PODSTAWY PROGRAMOWANIA W JAVA

dr inż. Michał Tomaszewski

katedra Metod Programowania
Polsko-Japońska Akademia Technik Komputerowych

METODY

METODY

Metodą jest wyodrębniony i nazwany **opis czynności** definiowany w ciele klasy.

METODY

Metodą jest wyodrębniony i nazwany **opis czynności** definiowany w ciele klasy.

Deklaracja metody składa się z **nagłówka** opisującego jej zewnętrzne właściwości oraz **ciała** wyszczególniającego czynności.

METODY - NAGŁÓWEK

Nagłówek metody opisuje:

- kto może z niej korzystać

public

METODY - NAGŁÓWEK

Nagłówek metody opisuje:

- kto może z niej korzystać
- czy jest metodą statyczną czy nie-statyczną

```
public static
```

METODY - NAGŁÓWEK

Nagłówek metody opisuje:

- kto może z niej korzystać
- czy jest metodą statyczną czy nie-statyczną
- czy i jakiego typu rezultat dostarcza (np. int, całkowitego)

```
public static void
```

METODY - NAGŁÓWEK

Nagłówek metody opisuje:

- kto może z niej korzystać
- czy jest metodą statyczną czy nie-statyczną
- czy i jakiego typu rezultat dostarcza (np. int, całkowitego)
- jaką ma nazwę i jakie jest je przeznaczenie (np. divide, wykonuje dzielenie)

```
public static void main
```

METODY - NAGŁÓWEK

Nagłówek metody opisuje:

- kto może z niej korzystać
- czy jest metodą statyczną czy nie-statyczną
- czy i jakiego typu rezultat dostarcza (np. int, całkowitego)
- jaką ma nazwę i jakie jest je przeznaczenie (np. divide, wykonuje dzielenie)
- jakiego typu są jej parametry (np. int, int, oba typu całkowitego)

```
public static void main (String[] args)
```

METODY - NAGŁÓWEK

Nagłówek metody opisuje:

- kto może z niej korzystać
- czy jest metodą statyczną czy nie-statyczną
- czy i jakiego typu rezultat dostarcza (np. int, całkowitego)
- jaką ma nazwę i jakie jest je przeznaczenie (np. divide, wykonuje dzielenie)
- jakiego typu są jej parametry (np. int, int, oba typu całkowitego)
- czy wysyła wyjątki (np. throws ArithmeticException)

```
public static void main (String[] args)
```

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

Ciało metody:

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

Ciało metody:

- może zawierać instrukcje

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

Ciało metody:

- może zawierać instrukcje
  - w deklaracjach zmiennych nie wolno używać sepecyfikatorów dostępu;

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

Ciało metody:

- może zawierać instrukcje
  - w deklaracjach zmiennych nie wolno używać sepecyfikatorów dostępu;
  - odwołania do zmiennych definiowanych lokalnie będą poprzedzone przypisaniem wartości;

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

Ciało metody:

- może zawierać instrukcje
  - w deklaracjach zmiennych nie wolno używać sepecyfikatorów dostępu;
  - odwołania do zmiennych definiowanych lokalnie będą poprzedzone przypisaniem wartości;
- nie może zawierać definicji metody

CIAŁO METODY

Ciałem metody jest **blok** ujęty w nawiasy **klamrowe**.

```
public static void fun()
{
    int fix1 = 10, fix2;
    System.out.println(fix1); // poprawnie
    System.out.println(fix2); // nie poprawnie
}
```

Przykład

METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

Nazwa lokalna zadeklarowania w pewnym bloku jest widoczna od miejsca jej zadeklarowania, do końca **tego** bloku.

METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

Nazwa lokalna zadeklarowania w pewnym bloku jest widoczna od miejsca jej zadeklarowania, do końca **tego** bloku.

Jeśli w bloku zadeklarowano nazwę `name`, to zabrania się, aby w jakimkolwiek **zawartym** w nim bloku zadeklarowano taką samą nazwę.

METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

Nazwa lokalna zadeklarowania w pewnym bloku jest widoczna od miejsca jej zadeklarowania, do końca **tego** bloku.

Jeśli w bloku zadeklarowano nazwę `name`, to zabrania się, aby w jakimkolwiek **zawartym** w nim bloku zadeklarowano taką samą nazwę.

Nie zabrania się natomiast deklarowania takich samych nazw w rozłącznych blokach.

METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

```
public void fun(int a){
     int a = 10;
    {
        int b = 20;
        {
            int c = 30;
             int b = 40;
        }

        {
            int c = 50;
        }
    }
}
```

METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

```
public void fun(int a){
     int a = 10;
    {
        int b = 20;
        {
            int c = 30;
             int b = 40;
        }

        {
            int c = 50;
        }
    }
}
```

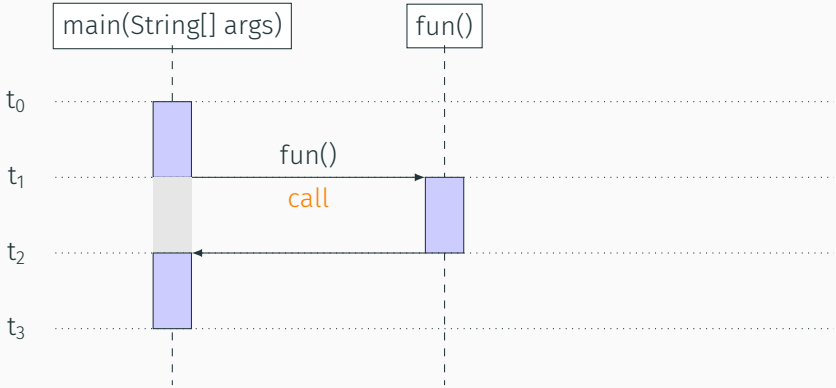
METODY - LOKALNOŚĆ NAZW

Nazwy zadeklarowane w **ciele** metody są nazwami jej zmiennych **lokalnych**.

```
public void fun(int a){
     int a = 10;
    {
        int b = 20;
        {
            int c = 30;
             int b = 40;
        }

        {
            int c = 50;
        }
    }
}
```

METODY - WYKONANIE CIAŁA METODY



METODY - WYWOŁANIE METODY

Wywołanie metody ma na celu wykonanie czynności określonych przez jej **ciało** oraz **argumenty**.

METODY - WYWOŁANIE METODY

Wywołanie metody ma na celu wykonanie czynności określonych przez jej **ciało** oraz **argumenty**.

Wartości argumentów z wywołania metody są przypisywane do parametrów metody.

METODY - WYWOŁANIE METODY

Wywołanie metody ma na celu wykonanie czynności określonych przez jej **ciało** oraz **argumenty**.

Wartości argumentów z wywołania metody są przypisywane do parametrów metody.

Jeżeli argumentami metody są:

Wywołanie metody ma na celu wykonanie czynności określonych przez jej **ciało** oraz **argumenty**.

Wartości argumentów z wywołania metody są przypisywane do parametrów metody.

Jeżeli argumentami metody są:

- **wyrażenia** – to tuż przed wywołaniem metody każde wyrażenie jest opracowane **od–lewej–do–prawej**, a jego wartości przypisuje się parametrowi metody;

Wywołanie metody ma na celu wykonanie czynności określonych przez jej **ciało** oraz **argumenty**.

Wartości argumentów z wywołania metody są przypisywane do parametrów metody.

Jeżeli argumentami metody są:

- **wyrażenia** – to tuż przed wywołaniem metody każde wyrażenie jest opracowane **od–lewej–do–prawej**, a jego wartości przypisuje się parametrowi metody;
- zmienne typów prostych - wówczas wartości przypisuje się parametrowi metody;

Wywołanie metody ma na celu wykonanie czynności określonych przez jej **ciało** oraz **argumenty**.

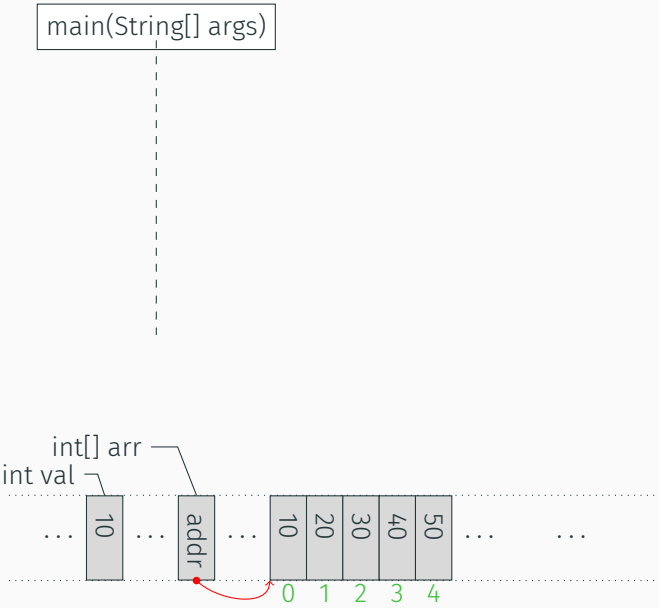
Wartości argumentów z wywołania metody są przypisywane do parametrów metody.

Jeżeli argumentami metody są:

