Programowanie komputerów I

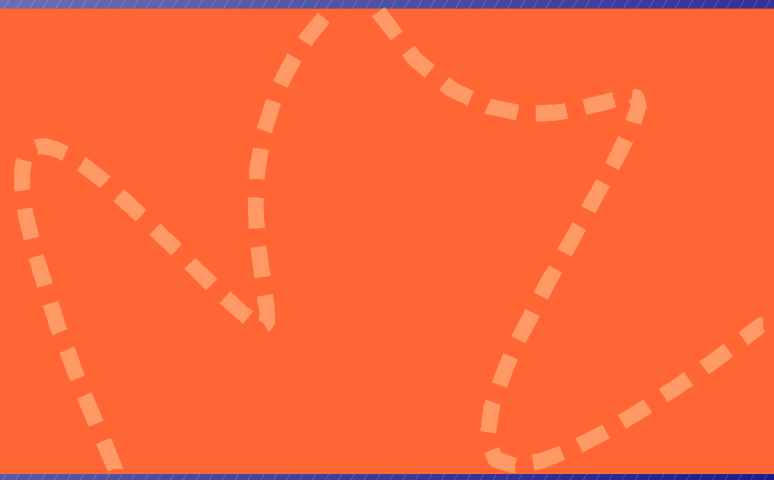
Instrukcje podstawowe i złożone

Willy Picard
Katedra Technologii Informacyjnych
Akademia Ekonomiczna w Poznaniu
<*picard@kti.ae.poznan.pl*>

Agenda

- Cel(e) wykładu
- Zmienne i zakres
- Podstawowe instrukcje
- Instrukcje warunkowe
- Pętle
- Instrukcje rozgałęzienia
- Podsumowanie

Cel(e) wykładu



Przegląd wykładu

Podstawowe pojęcia

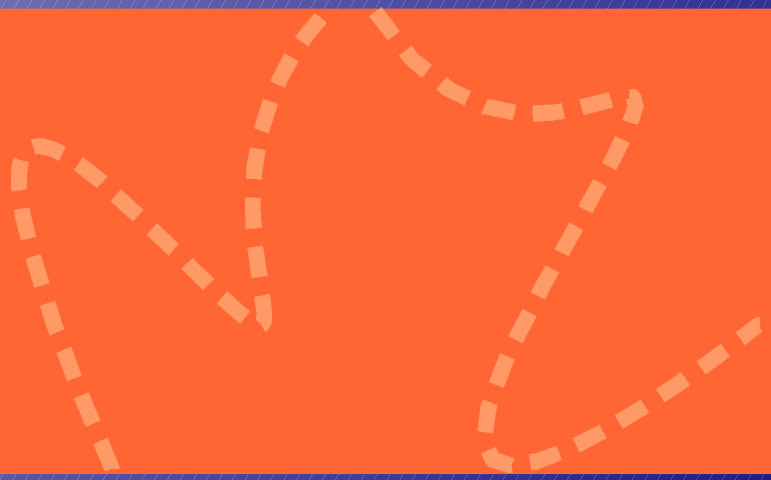
- 1: Wprowadzenie
- 2: Podstawowe struktury danych & instrukcje
- 3: Programowanie obiektowe I
- 4: Programowanie obiektowe II
- 5: Programowanie obiektowe III
- 6: Zaawansowane struktury danych
- 7: Wątki & Wyjątki

CELE WYKŁADU

Cel na dziś

- Przedstawić instrukcje

Zmienne i zakres



Definicja zmiennej

Zmienna jest symbolem
o ustalonej nazwie,
któremu mogą być
przypisane dane

Identyfikatory w Javie

- Ciąg o nieograniczonej długości
  - Liter *A-Z, a-z, _, $*
  - Cyfr *0-9*
- Pierwszy element ciągu: litera
- Uwaga!
  - Słowa kluczowe
  - Literały *true* lub *false*
  - null *null*

Słowa kluczowy w Javie

abstract	double	int	strictfp
boolean	else	interface	super
break	extends	long	switch
byte	final	native	synchronized
case	finally	new	this
catch	float	package	throw
char	for	private	throws
class	goto	protected	transient
const	if	public	try
continue	implements	return	void
default	import	short	volatile
do	instanceof	static	while

