

Szerszy wstęp do Javy, nauka o krok dalej niż twoja leniwa dupa chce

Spis treści

Wyskakujące okno dialogowe	2
Funkcja Random, losowe liczby.....	3
Metody pracy na Stringach	4
ArrayList lepsza tablica	5
ArrayList dwuwymiarowy.....	7
Pętla for-each	8
Dziedziczenie w klasach.....	9

Adnotacja: Wszystkie kody zamieszczone na screenach znajdują się na moim githubie: <https://github.com/Rolaski>

Wyskakujące okno dialogowe

Metoda do wpisywania danych w nowym oknie dialogowym. Zmienne typu int, double muszą być konwertowane(parse) ponieważ wszystkie są standardowo w Stringu (trochę jak w JavaScriptcie 😊). Przyjemna metoda wprowadzania

Pamiętaj o komendzie importującej!

```
import javax.swing.JOptionPane;
no usages
public class Main
{
    no usages
    public static void main(String[] args)
    {
        String name = JOptionPane.showInputDialog("Podaj swoje imie");
        JOptionPane.showMessageDialog( parentComponent: null, message: "Hello "+name);

        int age = Integer.parseInt(JOptionPane.showInputDialog("Podaj wiek"));
        JOptionPane.showMessageDialog( parentComponent: null, message: "Masz "+age+" lat");

        double height = Double.parseDouble(JOptionPane.showInputDialog("Podaj wzrost"));
        JOptionPane.showMessageDialog( parentComponent: null, message: "Masz "+height+"cm wzrostu");

        System.out.println(name+" "+age+" "+height);
    }
}
```

Funkcja Random, losowe liczby

Randomowe cyfry wytłumaczenie jak to działa.

W nawiasie masz zakres cyfry np. 6 cyfr łącznie, standardowo pokaże nam od 0 do 5 (bo to w sumie 6 cyfry), +10 czy -10 to operacja dodawania, odejmowania od 0 i wtedy zmieniamy zakres początkowy bo jak napisane wyżej zawsze zaczyna od 0

Pamiętaj o komendzie importującej!

```
import java.util.Random;
no usages
public class RandomNumbers
{
    no usages
    public static void main(String[] args)
    {
        Random randomNumber = new Random();
        int x;
        double y = randomNumber.nextDouble();
        boolean z = randomNumber.nextBoolean();
        for(int i=0; i<=10; i++)
        {
            System.out.print(randomNumber.nextInt( bound: 6)+ " ");
            // wyświetli od 0 do 5; 6 to ilość cyfr
        }
        System.out.println();
        for(int i=0; i<=10; i++)
        {
            System.out.print(randomNumber.nextInt( bound: 10)+11+ " ");
            //10 to zakres, standardowo od 0, dodajemy 11 żeby zaczął od 10
        }
        System.out.println();
        for(int i=0; i<=10; i++)
        {
            System.out.print(randomNumber.nextInt( bound: 21)-10+ " ");
            //<-10,10> leci od -10 i zakres to 21 cyfr
        }
    }
}
```

Metody pracy na Stringach

Praca na Stringach, porównywanie, długość i inne przydatne operacje

Pamiętaj o komendzie importującej!

```
no usages
public class StringMethods
{
    no usages
    public static void main(String[] args)
    {
        String name = "Jacob";
        String name1 = "jacob";
        String name2 = "";
        String name3 = "    Maciek    ";
        boolean result = name.equals("Frajer");
        boolean result1 = name.equalsIgnoreCase(name1);
        //equalsIgnoreCase - polecenie ma w dupie duże i małe znaki

        int WordLength = name1.length();           //długość słowa
        char ThirdLetter = name.charAt(2);          //wyświetla np. 2 znak Stringu name
        int SearchingSign = name.indexOf("o");      //wyświetla nr indeksu konkretnego znaku np. o
        boolean isEmpty = name2.isEmpty();          //sprawdza czy String name2 jest pusty
        String Uppercase = name.toUpperCase();      //zmienia znaki na duże, analogicznie .toLowerCase
        String Trim = name3.trim();                 //usuwa spacje, ucina je
        String Replace = Trim.replace( target: "a", replacement: "o"); //zamiana znaku a na o

        System.out.println(result);
        System.out.println(result1 + " " + WordLength);
        System.out.println("Pierwsza litera słowa " + name + " to: " + ThirdLetter);
        System.out.println("Litera 'o' ma indeks nr." + SearchingSign);
        System.out.println(isEmpty);
        System.out.println("Słowo " + name + " ma teraz duże litery -> " + Uppercase);
        System.out.println("Teraz pozbyliśmy się niepotrzebnych miejsc: " + Trim);
        System.out.println("Maciek po przejściach: " + Replace);
    }
}
```