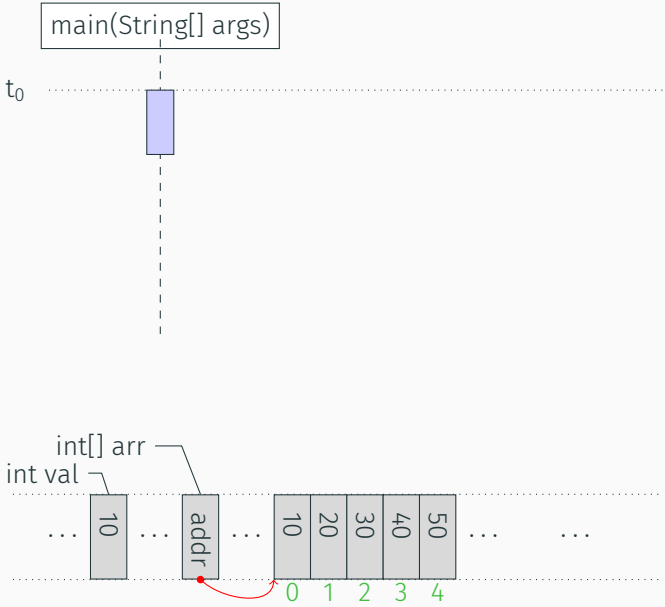
- **wyrażenia** – to tuż przed wywołaniem metody każde wyrażenie jest opracowane **od–lewej–do–prawej**, a jego wartości przypisuje się parametrowi metody;
- zmienne typów prostych - wówczas wartości przypisuje się parametrowi metody;
- zmienne odnośnikowe/odnośniki – to zmienna obiektowa identyfikowana przez przypisane mu odniesienie jest **argumentem obiektowym**

