Kolokwium #2 - Programowanie obiektowe - Zestaw A17

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 `MealSet` (pol. Zestaw Posiłków), która powinna być umieszczona w pakiecie `fooddesign`.
- Klasa powinna posiadać prywatne pola:
  - `name`, (nazwa zestawu posiłków), typ `String`.
  - `numberOfItems`, (liczba elementów w zestawie), 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 (`""`).
  - Liczba elementów musi być liczbą dodatnią, w przeciwnym wypadku ustaw ją na 5.0.
- 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 `TestMealSet` (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 `MealSet` i sprawdź działania klonowania (kopiowania) - zmień nazwę oryginału, a następnie sprawdź nazwy obu obiektów.

Zadanie 2. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `school`.
- Napisz klasę `SchoolBook`, która zawiera pola: `name` (typu `String`), `author` (typu `String`) i `rating` (typu `float` reprezentującego ocenę książki). Zaimplementuj generyczny interfejs `Comparable` tak, aby obiekty klasy `SchoolBook` były porównywane według kryterium: malejąco według długości pola `name`. Stwórz listę tablicową 4 obiektów klasy `SchoolBook` i posortuj ją według tego kryterium.

Zadanie 3. (15pkt max.)

- Wykonaj czynności w pakiecie `arrayutils`.
- Utwórz statyczną metodę generyczną `shiftElementsRight`, która przyjmuje tablicę elementów typu generycznego `T`. Metoda powinna przesuwać wszystkie elementy tablicy o jedną pozycję w prawo. Ostatni element tablicy powinien zostać przeniesiony na początek. Metoda ma działać w miejscu, modyfikując argument.
- Napisz klasę `Book`, która zawiera pola: `title` (typu `String`), `author` (typu `String`) i `year` (typu `int` reprezentującego rok wydania).
- Stwórz przypadek testowy, aby zademonstrować działanie metody na niepustej tablicy obiektów typu `Book`.

Zadanie 4. (15pkt max.)

W pakiecie `algorithm` utwórz metodę `isSorted`, która sprawdza, czy tablica elementów typu generycznego `T` jest posortowana. Typ `T` powinien implementować interfejs `Comparable<T>`. Metoda zwraca `true`, jeśli elementy są uporządkowane rosnąco, i `false` w przeciwnym przypadku. Na przykład, `isSorted(new Integer[]{1, 2, 3, 4, 5})` powinno zwrócić `true`, a `isSorted(new Integer[]{5, 3, 4, 1, 2})` powinno zwrócić `false`. Utwórz przypadek testowy dla tej metody.