Przekłady identyfikatorów w Javie

- Poprawne
  - `mojaZmienna`
  - `Moja2gaZmienna` (lepiej: `mojaDrugaZmienna`)
  - `_wewnętrznaZmienna`
  - `_kocham_znak_podkreszlenia`
  - `$zmiennaDoStaregoSystemu`
- Błędne
  - `1Zmienna`
  - `vice-versa`

Definicja zmiennej w Javie

- Z przypisaniem wartości
  - *typ nazwa = wartość;*

```
int licznik = 0;
```

- Bez przypisania wartości
  - *typ nazwa;*
  - *nazwa = wartość;*

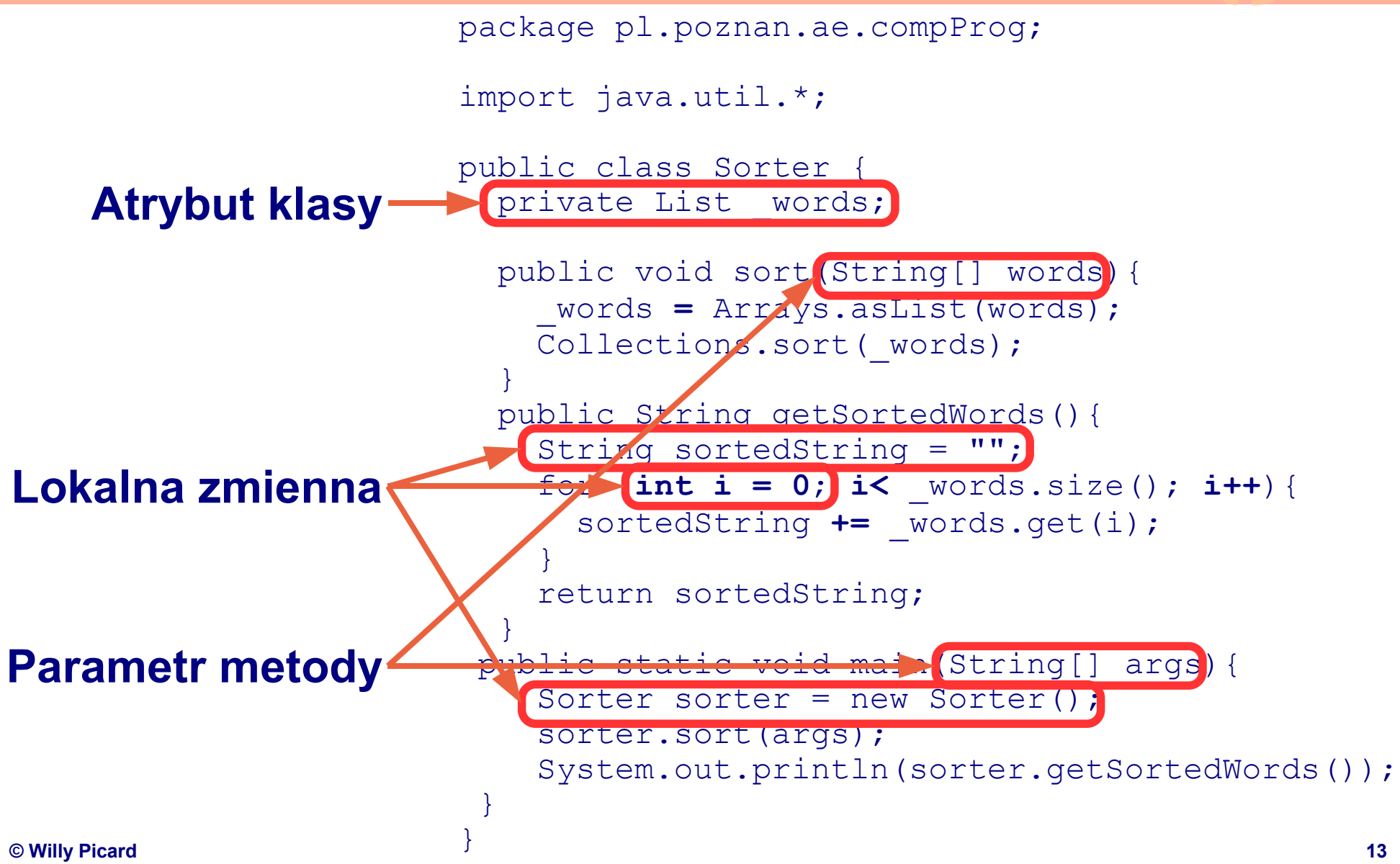
```
int licznik;
licznik = 0;
```

Klasyfikacja zmiennych w Javie

- Cztery kategorie
  - Atrybut klasy
  - Lokalna zmienna
  - Parametr metody
  - Parametr obsługi wyjątku

```
class MojaKlasa {
    ...
    deklaracja atrybutu klasy
    ...
    public void mojaMetoda(parametry metody) {
        ...
        deklaracja lokalnej zmiennej
        ...
        catch (parametr obsługi wyjątku) {
            ...
        }
        ...
    }
    ...
}
```

