PODSTAWY PROGRAMOWANIA W JAVA

dr inż. Michał Tomaszewski

katedra Metod Programowania
Polsko-Japońska Akademia Technik Komputerowych

TABLICE

Tablica jest jednorodnym zbiorem uporządkowanych elementów, które są identyfikowane przy pomocy pozycji (indeksu) wyrażonego w relacji do początkowego elementu.

Rodzaje tablic:

- statyczne (static array) - indeksacja elementów i wielkość są ustalane przed uruchomieniem programu;

Rodzaje tablic:

- statyczne (static array) - indeksacja elementów i wielkość są ustalane przed uruchomieniem programu;
- dynamiczne - na zadanym stosie (fixed stack-dynamic array) - indeksacja elementów jest ustalona, a miejsce jest alokowane podczas opracowania deklaracji;

Rodzaje tablic:

- statyczne (static array) - indeksacja elementów i wielkość są ustalane przed uruchomieniem programu;
- dynamiczne - na zadanym stosie (fixed stack-dynamic array) - indeksacja elementów jest ustalona, a miejsce jest alokowane podczas opracowania deklaracji;
- dynamiczne - na zadanej stercie (fixed heap-dynamic array) - indeksacja elementów jak i powiązanie miejsca następuje po alokacji zasobów;

Rodzaje tablic:

- statyczne (static array) - indeksacja elementów i wielkość są ustalane przed uruchomieniem programu;
- dynamiczne - na zadanym stosie (fixed stack-dynamic array) - indeksacja elementów jest ustalona, a miejsce jest alokowane podczas opracowania deklaracji;
- dynamiczne - na zadanej stercie (fixed heap-dynamic array) - indeksacja elementów jak i powiązanie miejsca następuje po alokacji zasobów;
- dynamiczne - na stercie (heap-dynamic array) - indeksacja elementów jak i zarezerwowane miejsce mogą się zmienić w czasie życia tablicy

Tablicą jest **zmienna** składająca się z zestawu **elementów**. Każdy element jest zmienną typu **prostego** lub **odnośnikowego**.

Tablicą jest **zmienna** składająca się z zestawu **elementów**. Każdy element jest zmienną typu **prostego** lub **odnośnikowego**.

Wszystkie elementy tablicy są takiego samego typu.

Tablicą jest **zmienna** składająca się z zestawu **elementów**. Każdy element jest zmienną typu **prostego** lub **odnośnikowego**.

Wszystkie elementy tablicy są takiego samego typu.

Liczba elementów nie może być większa od największej liczby typu **int**.

Tablicą jest **zmienna** składająca się z zestawu **elementów**. Każdy element jest zmienną typu **prostego** lub **odnośnikowego**.

Wszystkie elementy tablicy są takiego samego typu.

Liczba elementów nie może być większa od największej liczby typu **int**.

Z każdym elementem tablicy związany jest **indeks**. Indeks jest **nieujemną** liczbą, poczynając od numeru **0**.

FABRYKACJA TABLIC

Przykładowa instrukcja

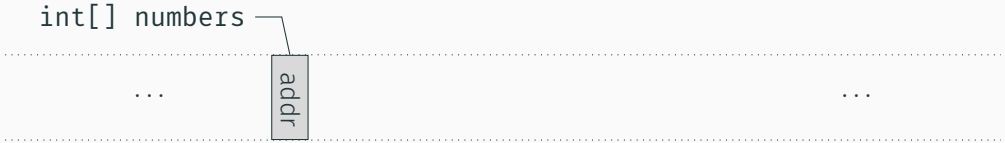
```
int[] numbers;
```

nie tworzy tablicy o elementach typu `int`.

Przykładowa instrukcja

```
int[] numbers;
```

nie tworzy tablicy o elementach typu `int`.

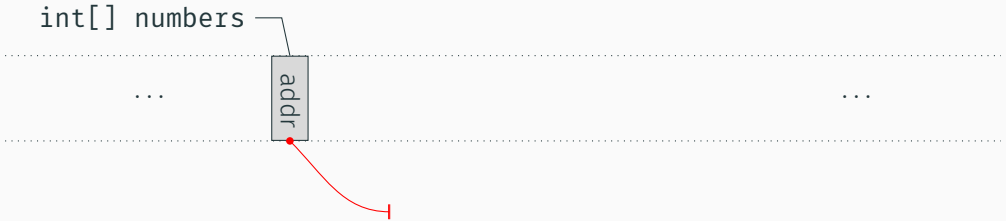


Przykładowa instrukcja

```
int[] numbers;
```

nie tworzy tablicy o elementach typu `int`.

