Program 2a:

```
class MyMusicClass:
def __init__(self, artist, year, most_popular, listeners):
    self.artist = artist
    self.year = year
    self.most_popular = most_popular
    self.listeners = listeners


musicList = []
n = 0

try:
    for i in range(7):
        n += 1
        print(f"\nPosition number {n}.")
        musicList.append(MyMusicClass(input("Artist's name: "),
                                      int(input("Creation or debut year: ")),
                                      input("Most popular song on Spotify: "),
                                      int(input("Number of listeners: "))))
except ValueError:
    print(f"\nYou've entered an unacceptable value at position number {n}.")
```

Program 2b:

```
musicList = []
n = 0

try:
    for i in range(7):
        n += 1
        print(f"\nPosition number {n}.")
        musicList.append([input("Artist's name: "),
                          int(input("Creation or debut year: ")),
                          input("Most popular song on Spotify: "),
                          int(input("Number of listeners: "))])
except ValueError:
    print(f"\nYou've entered an unacceptable value at position number {n}.")
```

1 Wstęp

Różne podejścia zapisane jako programy 2a i 2b umożliwiają porównanie dwóch paradygmatów programowania – **obiektowego i proceduralnego**. Oba programy mają ten sam cel – zbieranie informacji na temat ulubionych artystów muzycznych i ich utworów, natomiast wykorzystują do tego różne struktury i metody.

2 Podobieństwa

1. **Cel i funkcjonalność:**
   Oba programy mają ten sam cel i identyczną funkcjonalność.

2. **Proces zbierania danych:**
   Programy używają pętli *for* do siedmiokrotnej iteracji, pobierając wymagane informacje od użytkownika. Struktura procesu wprowadzania danych (artysta, rok, najpopularniejsza piosenka, liczba odsłuchań) jest identyczna.

3. **Obsługa błędów:**
   Oba programy posiadają obsługę błędów za pomocą bloku *try-except* do przechwytywania *ValueError*, gdy użytkownik wprowadzi wartości niebędące liczbami całkowitymi tam, gdzie są one oczekiwane (rok debiutu lub powstania oraz liczba słuchaczy).

4. **Informacje zwrotne dla użytkownika:**
   Oba programy przekazują użytkownikowi informację zwrotną – komunikat wyświetlający numer pozycji na liście, do której wprowadzane są dane. Pomaga to w śledzeniu procesu wprowadzania danych.

5. **Ograniczenia obsługi:**
   Oba programy ograniczają liczbę wpisów do siedmiu, zgodnie z poleceniem oraz z *range(7)* w pętli *for*.

3 Różnice

1. **Struktury danych:**
   Program 2a używa klasy *MyMusicClass* do zdefiniowania struktury przechowywania danych, program 2b używa prostej listy list bez predefiniowanej struktury.

2. **Podejście obiektowe vs proceduralne:**
   Program 2a stosuje podejście obiektowe, gdzie każda pozycja jest instancją *MyMusicClass*. Program 2b używa podejścia proceduralnego, przechowując dane jako listę.

3. **Integralność i czytelność danych:**
   Podejście obiektowe w programie 2a zapewnia lepszą czytelność danych. Każda instancja ma oznaczone atrybuty (*artist*, *year* itd.), co sprawia, że dane są bardziej uporządkowane. Podejście listowe w programie 2b może prowadzić do zamieszania i błędów w obsłudze danych.

4. **Użycie pamięci:**
   Ogólnie rzecz biorąc, instancje obiektów (jak w programie 2a) zużywają więcej pamięci niż proste listy (jak w programie 2b). Różnica ta może być znacząca w aplikacjach o większej skali.

4 Obserwacje ilościowe

- Liczba linii kodu jest nieco większa w programie 2a ze względu na definicję klasy.
- Oba programy są zaprojektowane do zbierania informacji na temat dokładnie siedmiu pozycji, więc *musicList* ma również 7 elementów.

5 Obserwacje jakościowe

- Projekt obiektowy jest lepiej skalowalny. Jeśli program wymaga rozszerzenia lub modyfikacji, posiadanie dedykowanej klasy dla danych ułatwia to zadanie.
- Podejście proceduralne może być prostsze do zrozumienia dla początkujących, ponieważ nie wymaga rozumienia klas i obiektów.

6 Podsumowanie

Chociaż oba programy mają **ten sam cel**, ich podejścia oferują **różne zalety i wady** w zakresie organizacji danych, skalowalności i czytelności. Wybór między tymi dwoma metodami zależałby od konkretnych wymagań projektu, takich jak skala danych, potrzeba przyszłych modyfikacji, czy też znajomość programowania obiektowego przez programistów.