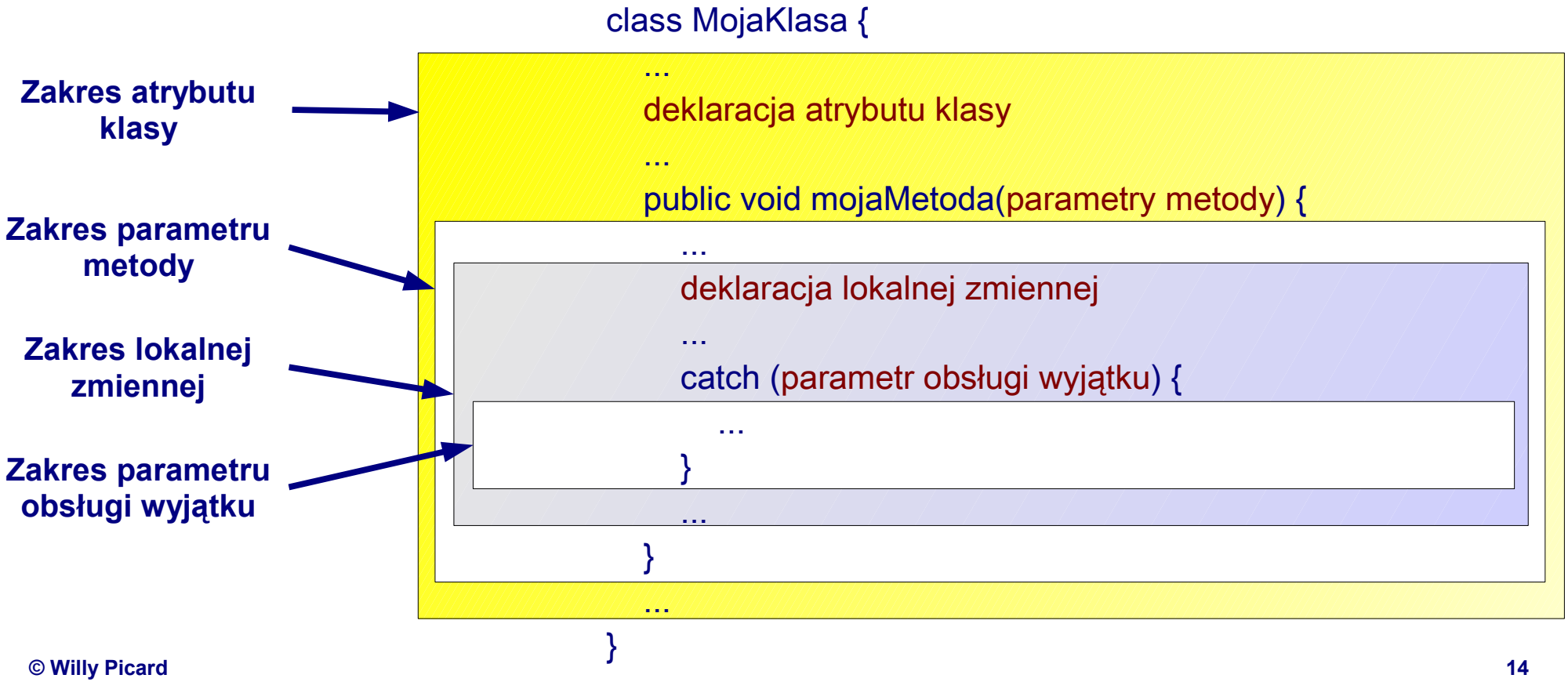
Przykład



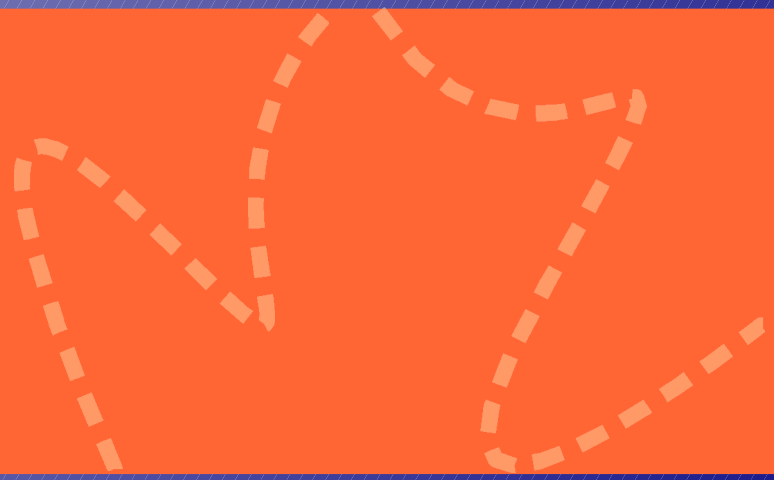

```
package pl.poznan.ae.compProg;

import java.util.*;

public class Sorter {
  private List _words;

  public void sort(String[] words){
    _words = Arrays.asList(words);
    Collections.sort(_words);
  }
  public String getSortedWords(){
    String sortedString = "";
    for(int i = 0; i< _words.size(); i++){
      sortedString += _words.get(i);
    }
    return sortedString;
  }
  public static void main(String[] args){
    Sorter sorter = new Sorter();
    sorter.sort(args);
    System.out.println(sorter.getSortedWords());
  }
}
```

Zakres

- Skąd można odwołać się do danej zmiennej?
- Zakres ≠ widoczność



Podstawowe instrukcje



Definicja instrukcji

Instrukcja jest
częścią
programu, którą
można uruchomić

Pusta instrukcja

- Składnia: ;
- Nic nie robi(!)
- Zawsze wykonuje się bez błędu

Definicja wyrażenia

Wyrażenie jest ciągiem zmiennych, operatorów i odwołań do metod, którego ewaluacja jest pojedynczą wartością

Przykłady wyrażeń

- `licznik = 0`
- `licznik++`
- `resetować(licznik)`
- `licznik + wartośćMinimalna`

Instrukcje wyrażeniowe

- `<wyrażenie>` **;** jest `<instrukcją>`
- Tylko niektóre wyrażenia
  - Wyrażenia przypisania
  - `++` lub `--`
  - Odwołania do metod
  - Wyrażenia tworzenia obiektów

Przykłady instrukcji wyrażeniowych

- `licznik = 0;`
- `licznik++;`
- `resetować(licznik);`
- `licznik + wartośćMinimalna;`