Przykład

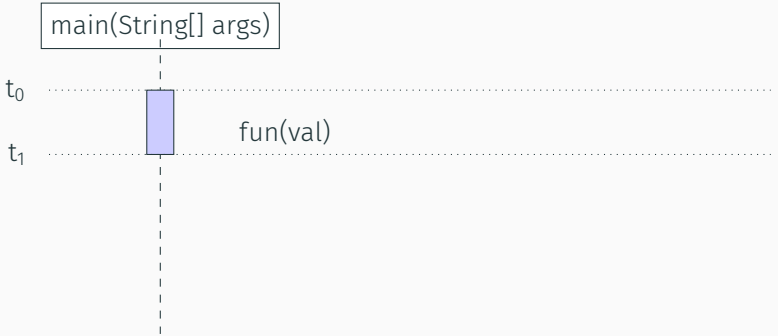
METODY - WYWOŁANIE METODY

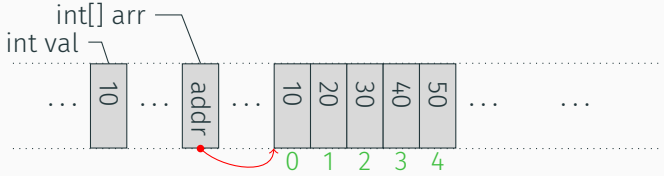


METODY - WYWOŁANIE METODY

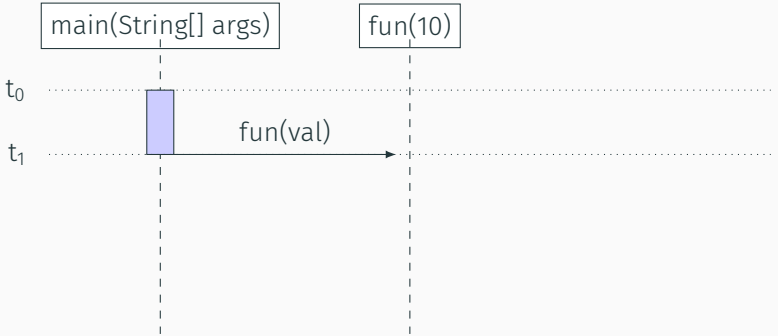


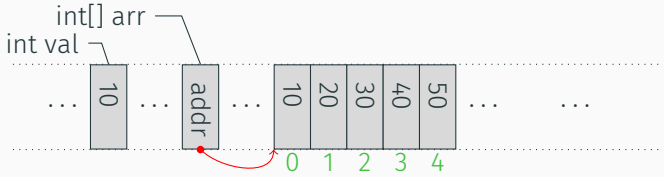
METODY - WYWOŁANIE METODY



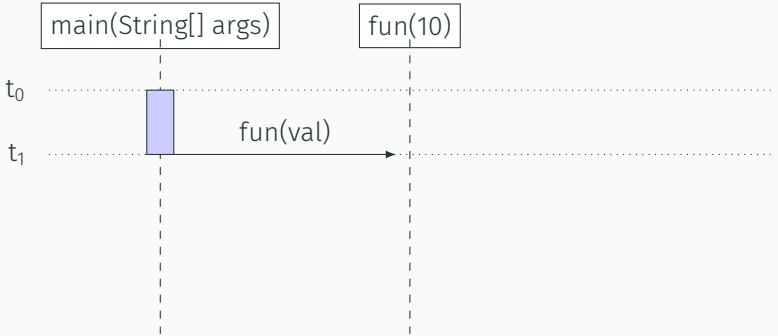


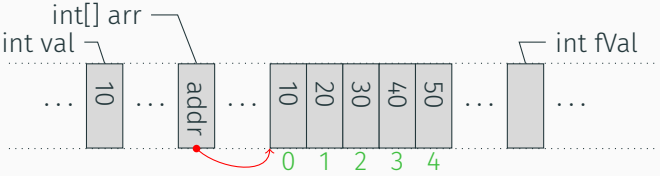
METODY - WYWOŁANIE METODY



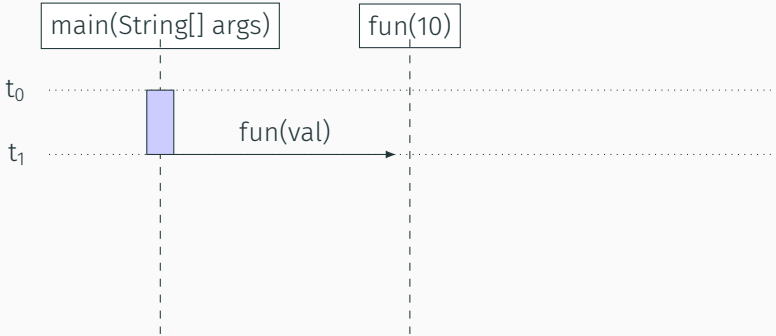


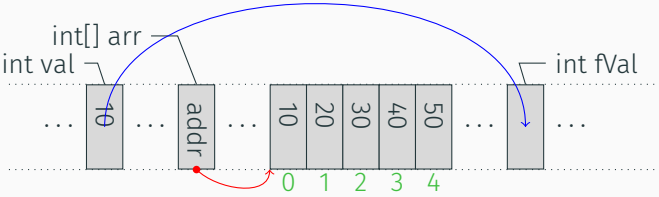
METODY - WYWOŁANIE METODY



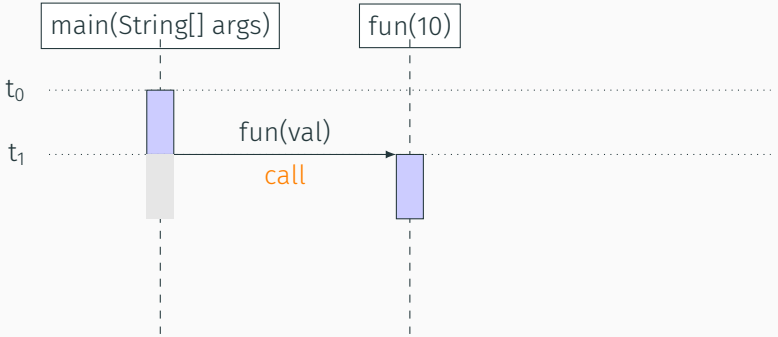


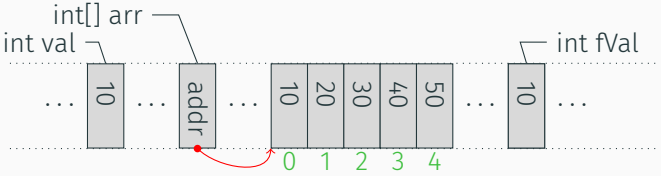
