Kolokwium #2 - Programowanie obiektowe - Zestaw A32

Imię i nazwisko, numer albumu .....

Informacje wstępne

- Łącznie do zdobycia max **60** punktów. Próg zaliczenia: 25 pkt (bez innych punktów).
- **Kolokwium należy wykonać na komputerach zamontowanych na stałe w pracowniach.**
- Student przesyłając rozwiązania oświadcza, że rozwiązał je samodzielnie.
- W trakcie kolokwium 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 kolokwium przez wszystkie grupy ćw.
- Należy zwracać uwagę na właściwe umieszczenie kodu (luzem lub w pakiecie).
- Kod musi się kompilować, aby był 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, ale nie należy tego nadużywać.
- Jeśli w poleceniu nie są sprecyzowane modyfikatory dostępu, należy dostępować zgodnie z zasadami hermetyzacji.
- **W rozwiązaniach należy uwzględniać dobre praktyki omawiane na wykładzie i ćwiczeniach, 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 kolokwium.
- **Nazwa archiwum powinna być wg schematu NUMERZESTAWU_NUMERALBUMU.zip gdzie numer zestawu znajduje się na górze kartki z poleceniami. np. A23_123456.zip.**
- Archiwum powinno być bez hasła.
- Kod zakomentowany nie będzie sprawdzany.
- Zawartość pendrive będzie pusta. Udostępniony będzie tylko w celu zgrania rozwiązań. Umieszczenie poleceń na pendrive powinno odbyć się w czasie kolokwium. Rozwiązania po czasie mogą nie być sprawdzane.
- Jeśli w poleceniu pojawia się informacja o konieczności zachowania formatowania napisów (np. wielkość znaków, znaki interpunkcyjne), to należy to bezwzględnie wykonać.
- Podpunkty będą oceniane kaskadowo — wykonanie ich bez wykonania wcześniejszych podpunktów może oznaczać zero punktów.
- 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).
- 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.
- Po kartkach z poleceniami można pisać i traktować jako brudnopis.

Zadanie 1. (15pkt max.)

A. Wykonaj poniższe czynności:

- Stwórz klasę `BasketballPlayer`, która powinna być częścią odpowiedniego pakietu `sports`.
- Klasa `BasketballPlayer` powinna posiadać dwa pola:
  - `name`: typu `String`, reprezentującego nazwę koszykarza.
  - `points`: tablica pięciu zmiennych typu `int`, reprezentująca liczbę zdobytych punktów podczas różnych meczów.
- Zaimplementuj w klasie `BasketballPlayer` interfejs `Cloneable`.
- Nadpisz metodę `clone` z interfejsu `Cloneable`, aby umożliwić klonowanie obiektów klasy `BasketballPlayer`.
- W zadaniu uwzględnij głębokie kopiowanie dla pola będącego tablicą.

B. Wykonaj poniższe czynności:

- Napisz metodę `main` w klasie `TestBasketballPlayer` w tym samym pakiecie, a w niej:
  - Utwórz obiekt klasy `BasketballPlayer`.
  - Sklonuj utworzony obiekt `BasketballPlayer`.
  - Zmień liczbę punktów na pierwszej pozycji w tablicy `points` oryginalnego koszykarza (stwórz w tym celu odpowiednią metodę).
  - Wyświetl liczbę punktów obu koszykarzy (oryginału i jego klona), aby sprawdzić, czy zmiany w jednym obiekcie nie wpływają na drugi, świadcząc o ich niezależności.

Zadanie 2. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `service`.
- Napisz klasę `ServiceProvider` z polami `id` (typu int), `name` (typu String) oraz `experienceLevel` (typu double). Zaimplementuj generyczny interfejs `Comparator` do porównywania obiektów po polu `experienceLevel` (od najwyższego do najniższego poziomu doświadczenia), a w przypadku równości po polu `name` (zgodnie z porządkiem naturalnym). Stwórz listę tablicową 5 obiektów klasy `ServiceProvider` i posortuj ją zgodnie z opisanym kryterium.

Zadanie 3. (15pkt max.)

- Wykonaj następujące czynności w pakiecie `arrayAlg`.
- Utwórz statyczną metodę generyczną `findMinIndex`, która przyjmuje tablicę elementów typu generycznego `T`. Metoda powinna zwrócić indeks elementu o najmniejszej wartości w tablicy (w przypadku kilku takich elementów, zwróć najniższy indeks takie elementu). Do porównywania elementów tablicy, typ `T` musi implementować interfejs `Comparable<T>`. W przypadku, gdy tablica jest null lub pusta, metoda powinna wyrzucać wyjątek typu `IllegalArgumentException`.
- Napisz klasę `Book`, która zawiera pola: `title` (typu `String`), `author` (typu `String`) i `yearPublished` (typu `int`).
- Stwórz przypadek testowy w klasie `TestArrayAlg`, aby zademonstrować działanie metody `findMinIndex` na niepustej tablicy obiektów typu `Book`. W teście sprawdź, czy metoda poprawnie zwraca indeks książki opublikowanej najwcześniej.

Zadanie 4. (15pkt max.)

- W pakiecie `algorithm`, utwórz statyczną metodę generyczną `isSecondGreater`. Metoda ta powinna przyjmować dwa argumenty typu generycznego `T`. Metoda powinna zwracać `true`, jeśli drugi argument jest większy od pierwszego zgodnie z metodą `compareTo` zdefiniowaną dla typu `T`. W przypadku, gdy którykolwiek z argumentów jest `null`, metoda powinna wyrzucać wyjątek `NullPointerException`. W przeciwnym razie, metoda powinna zwracać `false`.

- Stwórz klasę `TestAlgorithm` z metodą `main`. W metodzie `main` przetestuj działanie metody `isSecondGreater` z różnymi typami danych i różnymi scenariuszami, w tym z wartościami `null`.