Przykładowy egzamin - Zestaw E11

Ostatnia aktualizacja pliku: 22.01.2024 07:36.

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 `Hospital` (pol. Szpital), która powinna być umieszczona w pakiecie `healthcare`.
- Klasa powinna posiadać prywatne pola:
  - `name`, (nazwa szpitala), typ `String`.
  - `capacity`, (pojemność szpitala, czyli liczba łóżek), typ `double`.
- Napisz dwuargumentowy konstruktor tej klasy. Kolejność argumentów powinna być taka sama jak wyżej. Zapewnij niezależnie warunki sprawdzające poprawność:
  - Nazwa nie może być nullem - jeśli jest nullem, to zamień ją na napis pusty (`""`).
  - Pojemność musi być liczbą dodatnią, w przeciwnym wypadku ustaw ją na 50.0.
- Napisz metody typu `getter` i `setter` dla wszystkich pól. Pamiętaj, aby sprawdzić kryteria podane w konstruktorze dokładnie tak samo.
- Nadpisz metodę `toString` tak, aby zwracała napis z reprezentacją obiektu. Na początku powinna być nazwa klasy (pobrana "z systemu") - potem wartości wszystkich pól. Zwróć uwagę na wielkość znaków i znaki interpunkcyjne. Schemat:

```
[NazwaKlasy]: Name: [name]. Capacity: [capacity].
```

lub jeśli nazwa nie jest ustalona (jest pustym napisem):

```
[NazwaKlasy]:  Capacity: [capacity].
```

- Nadpisz metodę `equals`. Dwa szpitale są sobie "równe" wtedy i tylko wtedy, gdy wszystkie ich pola są identyczne. Nadpisz metodę `hashCode()`, która generuje kod hash dla odpowiedniego obiektu. Metoda ta powinna być zgodna z metodą `equals()`.

B. Wykonaj poniższe czynności:

- Klasa `Clinic` (pol. Klinika) powinna być umieszczona w pakiecie `healthcare` w innym pliku niż klasa `Hospital`.
- Klasa `Clinic` dziedziczy po klasie `Hospital`. Klasa powinna posiadać dodatkowe pole `rating`, typu `double` (ocena kliniki).
- Napisz trzyargumentowy konstruktor tej klasy. Kolejność argumentów powinna być taka sama jak wyżej (najpierw z klasy bazowej, potem pochodnej). Zapewnij niezależnie warunki sprawdzające poprawność dodatkowo, że ocena musi być liczbą z zakresu od 0.0 do 5.0 (włącznie) - w przeciwnym wypadku ustaw ją jako 3.0.
- Napisz metody typu `getter` i `setter` tylko dla pola z klasy pochodnej. Pamiętaj, aby sprawdzić kryteria podane w konstruktorze dokładnie tak samo.
- Nadpisz metodę `toString` tak, aby zwracała napis z reprezentacją obiektu. Na początku powinna być nazwa klasy ("z systemu") - potem wartości wszystkich pól. Zwróć uwagę na łamanie linii, wielkość znaków i znaki interpunkcyjne. Schemat:

```
[NazwaKlasy]: Name: [name]. Capacity: [capacity].
Rating: [rating].
```

lub jeśli nazwa nie jest ustalona (jest pustym napisem):

```
[NazwaKlasy]: Capacity: [capacity].
Rating: [rating].
```

- Nadpisz metodę `equals`. Dwie kliniki są sobie "równe" wtedy i tylko wtedy, gdy mają takie same wszystkie pola (zarówno z klasy bazowej jak i pochodnej). Nadpisz metodę `hashCode()`, która generuje kod hash dla odpowiedniego obiektu. Metoda ta powinna być zgodna z metodą `equals()`.
- Zapewnij zgodność pozostałych metod z metodami z klasy bazowej.

C. Wykonaj poniższe czynności:

- Stwórz klasę `TestHospital` (klasa testująca) umieść w pakiecie tym samym co klasy z punktu w A i B, ale w innym pliku. Umieść w tej klasie tylko metodę `main`. W metodzie `main` stwórz 5 obiektów w typach definiowanych w zadaniu A i B. Następnie sprawdź poprawność działania metody `equals` na co najmniej 5 różnych sposobów.

Zadanie 2. (13pkt max.)

- Poniższe czynności wykonaj w pakiecie `playlist`.
- Napisz klasę `Song` z polami `title` (typu `String`), `artist` (typu `String`) oraz `duration` (typu `int`). Zaimplementuj dwie klasy implementujące generyczny interfejs `Comparator`: `DurationComparator` do porównywania obiektów po polu `duration` (od najdłuższej do najkrótszej piosenki) oraz `ArtistTitleComparator` do porównywania obiektów po polu `artist` (zgodnie z porządkiem leksykograficznym) i w przypadku równości po polu `title`. Stwórz tablicę 5 obiektów klasy `Song` i posortuj ją (w jednej instrukcji) zgodnie z oboma kryteriami (najpierw po długości utworu,a następnie po artyście i tytule).

Zadanie 3. (12pkt max.)

- Poniższe czynności wykonaj w pakiecie `moto`.
- Napisz statyczną metodę generyczną `maxValue`, która przyjmuje tablicę elementów typu generycznego `T`, gdzie `T` rozszerza `Comparable<T>`. Metoda powinna zwracać największy element (zgodnie z porządkiem naturalnym) z tablicy. Upewnij się, że metoda nie akceptuje `null` i pustej tablicy (o zerowej liczbie elementów). Stwórz klasę `Vehicle` z polami `model` i `speed`, implementującą generyczny Comparable, i przetestuj metodę `maxValue` na tablicy obiektów `Vehicle`.

Zadanie 4. (12pkt max.)

- Poniższe czynności wykonaj w pakiecie `university`.
- Zaimplementuj metodę `countElements(Iterator<T> items, T element)`, która zlicza ile razy dany element pojawił się w kolekcji implementującej interfejs `Iterator`. Metoda powinna porównywać elementy przy użyciu metody `equals`. Stwórz klasę `Student` z polami `name` (typu `String`) i `grade` (typu `double`) i przetestuj metodę `countElements` na liście tablicowej obiektów typu `Student`.