METODY - WYWOŁANIE METODY



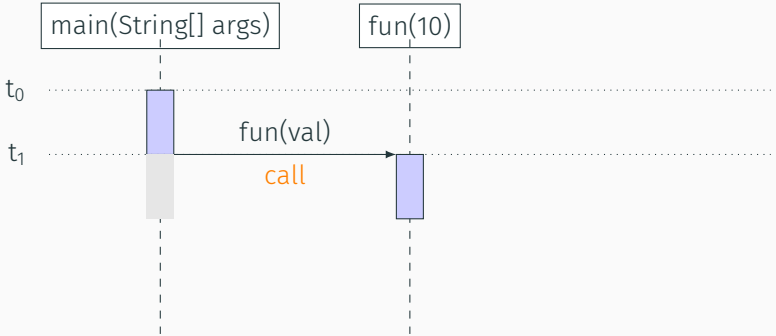


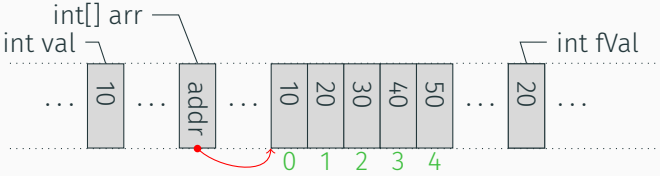
METODY - WYWOŁANIE METODY



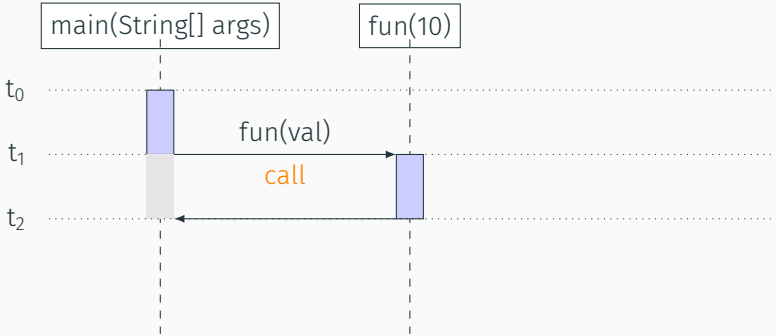


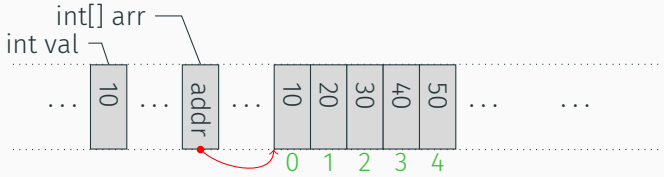
METODY - WYWOŁANIE METODY



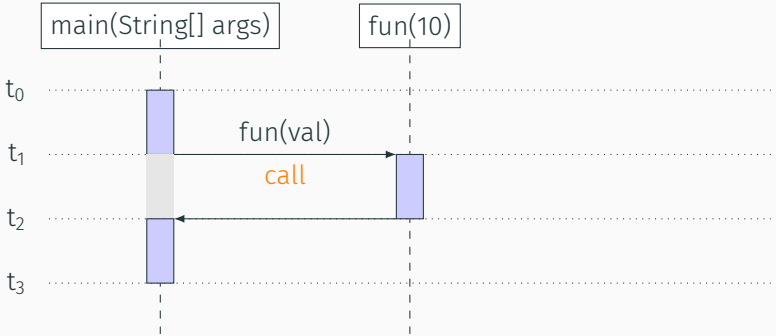


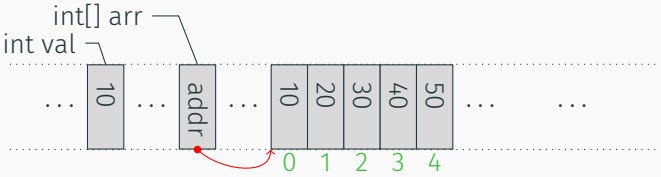
METODY - WYWOŁANIE METODY





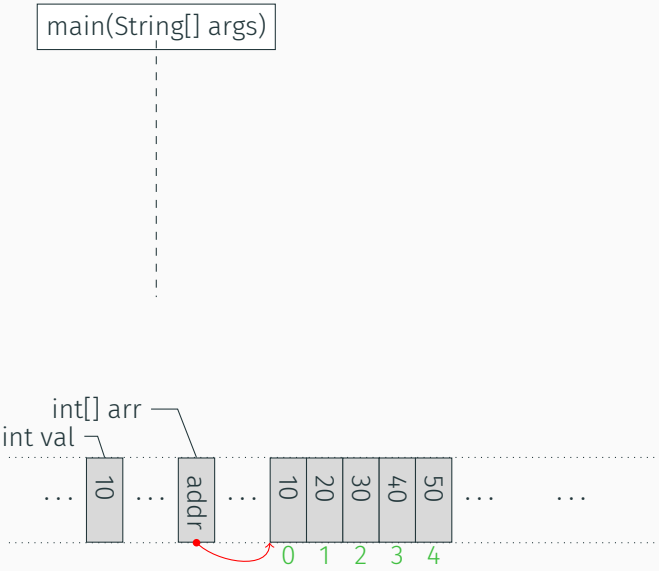
METODY - WYWOŁANIE METODY



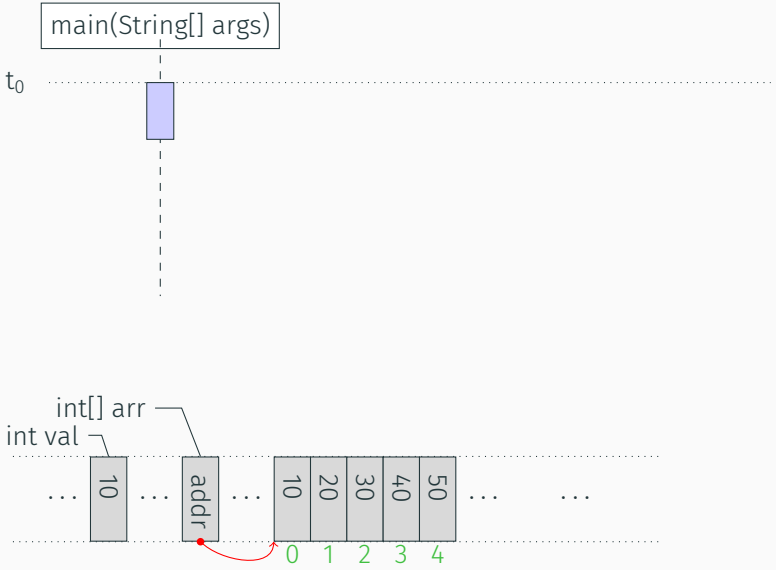


Przykład

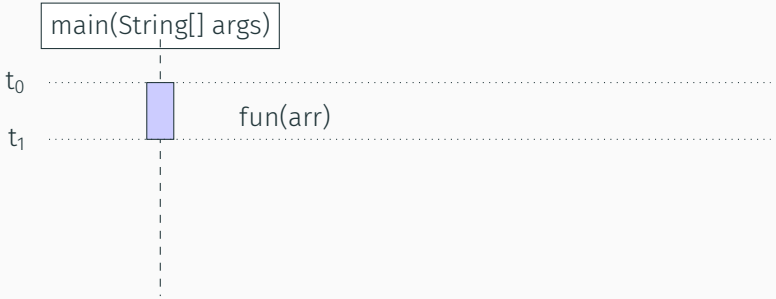