Definicja bloku

Blok jest ciągiem zero lub więcej wyrażeń między nawiasami klamrowymi

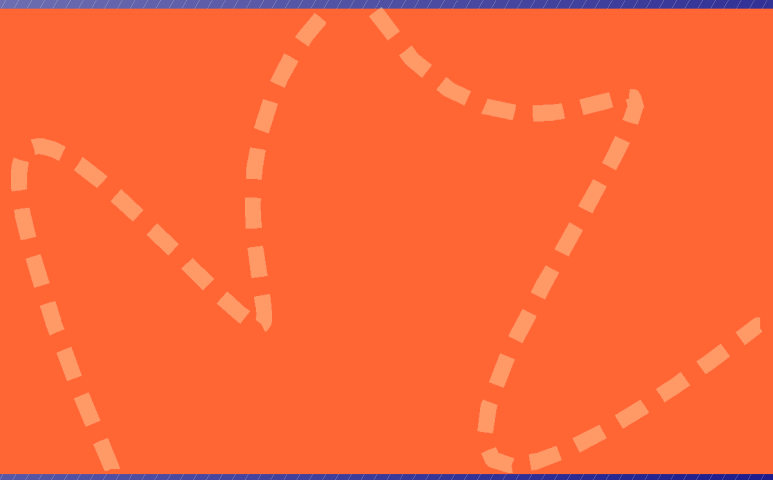
Pusty blok

- Składnia: { }
- Nic nie robi(!)
- Zawsze wykonuje się bez błędu

Przykład bloków

```
{
   licznik = 0;
   licznik++;
   {
      resetować(licznik);
      licznik+wartośćMinimalna;
   }
}
```

Instrukcje warunkowe



Definicja instrukcji warunkowej

Instrukcja warunkowa wykonuje różne instrukcje w zależności od wartości danego predykatu

Instrukcje warunkowe w Javie

- Dwie instrukcje
  - `if-else`
  - `switch-case`
- `if-else`
  - alternatywa
- `switch-case`
  - wybór

Instrukcja *If-else*

- Dwie postaci
  - `if (warunek) instrukcja`
  - `if (warunek) instrukcja1 else instrukcja2`
- Warunek
  - Wyrażenie logiczne
- Instrukcje
  - Podstawowe
  - Złożone

Przykład *If-else*

```
if ( (liczba % 2) == 0) {
   System.out.println(liczba+" jest parzysta");
} else {
   System.out.println(liczba+" jest nieparzysta");
}
```

Instrukcja *Switch-case*

- Jedna postać

```
switch (<warunek>) {
   case wartość: instrukcja;
   opcjonalnie default: instrukcja;
 }
```

- Warunek
  - Wyrażenie: char, byte, short, lub int
- Instrukcje
  - Proste
  - Złożone

Przykład *Switch-case*

```
switch (liczba % 3)  {
   case 0:
     System.out.println(liczba+" można dzielić przez 3");
     break;
   case 1:
     System.out.println("Liczba%3=1");
     break;
   case 2:
     System.out.println("Liczba%3=2");
     break;
}
```

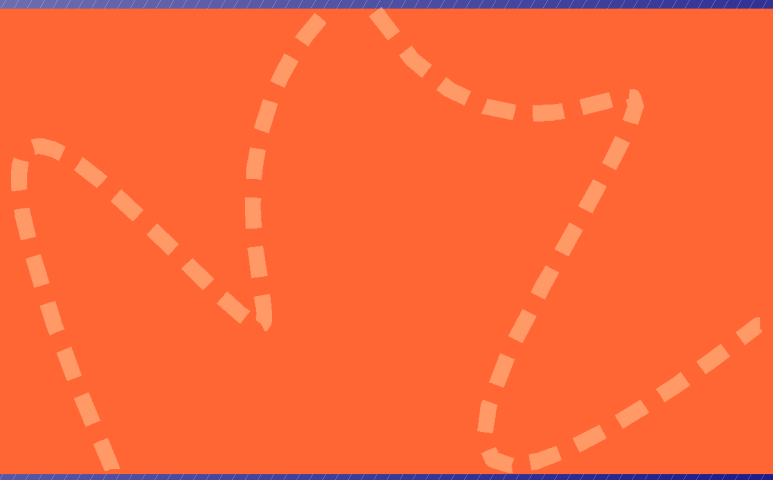
„Let's Have a *Break*”

```
switch (dzień)  {
   case 1:
   case 2:
   case 3:
   case 4:
   case 5:
     System.out.println(“Praca”);
     break;
   case 6:
   case 7:
     System.out.println(“Siesta!”);
     break;
}
```

