Przykładowe kolokwium #2 - Zestaw Z15

Ostatnia aktualizacja pliku: 09.01.2024 21:58.

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.)

Utwórz następujące klasy i metody:

A. Klasa `ArtGallery` w pakiecie `art` z prywatnymi polami:

- `name`: typu `String`, reprezentujący nazwę galerii.
- `city`: typu `String`, reprezentujący miasto, w którym znajduje się galeria.
- `paintings`: typu `ArrayList<String>`, lista przechowująca nazwy obrazów.

B. Metody w klasie `ArtGallery`:

- Metoda `addPainting(String painting)`: dodaje obraz do listy `paintings`.
- Metoda `removePainting(String painting)`: usuwa obraz z listy `paintings`.
- Konstruktory, gettery, settery, `toString()`, `equals()` i `hashCode()`.

C. Klasa `ContemporaryGallery`, dziedzicząca po `ArtGallery` w tym samym pakiecie, z dodatkowym prywatnym polem `numberOfInstallations`: typu `int`, reprezentujący liczbę instalacji artystycznych w galerii.

D. Metody w klasie `ContemporaryGallery`:

- Konstruktory, gettery i settery dla `numberOfInstallations`.
- Nadpisane metody `toString()`, `equals()` i `hashCode()`.

E. Napisz klasę testującą Klasa `TestArtGallery` w tym samym pakiecie:

- W metodzie `main` utwórz obiekty klasy `ArtGallery` i `ContemporaryGallery`.
- Testuj działanie metod dodawania i usuwania obrazów.
- Wyświetl informacje o obu galeriach, aby sprawdzić poprawność działania metod.

Zadanie 2. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `books`.
- Napisz klasę `Book`, która zawiera pola: `title` (typu `String`): reprezentuje tytuł książki; `author` (typu `String`): reprezentuje autora książki; `yearOfPublication` (typu `int`): reprezentuje rok publikacji książki. Zaimplementuj generyczny interfejs `Comparable` w taki sposób, aby obiekty klasy `Book` były porównywane według malejąco według długości tytułu. Stwórz listę tablicową (`ArrayList`) 4 obiektów klasy `Book` i posortuj ją według sprecyzowanego kryterium.

Zadanie 3. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `testing`.
- Utwórz statyczną metodę generyczną `compareAndPrint`, która przyjmuje dwa argumenty typu generycznego `T`. Metoda powinna wypisywać na ekranie informację, czy oba argumenty są sobie równe, wykorzystując metodę `equals`. Stwórz klasę `Vehicle` z polami `model` i `speed`, nadpisz w niej `equals` i `hashCode`, a następnie przetestuj metodę `compareAndPrint` na obiektach typu `Vehicle`.

Zadanie 4. (15pkt max.)

- Poniższe czynności wykonaj w pakiecie `default`.
- Utwórz statyczną metodę generyczną `fillWithDefaultValue`, która przyjmuje kolekcję elementów typu T oraz pojedynczy element tego samego typu. Metoda powinna wypełnić całą kolekcję przekazanym elementem. Metoda nie zwraca nowej kolekcji, lecz modyfikuje przekazaną. Dodaj zabezpieczenie, aby metoda nie mogła być wywołana z kolekcją o wartości null.