METODY - WYWOŁANIE METOD

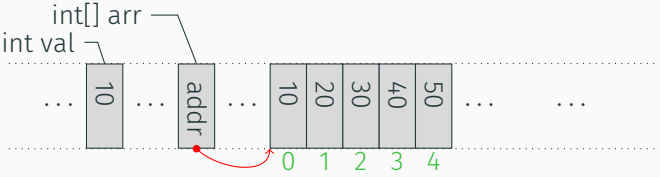


METODY - WYWOŁANIE METOD

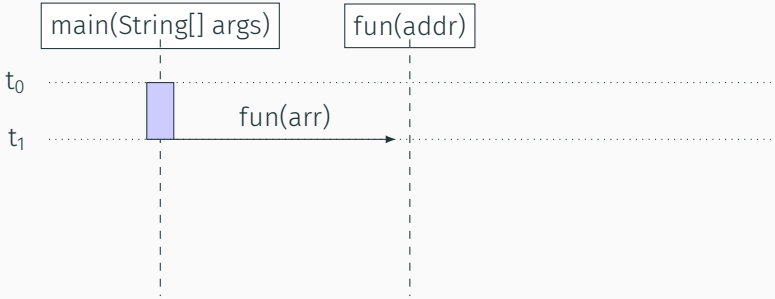


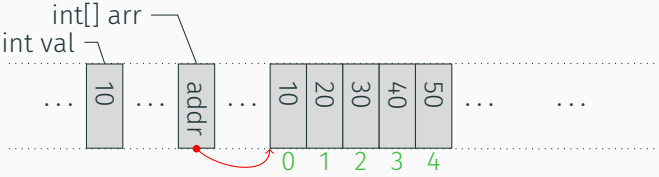
METODY - WYWOŁANIE METOD



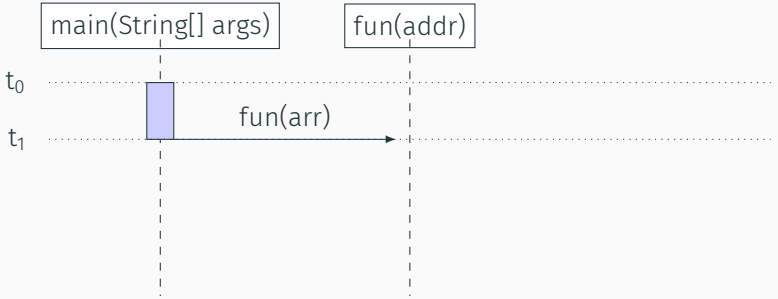


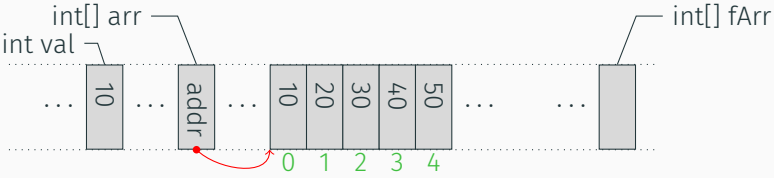
METODY - WYWOŁANIE METOD



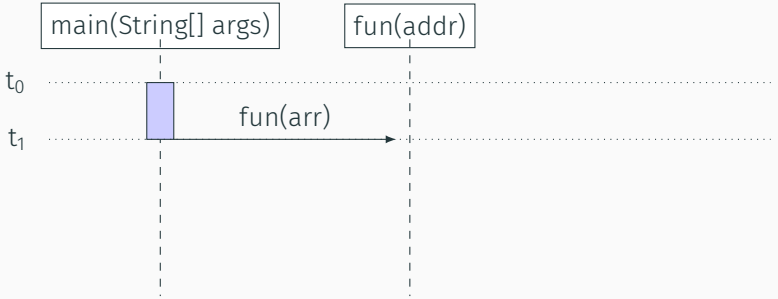


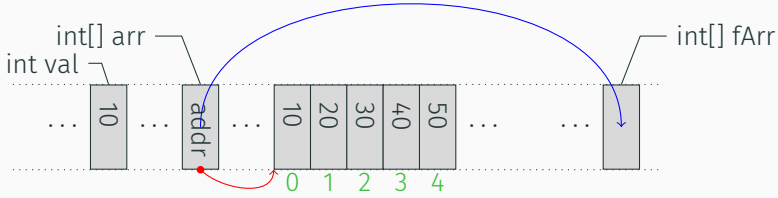
METODY - WYWOŁANIE METOD



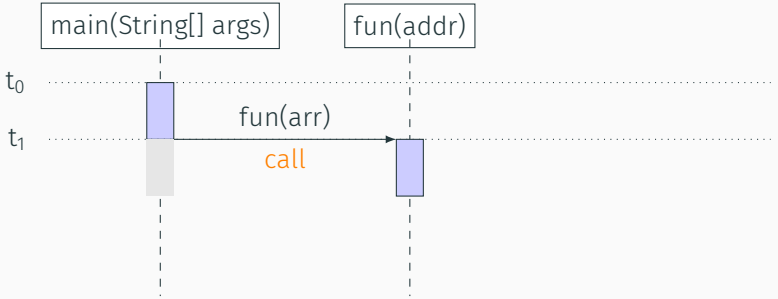


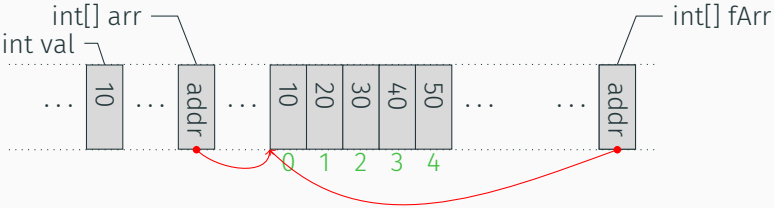
METODY - WYWOŁANIE METOD



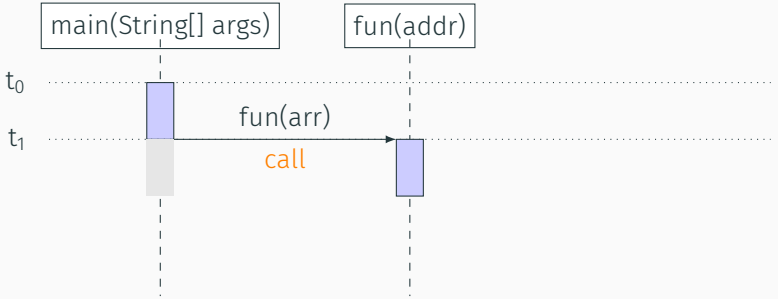


