Programowanie obiektowe

Wykład 2: Typy proste, tablice, łańcuchy

dr inż. Marcin Luckner
mluckner@mini.pw.edu.pl

Wydział Matematyki i Nauk Informacyjnych

Wersja 1.4
2 marca 2022

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Politechnika Warszawska
Unia Europejska Europejski Fundusz Społeczny

Projekt „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca” współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Zadanie 10 pn. „Modyfikacja programów studiów na kierunkach prowadzonych przez Wydział Matematyki i Nauk Informacyjnych”, realizowane w ramach projektu „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca”, współfinansowanego jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Politechnika Warszawska
Unia Europejska Europejski Fundusz Społeczny

Silne typowanie

- Java jest językiem *silnie typowanym*.
- Każda zmienna musi posiadać zadeklarowany typ.
- Typ jest określeniem klasy obiektu lub typem prostym.

Nazewnictwo zmiennych

Deklaracje zmiennych

```
String plikWejsciowy;
int liczbaRekordowDoWczytania;
```

- Nazwy zmiennych powinny oddawać ich znaczenie.
- Nie powinny zawierać informacji o typie danych.
- Więcej o nazewnictwie zmiennych w [Martin, 2010]
- Szczegółowe ograniczenia co do nazw dopuszczalnych w Javie [Schildt, 2020].

Typy proste

- Typy proste reprezentują liczby, znaki i wartości logiczne.
- Są jedynymi elementami Javy nie będącymi obiektami.
- Zostały wprowadzone, aby ułatwić przejście do Javy programistom C++.
- W przeciwieństwie co Pythona ich zakres jest stały i niezależny od procesora.

Liczby całkowite

Tabela 1: Zakres liczb całkowitych

typ	liczba bajtów	przybliżony zakres
`byte`	1	$[-128, 127]$
`short`	2	± 32 tysiące
`int`	4	± 2 miliardy
`long`	8	$\pm 9 \times 10^{18}$

- Stały zakres wartości zapewnia przenośność kodu pomiędzy platformami.
- Najczęściej używanym zakresem jest `int`.
- Pozostałe typy są używane przy ekstremalnych obliczeniach lub do redukcji zajętości pamięci.

Liczby rzeczywiste

Tabela 2: Zakres liczb zmiennoprzecinkowych

typ	liczba bajtów	liczba znaczących cyfr dziesiętnych
`float`	4	6-7
`double`	8	15

- W zapisie liczb rzeczywistych przecinek zmienia swoje położenie.
- Wskazane jest stosowanie typu `double`, aby zwiększyć precyzję obliczeń.
- Istnieją specjalne wartości do zapisu nieskończoności $\pm\infty$ i wartości nie będącej liczbą `NaN`.

Znak `char`

- Typ `char` miał w założeniu reprezentować pojedynczy znak.
- Zapis w kodowaniu UTF-16 pozwala na wyrażenie ponad 65 tysięcy unikalnych znaków.
- Jednak wprowadzenie do kodowania języków wschodnich doprowadziło do przekroczenia zakładanego zakresu i interpretacja `char` przestała być jednoznaczna.
- W efekcie bezpieczniej jest używać łańcuchów znaków `String`.

Typ logiczny `boolean`

- Typ logiczny `boolean` reprezentuje wartości:
  - prawda (`true`)
  - fałsz (`false`)
- Wartości logicznych nie można konwertować na wartości całkowite (0, 1).
- Dzięki temu unikamy pomyłek między przypisaniem (`x = 0`), a porównaniem (`x == 0`)

liczby jako obiekty

Tabela 3: Zakres liczb zmiennoprzecinkowych

typ prosty	typ złożony
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`

- Typ proste mają swoje odpowiedniki obiektowe.
- Obiekty reprezentują typy proste i oferują dodatkowe API.

Pakowanie i rozpakowywanie

Relacje między typami prostymi a złożonymi

```
 1  int i, j, k;
 2
 3  i = 10;
 4  j = 20;
 5  k = i + j;
 6
 7  Integer objectI, objectJ, objectK;
 8
 9  objectI = i; //new Integer(i);
10  objectJ = j; //new Integer(j);
11  objectK = objectI + objectJ;
12  objectK = i + j;
13  objectK = objectI + j;
14
15  short s =objectK.shortValue();
16  s = (short) k;
```

Łańcuchy znaków `String`

- Klasa `String` reprezentuje teksty w kodowaniu Unicode (UTF-16).
- Znaki są indeksowane od 0.
- Oferuje szerokie API do pracy z tekstami (zobacz np. [Horstmann, 2019])
- Automatyczna konwersja innych typów.
- Konkatenacja (operator $+$).
- Obiekty klasy `String` są niezmienne.