ArrayList lepsza tablica

Sposób na uzyskanie tablicy, listy tzw. ArrayList. Funkcja która daje nam nowe możliwości i przyjemne do pracy z nią komendy.

Pamiętaj o komendzie importującej!

Tutaj operacje na Stringach:

```
import java.util.ArrayList;
no usages
public class Array_List
{
    no usages
    public static void main(String[] args)
    {
        //nawiasy <> deklarują typ zmiennych i z góry to musi być String
        ArrayList<String> food = new ArrayList<String>();
        food.add("zapiekanka");
        food.add("zapiekanka z dodatkami");
        food.add("hamburger");
        food.add("hot-dog");
        food.add("hot-dog z dodatkami");
        food.set(0, "zapiekanka bez dodatków"); //zamiana indeksu 0 na cos nowego
        food.remove(index: 2); //usunięcie pozycji nr2
        food.clear(); //wyczyszczenie listy food

        for(int i=0; i< food.size(); i++)
        {
            System.out.print(food.get(i)+", ");
        }
    }
}
```

Warto tutaj dodać że pętla for jest **niepotrzebna!**

Komenda `System.out.println(„Tutaj nazwa listy”)` wyświetli nam wszystkie elementy listy

A tutaj przykład jak używać ArrayLista na innych typach niż String, ponieważ została ona stworzona pod Stringi to musimy ją skonwertować do innego typu, poniżej przykład zamiany ArrayLista na inty (Integer), dla zmiennych double to będzie po prostu <Double>

```
import java.util.ArrayList;
no usages
public class Array_List
{
    no usages
    public static void main(String[] args)
    {
        ArrayList<Integer> Age = new ArrayList<>();
        Age.add(56);
        Age.add(19);
        Age.add(34);
        Age.set(2, 79);
    }
}
```

Jak widać ArrayList jest świetnym przykładem pracy na tablicach za pomocą tej sprytnej funkcji, która znacznie ułatwia nam pracę

ArrayList dwuwymiarowy

Świetny przykład zastosowania ArrayList dla tabel dwuwymiarowych. Komendy i ich łatwe użycie jest wygodną możliwością korzystania z nich. Nie musimy tworzyć wielu pętli do tworzenia tablicy czy ich wypisywania co jest idealnym przykładem optymalizacji naszego kodu. Tutaj zamiast dwóch importów wystarczy skorzystać z gwiazdki(*), która pozwala na import wszystkich potrzebnych tam bibliotek.

```
import java.util.*;
no usages
public class Array_List_2D
{
    no usages
    public static void main(String[] args)
    {
        ArrayList<String> bakeryList = new ArrayList<String>();
        bakeryList.add("Mąka");
        bakeryList.add("Pszenica");
        bakeryList.add("Chleb");
        ArrayList<String> produceList = new ArrayList<String>();
        produceList.add("pomidory");
        produceList.add("ogórki");
        produceList.add("kukurydza");
        ArrayList<String> drinksList = new ArrayList<String>();
        drinksList.add("kawa");
        drinksList.add("cappuccino");
        drinksList.add("tymbark");

        ArrayList<ArrayList<String>> groceryList = new ArrayList<>(); //na końcu <> musi być puste!
        groceryList.add(bakeryList);
        groceryList.add(produceList);
        groceryList.add(drinksList);

        System.out.println(bakeryList); //wyświetla całą listę
        System.out.println(bakeryList.get(2)); //wyświetla element o indeksie nr2
        System.out.println(groceryList);
        System.out.println(groceryList.get(1).get(0)); //pierwszy get to nr listy a drugi get to element
    }
}
```

Warto zauważyć że kiedy tworzymy tablice dwuwymiarową po przypisaniu tj. = new ArrayList<>(); specjalnie zostawiamy nawiasy <> puste aby program działał poprawnie, w innym wypadku pokazują się błędy!