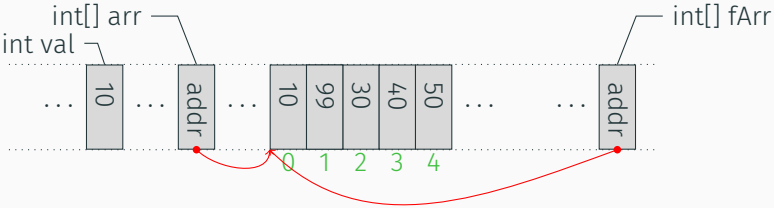
METODY - WYWOŁANIE METOD



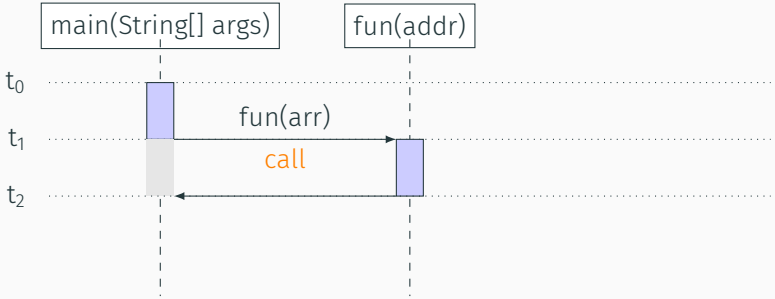


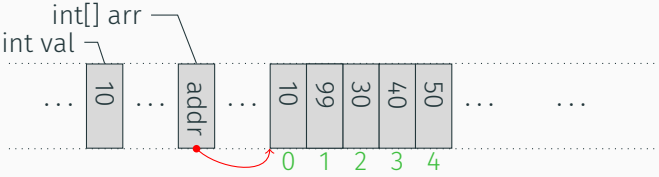
METODY - WYWOŁANIE METOD



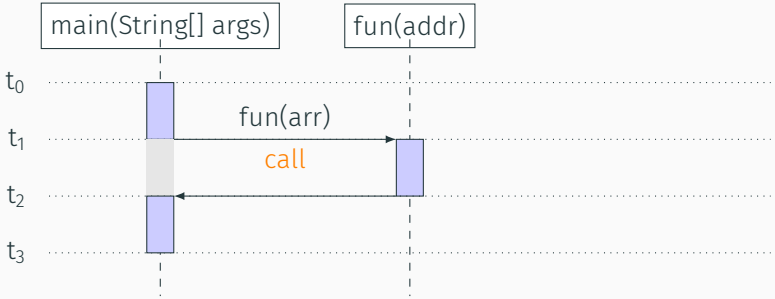


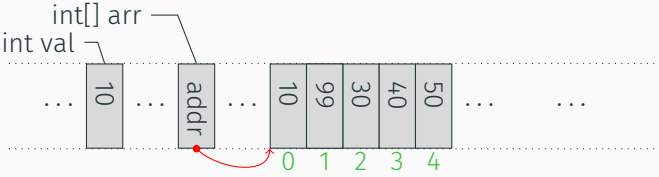
METODY - WYWOŁANIE METOD





METODY - WYWOŁANIE METOD





WYWOŁANIE METODY - SKOJARZENIA

Do każdego wywołania metody dobiera się definicję, która **najlepiej** pasuje do tego wywołania, a następnie każdy parametr metody **kojarzy się** z odpowiadającym mu argumentem.

