

## Kolokwium #2 - Programowanie obiektowe - Zestaw A15

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 `Hotel` (pol. `Hotel`), która powinna być umieszczona w pakiecie `tourism`.
- Klasa powinna posiadać prywatne pola:
  - `name`, (nazwa hotelu), typ `String`.
  - `capacity`, (pojemność hotelu, czyli liczba dostępnych pokoi), 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`.
- Nadpisz metodę `toString` tak, aby zwracała napis z reprezentacją obiektu.
- Zaimplementuj w tej klasie metodę `clone` z interfejsu `Cloneable`.

B. Wykonaj poniższe czynności:

- Stwórz klasę `TestHotel` (klasa testująca) umieść w pakiecie tym samym co klasy z punktu A, ale w innym pliku. Umieść w tej klasie tylko metodę `main`. W metodzie `main` stwórz 2 obiekty typu `Hotel` i sprawdź działania klonowania (kopiowania) - zmień nazwę oryginału, a następnie sprawdź nazwy obu obiektów.

## Zadanie 2. (15pkt max.)

A. Utwórz rekord `BusRoute` w pakiecie `transport`, który powinien zawierać trzy pola:

- `routeNumber`: typu `String`, reprezentującego numer trasy autobusowej.
- `startLocation`: typu `String`, reprezentującego punkt początkowy trasy.
- `endLocation`: typu `String`, reprezentującego punkt końcowy trasy.

B. Dodaj do rekordu `BusRoute`:

- Kompaktowy konstruktor, który weryfikuje, czy trasa (`routeNumber`) nie jest pusta. Jeśli `routeNumber` jest pusty lub `null`, konstruktor powinien rzucać wyjątek `IllegalArgumentException`.
- Metodę `isLongRoute`, która zwraca `true`, jeśli trasa autobusu jest uznawana za długą (długość napisu `routeNumber` jest większa niż 4), i `false` w przeciwnym przypadku.
- Metodę `printDetails`, która wyświetla informacje o trasie, włączając w to numer trasy, punkt początkowy i końcowy.

C. W pakiecie `transport`, utwórz klasę testową `TestBusRoute` z metodą `main`, w której:

- Utwórz dwa obiekty typu `BusRoute` z różnymi danymi.
- Wywołaj metodę `printDetails` na każdym z obiektów, aby wyświetlić ich szczegóły.
- Sprawdź, które trasy są długie, używając metody `isLongRoute`.

## Zadanie 3. (15pkt max.)

- Wykonaj czynności w pakiecie `swap`.
- Utwórz statyczną metodę generyczną `swapElements`, która przyjmuje tablicę elementów typu generycznego `T` oraz dwa indeksy w tej tablicy. Metoda powinna zamieniać miejscami elementy znajdujące się na tych indeksach.

- Na przykład, `swapElements(new String[]{"apple", "banana", "cherry"}, 0, 2)` powinno zwrócić tablicę `{"cherry", "banana", "apple"}`.
- Napisz klasę `VideoGame`, która zawiera pola: `name` (typu `String`), `developer` (typu `String`) i `rating` (typu `float` reprezentującego ocenę gry).
- Stwórz przypadek testowy, aby zademonstrować działanie metody na niepustej tablicy obiektów typu `VideoGame`.

#### Zadanie 4. (15pkt max.)

- W pakiecie `algorithm` utwórz statyczną metodę generyczną `isSorted`, która przyjmuje trzy parametry typu generycznego `T`, przyjmując, że `T` implementuje interfejs `Comparable<T>`. Metoda powinna zwracać `true`, jeśli elementy są posortowane w kolejności rosnącej, lub `false` w przeciwnym przypadku.
- Na przykład, `isSorted(1, 2, 3)` powinno zwrócić `true`, natomiast `isSorted(1, 3, 2)` powinno zwrócić `false`.
- Stwórz przypadek testowy.