Niezmienność tekstów

- Utworzony `String` nie może być modyfikowany. Nie da się zmienić poszczególnych znaków w tekście.
- Dlatego konkatenacja tekstów tworzy nowy obiekt.

Łączenie tekstów

```
1  String s1 = "The first line";
2  String s2 = s1.substring(4,9);
3  s1 = s1 + " is " + s2;
```

- Niezmienność umożliwia współdzielenie tekstów.
- Wszystkie teksty znajdują się w jednej puli.
- Odwołując się do tego samego tekstu w różnych miejscach kodu, nie musimy tworzyć jego kopii w pamięci.

Tablice `Array`

- Tablica `Array` jest obiektem zawierającym zbiór elementów tego samego typu.
- Posiada publiczne pole `length` informujące o liczbie elementów w tablicy.
- Chcąc stworzyć tablicę, musimy skorzystać z operatora `new` i zadeklarować ilość potrzebnej pamięci.

Alokacja tablicy

```
1  int[] tablica = new int[100];
```

Deklaracja a alokacja

- Sama deklaracja typu i nazwy tablicy nie powoduje przydzielenia pamięci.

Deklaracja tablicy

```
1  int[] tablica1;
2  int tablica2[];
```

- Dopiero użycie operatora `new` alokuje pamięć i pozwala inicjować tablicę wartościami.

Alokacja tablicy

```
1  tablica1 = new int[3]; //memory allocation
2  int[] tablica3 = {0,0,0}; //memory allocation and
       initiation
```

Inicjacja tablicy

- Nowe tablice liczb są inicjowane zerami.
- Elementy logiczne przyjmują wartość `false`
- Obiekty przyjmują specjalną wartość `null`.

Tablice wielowymiarowe

- Tablice są obiektami jednowymiarowymi (wektorami).
- Można stworzyć tablicę wielowymiarową zagnieżdżając tablice wewnątrz tablicy
- Należy pamiętać o konieczności alokacji zagnieżdżonych tablic.

Tablice wielowymiarowe - przykłady

Tablica dwuwymiarowa

```
1  int tablicaDwuWymiarowa[][]; //declaration
2  tablicaDwuWymiarowa = new int[3][]; //allocation
3  tablicaDwuWymiarowa[0] = new int[5]; //add first row
       for second dimension
4  tablicaDwuWymiarowa[1] = new int[5];
5  tablicaDwuWymiarowa[2] = new int[5];
```

Tablica trójkątna

```
1  int[][] tablicaTrojkatna = new int[10][] ;
       //declaration
2  for(int i=0; i<tablicaTrojkatna.length; i++)
3    tablicaTrojkatna[i] = new int[i+1];
```

Tablice wielowymiarowe - inicjalizacja

- Tablicę wielowymiarową można inicjować podczas deklaracji

Inicjalizacja tablicy

```
1  int tablica1[][] = {{0,1,2}, {5,5,5}, {3,2,1}};
```

- Można także odroczyć inicjację elementów

Inicjalizacja tablicy

```
1  int tablica2[][] = {{1}, null , {0,0,1}};
2  tablica2[1] = new int[10];
```

Tablice wielowymiarowe - przeglądanie

Przeglądanie tablicy

```
1  for(int i=0; i<tablica2.length; i++){
2   for(int j=0; j<tablica2[i].length; j++)
3    System.out.print(" ["+tablica2[i][j]+" ]");
4   System.out.println();
5  }
```

Wynik

[1]
[0][0][0][0][0][0][0][0][0][0]
[0][0][1]

Alternatywnie wywołanie

```
1  System.out.println(Arrays.deepToString(tablica2))
```

[[1], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1]]

Przykładowy program

```
1  package pl.edu.pw.mini.mluckner.op.lecture02;
2
3  public class FirstExample {
4
5      private String[] strings;
6
7      public FirstExample(String[] strings) {
8          this.strings = strings;
9      }
10
11     public static void help() {
12         System.out.println("Find the longest String");
13     }
14
15     public int findLongestString() {
16         int longestLength = 0;
17         for(String string:strings) {
18             if(string.length() > longestLength) {
19                 longestLength = string.length();
20             }
21         }
22         return longestLength;
23     }
24
25     public static void main(String[] args) {
26         FirstExample firstExample = new FirstExample(args);
27         help();
28         System.out.println("The longest String has
               "+firstExample.findLongestString()+" chars.");
29     }
30 }
```

Bibliografia

[Horstmann, 2019] Horstmann, C. S. (2019).
Java. Podstawy. Wydanie XI.
Helion.

[Martin, 2010] Martin, R. C. (2010).
Czysty kod. Podręcznik dobrego programisty.
Helion.

[Schildt, 2020] Schildt, H. (2020).
Java. Kompendium programisty. Wydanie XI.
Helion.