Uzupełnienie Informacji cz1

Krzysztof Gębicz

Wyjątki w Javie: teoria (try / catch / throw / throws)

Wyjątek (exception) - obiekt opisujący błąd lub nietypową sytuację w czasie wykonania programu (`Throwable` → `Exception` i `Error`).

Checked vs Unchecked

- *Checked exceptions* (np. `IOException`) - wymagane zadeklarowanie (`throws`) lub obsłużenie w `try/catch`.
- *Unchecked exceptions* (np. `NullPointerException`, `IllegalArgumentException`) - dziedziczą po `RuntimeException`, nie trzeba ich deklarować.

`throw` - służy do jawnego zgłoszenia wyjątku: `throw new IllegalArgumentException("msg");`.
`throws` - deklaracja, że metoda może zgłosić określone wyjątki: `void foo() throws`

```
try { ... } catch (Typ e) { ... } finally { ... }
```

- `try` - blok gdzie może wystąpić wyjątek,
- `catch` - obsługa z określonym typem wyjątku,
- `finally` - blok wykonywany zawsze (przydatny do sprzątania).

Multi-catch: `catch (IOException | SQLException e) { ... }`.

Plusy

- Przejrzysta obsługa błędów poza głównym logicznym przepływem.
- Możliwość propagacji błędu do miejsca, które umie go sensownie obsłużyć.
- Rich information (stack trace) ułatwia debugging.

Tablice zwykłe

Deklaracja i inicjalizacja:

- `int[] a; // deklaracja`
- `a = new int[5]; // inicjalizacja, domyślne wartości = 0`
- `int[] b = {1,2,3}; // skrócona forma`

Tablice są o stałej długości — rozmiar ustalany przy tworzeniu (`a.length`).
Typy: prymitywne (`int[]`) i obiektowe (`String[]`).

Wielowymiarowe tablice: `int[][] matrix = new int[3][4];` (tablica tablic).

Efektywność: niskie narzuty pamięciowe, szybki dostęp przez indeks.

Brak metod wygodnych (jak `add`, `remove`) — trzeba użyć `System.arraycopy` lub `Arrays` do operacji.

Plusy

Plusy

- Maksymalna wydajność i minimalne narzuty (dobrze dla prymitywów).
- Prosta struktura, deterministyczny czas dostępu O(1).
- Przydatne gdy rozmiar znany z góry i rzadko się zmienia.

Kiedy unikać / wady

- Brak elastyczności (stały rozmiar).
- Brak wygodnych metod (np. wstawiania w środku).
- Mniej bezpieczne operacje kopiowania (trzeba uważać na `System.arraycopy`).

Kiedy unikać / wady

- Brak elastyczności (stały rozmiar).
- Brak wygodnych metod (np. wstawiania w środku).
- Mniej bezpieczne operacje kopiowania (trzeba uważać na `System.arraycopy`).

java.util.Arrays i ArrayList

`java.util.Arrays` — klasa pomocnicza z metodami statycznymi:

- `sort`, `binarySearch`, `equals`, `deepEquals`, `toString`, `copyOf`, `asList`.
- Przydatna do operacji na tablicach.

`ArrayList<E>` — implementacja dynamicznej tablicy (`List`):

- Rozmiar dynamiczny, automatyczne rozszerzanie.
- Metody: `add`, `remove`, `get`, `set`, `size`, `contains`.
- Oparte na zwykłej tablicy wewnętrznej (zmiana rozmiaru przez kopiowanie).
- Amortyzowany koszt dodania: O(1) średnio; losowy dostęp: O(1); usuwanie/wstawianie w środku: O(n).
- Nie jest synchroniczna (nie-thread-safe) — użyj `Collections.synchronizedList` jeśli potrzeba.

Zalety

`java.util.Arrays` zalety

- Wiele gotowych, wydajnych metod do operacji na tablicach.
- `Arrays.sort` i `Arrays.binarySearch` — efektywne algorytmy.
- `Arrays.copyOf` ułatwia rozszerzanie tablic.

`ArrayList` zalety

- Dynamiczne rozmiary = wygoda.
- Bogate API kolekcji (`List`, `Collections`).
- Łatwe użycie z generykami i strumieniami (Streams).