**Array List**

Zalety:

1. Dynamiczny rozmiar: ArrayList automatycznie rozszerza się w miarę dodawania nowych elementów, dzięki czemu nie musisz określać jego rozmiaru z góry. Jest to przydatne, gdy nie znasz dokładnie liczby elementów do przechowania.
2. Łatwe dodawanie i usuwanie: ArrayList dostarcza wygodne metody do dodawania, usuwania i modyfikowania elementów. Operacje te są wydajne i można je łatwo wykonywać.
3. Szybki dostęp do elementów: ArrayList implementuje interfejs List, co oznacza, że umożliwia on szybki dostęp do elementów na podstawie ich indeksu. Odczyt i zapis elementów w ArrayList mają złożoność czasową O(1).
4. Elastyczne operacje na tablicach: ArrayList umożliwia wygodne operacje, takie jak sortowanie, przeszukiwanie i inne manipulacje na danych tablicowych.

Wady:

1. Wydajność przy dużych ilościach danych: Operacje takie jak dodawanie lub usuwanie elementów na końcu listy są szybkie, ale w przypadku dużych list operacje wstawiania lub usuwania elementów z początku lub środka listy mogą być mniej wydajne.
2. Wymagany dodatkowy miejsce w pamięci: ArrayList alokuje więcej pamięci niż potrzebna na przechowanie elementów, aby zapewnić dynamiczny rozmiar. W przypadku dużych list może to prowadzić do marnotrawstwa zasobów.
3. Brak synchronizacji: ArrayList nie jest synchronizowany, co oznacza, że nie jest bezpieczny w przypadku równoległego dostępu z wielu wątków. Jeśli aplikacja jest wielowątkowa, konieczne może być zastosowanie dodatkowych mechanizmów synchronizacji lub użycie innej klasy z kolekcji, która jest synchronizowana.
4. Brak wsparcia dla prymitywnych typów danych: ArrayList przechowuje tylko obiekty, co oznacza, że jeśli chcesz przechowywać prymitywne typy danych, musisz użyć ich odpowiednich klas opakowujących, co może prowadzić do nadmiernego narzutu na pamięć i wydajność.