**HashSet**

**Zalety HashSet:**

1. **Unikalność elementów:** HashSet automatycznie zapewnia, że przechowywane elementy są unikalne. Jeśli próba dodania elementu, który już istnieje w zbiorze, zostanie wykonana, nie spowoduje to dodania duplikatu.
2. **Szybkie operacje dodawania i usuwania:** Dzięki wykorzystaniu tablicy haszującej HashSet oferuje szybkie operacje dodawania (add()) i usuwania (remove()) elementów. Działanie te mają złożoność czasową bliską O(1) w średnim przypadku.
3. **Implementacja interfejsu Set:** HashSet implementuje interfejs Set, co oznacza, że zapewnia on zachowanie zbioru, czyli nie przechowuje duplikatów i nie gwarantuje określonego porządku elementów.

**Wady HashSet:**

1. **Brak zachowania kolejności:** HashSet nie zachowuje porządku dodawania elementów. Oznacza to, że nie można polegać na tym, że elementy zostaną przechowane w takiej samej kolejności, w jakiej zostały dodane.
2. **Brak dostępu do elementów po indeksie:** W przeciwieństwie do innych kolekcji, takich jak ArrayList, HashSet nie umożliwia dostępu do elementów za pomocą indeksu. Nie można bezpośrednio uzyskać dostępu do elementu w HashSet na podstawie jego indeksu.
3. **Nieprzewidywalny porządek iteracji:** Ponieważ HashSet nie zachowuje kolejności dodawania elementów, porządek iteracji przez zbiór może być nieprzewidywalny. W zależności od implementacji, porządek iteracji może różnić się między różnymi uruchomieniami programu.

Mimo tych wad, HashSet jest często wykorzystywany ze względu na swoją efektywność i możliwość zapewnienia unikalności elementów w kolekcji. Jeśli nie jest wymagane zachowanie określonego porządku elementów, HashSet może być dobrym wyborem do przechowywania i operowania na zbiorze danych.