Tworzy ona odnośnik do dowolnej 1-wymiarowej tablicy o bliżej nieokreślonej liczbie elementów typu `int`.



Przykładowa instrukcja

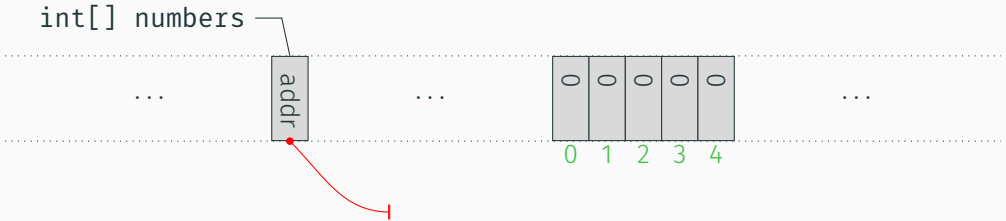
```
int[] numbers;
```

nie tworzy tablicy o elementach typu `int`.

Dopiero po użyciu fabrykatora

```
new int[5]
```

który utworzy 5-elementową tablicę o elementach typu `int`



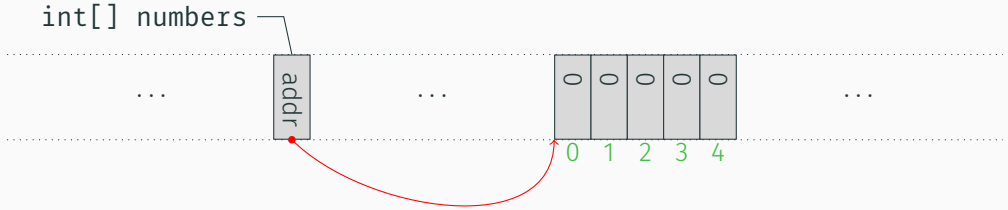
Przykładowa instrukcja

```
int[] numbers;
```

nie tworzy tablicy o elementach typu `int`.

i dostarczy do niej odniesienie

```
numbers = new int[5]
```



Przykładowa instrukcja

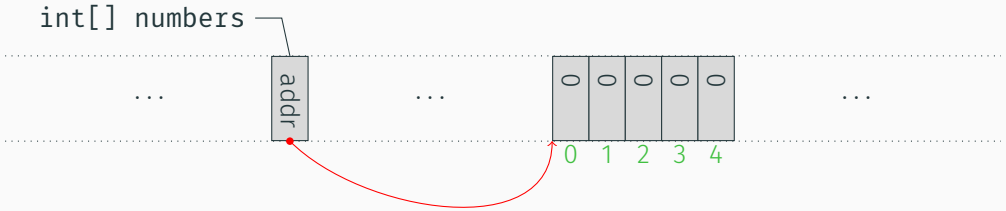
```
int[] numbers;
```

nie tworzy tablicy o elementach typu `int`.

i dostarczy do niej odniesienie

```
numbers = new int[5]
```

można, za pomocą odnośnika `numbers`, wykonywać operacje na elementach tablicy.



PRZYKŁAD 2

Instrukcja tablicy ma postać

```
{ elements }
```

w której `elements` jest listą wyrażeń, iteratorem tablicy, słowem kluczowym `null` lub słowem pustym.

Instrukcja tablicy ma postać

```
{ elements }
```

w której `elements` jest listą wyrażeń, iteratorem tablicy, słowem kluczowym `null` lub słowem pustym.

```
int[] vec = { 10, 20, 30};
```

Instrukcja tablicy ma postać

```
{ elements }
```

w której `elements` jest listą wyrażeń, iteratorem tablicy, słowem kluczowym `null` lub słowem pustym.

```
int[] vec = { 10, 20, 30};
```

Jeżeli dysponujemy nie pustym odnośnikiem do tablicy, wówczas wykorzystując odwołanie do `length` uzyskujemy informację o wielkości tablicy.

Instrukcja tablicy ma postać

```
{ elements }
```

w której `elements` jest listą wyrażeń, iteratorem tablicy, słowem kluczowym `null` lub słowem pustym.

