Kolokwium #2 - Programowanie obiektowe - Zestaw A11

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 `Destination` w pakiecie `travel`, która powinna zawierać trzy pola:
  - `name`: typu `String`, reprezentującego nazwę miejsca docelowego.
  - `country`: typu `String`, reprezentującego kraj, w którym znajduje się miejsce docelowe.
  - `distance`: typu `int`, reprezentującego odległość od punktu wyjścia w kilometrach.
- Zaimplementuj dwie klasy, które implementują generyczny interfejs `Comparator<Destination>`:
  - `DistanceComparator`: porównuje obiekty klasy `Destination` według odległości (`distance`), od najbliższego (najmniejszego) do najdalszego (najwyższego) miejsca docelowego.
  - `CountryNameComparator`: porównuje obiekty klasy `Destination` najpierw według kraju (`country`), a w przypadku równości - według nazwy (`name`). W obu przypadkach porządek powinien być odwrotny do naturalnego.

B. Wykonaj poniższe czynności:

- W klasie `TestDestination` w tym samym pakiecie w metodzie `main`:
  - Utwórz i posortuj tablicę obiektów `Destination` najpierw według odległości (używając `DistanceComparator`).
  - W przypadku, gdy dwa miejsca mają tę samą odległość, zastosuj `CountryNameComparator`, aby ustalić kolejność.
  - Po zakończeniu sortowania wyświetl posortowaną listę, aby sprawdzić, czy sortowanie przebiegło prawidłowo i czy kolejność miejsc jest zgodna z założeniami.

Zadanie 2. (15pkt max.)

A. Utwórz rekord `MusicTrack` w pakiecie `music`, który powinien zawierać trzy pola:

- `title`: typu `String`, reprezentującego tytuł utworu muzycznego.
- `artist`: typu `String`, reprezentującego artystę wykonującego utwór.
- `duration`: typu `double`, reprezentującego czas trwania utworu w minutach.

B. Dodaj do rekordu `MusicTrack`:

- Kompaktowy konstruktor, który weryfikuje, czy czas trwania (`duration`) jest większy od 0. Jeśli `duration` jest mniejszy lub równy 0, konstruktor powinien rzucać wyjątek `IllegalArgumentException`.
- Metodę `isLongTrack`, która zwraca `true`, jeśli czas trwania (`duration`) utworu jest dłuższy niż 5 minut, i `false` w przeciwnym przypadku.
- Metodę `printDetails`, która wyświetla informacje o utworze, włączając w to jego tytuł, artystę i czas trwania.

C. W pakiecie `music`, utwórz klasę testową `TestMusicTrack` z metodą `main`, w której:

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

Zadanie 3. (15pkt max.)

W pakiecie `sorting` zaimplementuj statyczną metodę generyczną `sortDescending`, która przyjmuje tablicę obiektów typu generycznego `T` i sortuje je w porządku malejącym. Zakłada się, że typ `T` implementuje interfejs `Comparable<T>`. Metoda powinna sortować w miejscu, modyfikując przekazaną tablicę. Utwórz przypadek testowy dla tej metody, używając tablicy obiektów klasy `Product` z polami `name` (nazwa, typ `String`) oraz `price` (cena, typ `double`).

Zadanie 4. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `finding`.
- Zaimplementuj statyczną metodę generyczną `findValueByKey`, która otrzymuje `HashMap<K, V>` oraz klucz typu K. Metoda powinna zwracać wartość skojarzoną z podanym kluczem. Jeśli klucz nie istnieje w mapie, metoda powinna zwracać `null`. Pamiętaj, aby obsłużyć potencjalne wyjątki związane z przekazaniem `null` jako argumentu. Stwórz przypadek testowy.