

## Kolokwium #2 - Programowanie obiektowe - Zestaw A31

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

B. Wykonaj poniższe czynności:

- Napisz metodę `main` w klasie `TestFootballPlayer` w tym samym pakiecie, a w niej
  - Utwórz obiekt klasy `FootballPlayer`.
  - Sklonuj utworzony obiekt `FootballPlayer`.
  - Zmień liczbę goli na ostatniej pozycji w tablicy `goals` oryginalnego piłkarza (stwórz w tym celu odpowiednią metodę).
  - Wyświetl liczbę goli obu piłkarzy (oryginału i jego kłona), aby sprawdzić, czy zmiany w jednym obiekcie nie wpływają na drugi, świadcząc o ich niezależności.

## Zadanie 2. (15pkt max.)

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

## Zadanie 3. (15pkt max.)

- Wykonaj następujące czynności w pakiecie `arrayAlg`.
- Utwórz statyczną metodę generyczną `findMaxIndex`, która przyjmuje tablicę elementów typu generycznego `T`. Metoda powinna zwrócić indeks elementu o największej wartości w tablicy (w przypadku kilku takich elementów, zwróć najniższy indeks takiego elementu). Do porównywania elementów tablicy, typ `T` musi implementować interfejs `Comparable<T>`. W przypadku, gdy tablica jest `null` lub pusta, metoda powinna wyrzucić wyjątek typu `IllegalArgumentException`.
- Napisz klasę `VideoGame`, która zawiera pola: `name` (typu `String`), `developer` (typu `String`) i `rating` (typu `float` reprezentującego ocenę gry).
- Stwórz przypadek testowy w klasie `TestArrayAlg`, aby zademonstrować działanie metody `findMaxIndex` na niepustej tablicy obiektów typu `VideoGame`. W teście sprawdź, czy metoda poprawnie zwraca indeks gry z najwyższą oceną.

## Zadanie 4. (15pkt max.)

- W pakiecie `algorithm`, utwórz statyczną metodę generyczną `areFirstAndThirdEqual`. Metoda ta powinna przyjmować trzy argumenty typu generycznego `T`. Powinna ona zwracać wartość `true`, jeśli pierwszy i trzeci argument są sobie równe zgodnie z metodą `equals` zdefiniowaną dla typu `T`, przy czym metoda powinna być zabezpieczona przed wartościami `null` (tzn. zwracać `false`, jeśli którykolwiek z argumentów jest `null`). W przeciwnym razie, metoda powinna zwracać `false`.

- Stwórz klasę `TestAlgorithm` zawierającą metodę `main`. W metodzie `main` utwórz przypadek testowy dla metody.

