Utwórz obiekty `RAMManager` i `DiskManager`, wywołaj dla nich `allocateMemory(int size)` i `freeMemory()`, oraz statycznie `MemoryManager.getMemoryType()`.

Zadanie 3. (12pkt max.)

- Poniższe czynności wykonaj w pakiecie `utilities`.
- Napisz statyczną metodę generyczną `appendFromEnd`.
- Metoda `appendFromEnd` dopisuje wszystkie elementy z jednej tablicy typu ArrayList do drugiej od końca.
- W implementacji tej metody należy użyć symbolu wieloznacznego `<? super E>`.
- Przetestuj działanie metody na dowolnym przykładzie.

Zadanie 4. (12pkt max.)

- Poniższe czynności wykonaj w pakiecie `algorithm`.
- Utwórz statyczną metodę generyczną `compareThree`, która przyjmuje trzy parametry typu generycznego `T` i zwraca `true`, jeśli wszystkie trzy elementy są równe ze sobą (zgodnie z `equals`), oraz `false` w przeciwnym przypadku. Typ `T` nie musi implementować żadnych interfejsów.
- Stwórz klasę `Student` z polami `name` (typu `String`) i `grade` (typu `float`). Nadpisz metodę `equals` i `hashCode` tak, aby dwa obiekty były równe, gdy są takie same wszystkie pola. Przetestuj metodę generyczną na obiektach typu `Student`.