Kolokwium #2 - Programowanie obiektowe - Zestaw A25

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 `Pilot`, która powinna być częścią odpowiedniego pakietu `aviation`.
- Klasa `Pilot` powinna posiadać dwa pola:
  - `name`: typu `String`, reprezentującego imię pilota.
  - `flightHours`: tablica pięciu zmiennych typu `double`, reprezentująca liczbę godzin lotu podczas różnych misji.
- Zaimplementuj w klasie `Pilot` interfejs `Cloneable`.
- Nadpisz metodę `clone` z interfejsu `Cloneable`, aby umożliwić klonowanie obiektów klasy `Pilot`.
- W zadaniu uwzględnij głębokie kopiowanie dla pola będącego tablicą.

B. Wykonaj poniższe czynności:

- Napisz metodę `main` w klasie `TestPilot` w tym samym pakiecie, a w niej
  - Utwórz obiekt klasy `Pilot`.
  - Sklonuj utworzony obiekt `Pilot`.
  - Zmień liczbę godzin lotu na pozycji trzeciej w tablicy `flightHours` oryginalnego pilota (stwórz w tym celu odpowiednią metodę).
  - Wyświetl liczby godzin lotu obu pilotów (oryginału i jego klona), aby sprawdzić, czy zmiany w jednym obiekcie nie wpływają na drugi, świadcząc o ich niezależności.

Zadanie 2. (15pkt max.)

- Wykonaj czynności w pakiecie `beverages`.
- Napisz klasę `BeverageItem`, która zawiera pola: `name` (typu `String`), `volume` (typu `double`) i `sugarContent` (typu `double`). Zaimplementuj generyczny interfejs `Comparable` w taki sposób, aby obiekty klasy `BeverageItem` były sortowane rosnąco według zawartości cukru (`sugarContent`). Stwórz listę tablicową 4 obiektów klasy `BeverageItem` i posortuj ją według sprecyzowanego kryterium.

Zadanie 3. (15pkt max.)

W pakiecie `checking`, utwórz statyczną metodę generyczną `containsDuplicates`, która akceptuje trzy argumenty tego samego typu generycznego `T`. Metoda powinna zwracać `true`, jeśli co najmniej dwa z tych argumentów są sobie równe, wykorzystując do tego porównania metodę `equals`. Zaimplementuj również przypadek testowy dla tej metody, aby sprawdzić jej działanie.

Zadanie 4. (15pkt max.)

W pakiecie `algorithm` napisz statyczną metodę generyczną `updateAtIndex`, która akceptuje trzy argumenty: tablicę typu generycznego `T`, indeks typu `int` oraz element typu `T`. Zadaniem metody jest zaktualizowanie elementu tablicy na pozycji określonej przez indeks, ustawiając na nim przekazany element. Upewnij się, że metoda weryfikuje poprawność indeksu. Stwórz przypadek testowy dla tej metody.