Pętla for-each

Jedną z nowych pętli którą warto się nauczyć z powodu jej prostoty jest pętla for-each, która na początku nie wydaje się taka intuicyjna z czasem jest świetnym zastosowaniem za naszego dobrego starego for-a.

Tutaj przykład zastosowania for-each-a do wypisywania danych z zwykłej tablicy oraz naszego ulubionego ArrayLista

```
import java.util.ArrayList;

no usages
public class foreachLoop
{
    no usages
    public static void main(String[] args)
    {
        String[] animals = {"kot", "pies", "bocian", "karp"};

        for(String i: animals)           //w petli foreach elementy to
        {                                //String - po jakich elementach chodzimy
            System.out.print(i+" ");     //i - to nasze typowe i z for-a
        }                                //animals - tablica która używamy
        System.out.println();

        ArrayList<String> food = new ArrayList<String>();
        food.add("Pomarańcza");
        food.add("Pieczarka");
        food.add("Kiwi");
        food.add("Ogórek");

        for(String i: food)
        {
            System.out.print(i+" ");
        }
    }
}
```


Dziedziczenie w klasach

Pojęcie dziedziczenia będzie dotyczyło naszych klas, przydaje nam się żeby zabrać wszystkie obiekty/elementy z klasy A do klasy B i dodajemy jedynie w niej jeden lub parę nowych rzeczy np. wypłata jak w naszym przykładzie. Jeżeli chcemy przekazać obiekty do innej klasy np. obiekt imię, nazwisko możemy do tego wykorzystać dziedziczenie, wykonujemy wtedy rozszerzenie klasy tj. *public class BookKeeping extends Employee*, nasza nowa klasa BookKeeping zawiera teraz obiekty z klasy Employee. Kiedy klasa A posiada obiekty B i nic poza tym mówimy wówczas o **kompozycji**. Zaczynamy od podstawowych obiektów które będą zawierać się wszędzie, czyli od naszego pracownika, w nim tworzymy konstruktor

```
package Inheritance;
2 usages 2 inheritors
public class Employee
{
    8 usages
    String name, surname, profession;
    8 usages
    int paycheck;

    2 usages
    public Employee()
    {
        name = "";
        surname = "";
        profession = "";
        paycheck = 0;
    }

    2 usages
    public Employee(String name, String surname, String profession, int paycheck)
    {
        this.name = name;
        this.surname = surname;
        this.profession = profession;
        this.paycheck = paycheck;
    }
}
```

Następnym naszym krokiem po stworzeniu bazowej klasy jest utworzenie nowej która będzie dziedziczyła elementy. Do tego służą nam dwa polecenia: *extends* i *super*. Tworzymy metodę konstruktora do której przekazujemy wraz z podaniem typu zmiennej obiektów, w środku metody wpisujemy komendę *super*(obiekt1, obiekt2, ...), gdzie obiekt to nazwa obiektu którą dziedziczymy z góry.

Nie zapomnij o dodaniu w konstruktorze odwołania do nowych obiektów jeśli takie stworzysz! (*this.obiekt = obiekt*).

```
package Inheritance;
import javax.swing.*;

2 usages
public class Boss extends Employee
{
    4 usages
    int bonus;
    4 usages
    int monthlyTax;

    1 usage
    public Boss()
    {
        bonus = 0;
        monthlyTax = 0;
    }

    no usages
    public Boss(String name, String surname, String profession, int paycheck, int bonus, int monthlyTax)
    {
        super(name, surname, profession, paycheck);
        this.bonus = bonus;
        this.monthlyTax = monthlyTax;
    }
}
```

```
package Inheritance;
import javax.swing.*;

2 usages
public class BookKeeping extends Employee
{
    8 usages
    int bonus;
    1 usage
    public BookKeeping()
    {
        bonus = 0;
    }

    no usages
    public BookKeeping(String name, String surname, String profession, int paycheck)
    {
        super(name, surname, profession, paycheck);
        this.bonus = bonus;
    }
}
```