WYWOŁANIE METODY - SKOJARZENIA

Do każdego wywołania metody dobiera się definicję, która **najlepiej** pasuje do tego wywołania, a następnie każdy parametr metody **kojarzy się** z odpowiadającym mu argumentem.

```
public void fun(int par)
{
}

public void fun(double par)
{
}
```

WYWOŁANIE METODY - SKOJARZENIA

Do każdego wywołania metody dobiera się definicję, która **najlepiej** pasuje do tego wywołania, a następnie każdy parametr metody **kojarzy się** z odpowiadającym mu argumentem.

```
public void fun(int par)
{
}

public void fun(double par)
{
}

        fun(12);
```

WYWOŁANIE METODY - SKOJARZENIA

Do każdego wywołania metody dobiera się definicję, która **najlepiej** pasuje do tego wywołania, a następnie każdy parametr metody **kojarzy się** z odpowiadającym mu argumentem.

```
public void fun(int par)
{
}

        fun(12);
```

WYWOŁANIE METODY - SKOJARZENIA

Do każdego wywołania metody dobiera się definicję, która **najlepiej** pasuje do tego wywołania, a następnie każdy parametr metody **kojarzy się** z odpowiadającym mu argumentem.

```
public void fun(int par)
{
}

public void fun(double par)
{
}

        fun(12.0);
```

Do każdego wywołania metody dobiera się definicję, która **najlepiej** pasuje do tego wywołania, a następnie każdy parametr metody **kojarzy się** z odpowiadającym mu argumentem.

```
public void fun(double par)
{
}

        fun(12.0);
```

METODY REZULTATOWE

Nagłówek metody podaje typ jej **rezultatu**. Każde wywołanie metody można traktować jako jego **nazwę**.

METODY REZULTATOWE

Nagłówek metody podaje typ jej **rezultatu**. Każde wywołanie metody można traktować jako jego **nazwę**.

Jeśli w miejscu określenia typu znajduje się słowo kluczowe **void**, to metoda jest **bez-rezultatowa**.

METODY REZULTATOWE

Nagłówek metody podaje typ jej **rezultatu**. Każde wywołanie metody można traktować jako jego **nazwę**.

Jeśli w miejscu określenia typu znajduje się słowo kluczowe **void**, to metoda jest **bez-rezultatowa**.

Powrót z wykonania metody bez-rezultatowej następuje w chwili wykonania instrukcji powrotu nie zawierającej wyrażenia. Instrukcję powrotu bez wyrażenia domniemywa się tuż przed klamrą kończącą ciało metody bez-rezultatowej. W wielu wypadkach umożliwia to **zrezygnowanie** z jawnego użycia takiej instrukcji.

METODY REZULTATOWE

Nagłówek metody podaje typ jej **rezultatu**. Każde wywołanie metody można traktować jako jego **nazwę**.

Jeśli w miejscu określenia typu znajduje się słowo kluczowe **void**, to metoda jest **bez-rezultatowa**.

Jeśli typ rezultatu metody jest różny od **void** to metoda jest **rezultatowa** i musi zawierać instrukcję powrotu zawierającą **wyrażenie**.

METODY - PRZECIĄŻANIE

Każda metoda wchodzi w skład pewnej klasy

METODY - PRZECIĄŻANIE

Każda metoda wchodzi w skład pewnej klasy, która może zawierać więcej niż jedną metodę o tej samej nazwie

Każda metoda wchodzi w skład pewnej klasy, która może zawierać więcej niż jedną metodę o tej samej nazwie – pod warunkiem że każda para takich metod **przeciążonych** będzie różniła się **sygnaturą**.

```
public int sum( int par1, int par2){
    return par1 + par2;
}

public int sum( int par1, int par2, int par3){
    return par1 + par2 + par3;
}
```

METODY - REKURENCJA

Metoda jest **rekurencyjna**, jeśli istnieje takie jej wywołanie, że jeszcze przed wykonaniem w niej instrukcji powrotu, ta sama metoda zostanie wywołana ponownie.

Zastosowanie rekurencji zazwyczaj upraszcza zapis algorytmu, ale może spowodować wydłużenie czasu obliczeń i zwiększenie zużycia pamięci operacyjnej.

PROBLEMY REKURENCYJNE

- wyświetl sekwencję liczb

PROBLEMY REKURENCYJNE

- wyświetl sekwencję liczb
- wyświetl sumę sekwencji liczb

PROBLEMY REKURENCYJNE

- wyświetl sekwencję liczb
- wyświetl sumę sekwencji liczb
- wylicz silnię liczby

PROBLEMY REKURENCYJNE

- wyświetl sekwencję liczb
- wyświetl sumę sekwencji liczb
- wylicz silnię liczby
- wylicz kolejne elementy ciągu Fibonacciego

PROBLEMY REKURENCYJNE

- wyświetl sekwencję liczb
- wyświetl sumę sekwencji liczb
- wylicz silnię liczby
- wylicz kolejne elementy ciągu Fibonacciego
- sprawdź czy tablica zawiera palindrom