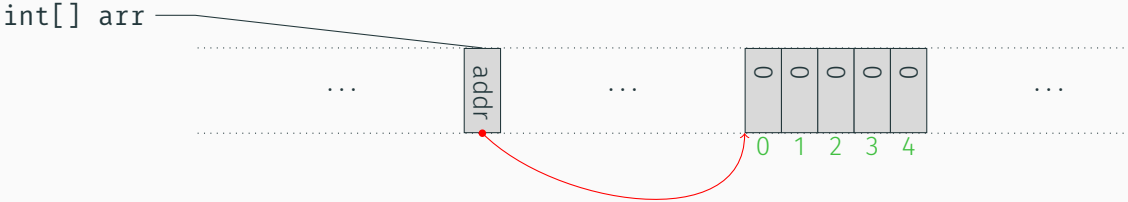
```
int[] vec = { 10, 20, 30};
```

Jeżeli dysponujemy nie pustym odnośnikiem do tablicy, wówczas wykorzystując odwołanie do `length` uzyskujemy informację o wielkości tablicy.

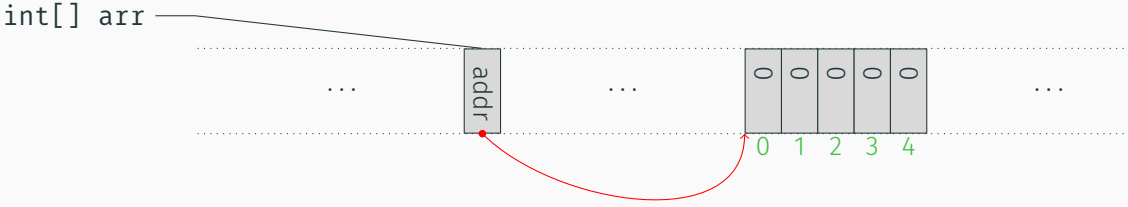
```
vec.length
```

PROBLEMY TABLICOWE

- odwołanie do elementu

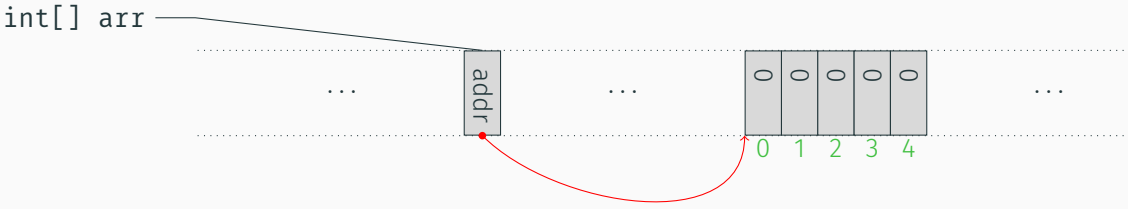


· odwołanie do elementu



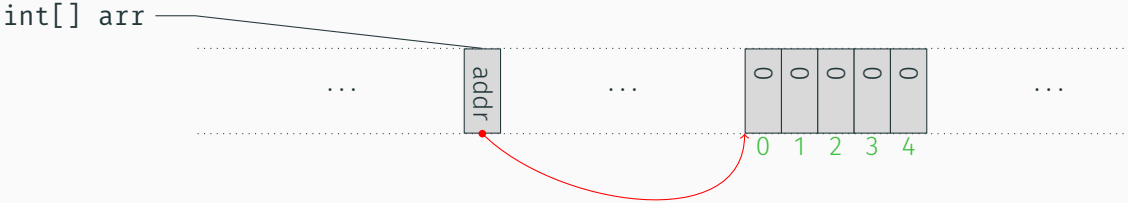
arr[0]

- odwołanie do elementu



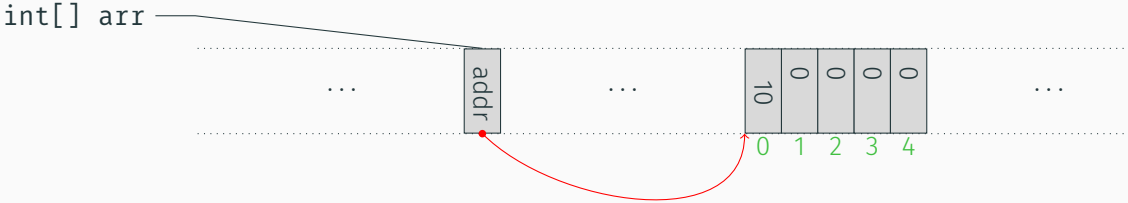

```
System.out.println(arr[0]);
```

· wypełnienie tablicy




```
arr[0] = 10;
```

- wypełnienie tablicy




```
arr[0] = 10;
```

PROBLEMY TABLICOWE

- wyszukanie zadanego elementu

PROBLEMY TABLICOWE

- wyszukanie zadanego elementu
- usunięcie wskazanego elementu

PROBLEMY TABLICOWE

- wyszukanie zadanego elementu
- usunięcie wskazanego elementu
- wstawienie elementu

Dziękuje