Default

```
switch (dzień)  {
   case 1:
   case 2:
   case 3:
   case 4:
   case 5:
     System.out.println("Praca");
     break;
   case 6:
   case 7:
     System.out.println("Siesta!");
     break;
   default:
     System.out.println("Z której planety jesteś?");
}
```

Pętle



Definicja instrukcji pętli

Instrukcja pętli
wykonuje w sposób
iteracyjny
instrukcje

Instrukcje pętli w Javie

- Trzy instrukcje
  - `while`
  - `do-while`
  - `for`

Instrukcja *While*

- Jedna postać
  - `while (warunek) instrukcja`
- Warunek
  - Wyrażenie logiczne
- Instrukcje
  - Proste
  - Złożone

Przykład *While*

```
int licznik = 0;
while (licznik != 3) {
   System.out.println("Licznik: "+licznik);
   licznik++;
}
```

Instrukcja *Do-while*

- Jedna postać
  - `do instrukcja while (warunek);`
- Warunek
  - Wyrażenie logiczne
- Instrukcje
  - Proste
  - Złożone

Przykład *Do-while*

```
int licznik = 0;
do {
   System.out.println("Licznik: "+licznik);
   licznik++;
} while (licznik !=3);
```

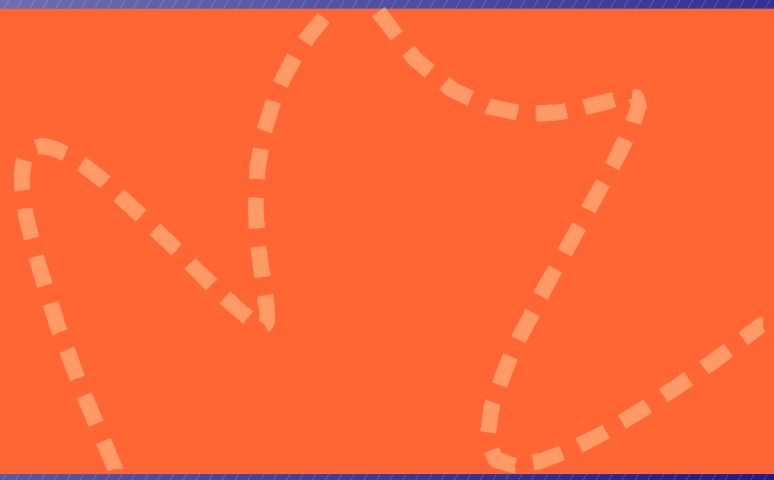
Instrukcja *For*

- Jedna postać
  - `for (init; warunek; zmiana) instrukcja`
- Wyrażenia opcjonalne
  - inicjalizacja
  - warunek
  - zmiana
- Instrukcje
  - Proste
  - Złożone

Przykład *For*

```
int mojaWartość = 10;
for (int i = 0; i < mojaWartość; i+=2){
   System.out.println(mojaWartość+“ > ”+i);
}
```

Instrukcje rozgałęzienia



Definicja instrukcji rozgałęzienia

Instrukcja rozgałęzienia
zmienia kolejność
wykonania instrukcji

Instrukcje rozgałęzienia w Javie

- Trzy instrukcje
  - `return`
  - `break`
  - `continue`

Instrukcja *Return*

- Dwie postaci
  - `return;`
  - `return <wartość>;`
- Wyjście z metody

Przykład *Return*

```
public int dzielPrzezTrzy(int liczba) {
  switch (liczba % 3)  {
     case 0:
       System.out.println(liczba+“ można dzielić przez 3”);
       return liczba;
     case 1:
       System.out.println(liczba+”%3=1”);
       break;
     case 2:
       System.out.println(liczba+”%3=2”);
       break;
   }
   return 0;
}
```

Instrukcja *Break*

- Dwie postaci
  - `break;`
  - `break <etykieta>;`
- Kończy instrukcje
  - `switch`
  - `while`
  - `do-while`
  - `for`

Przykład *Break*

```
switch (liczba % 3)  {
   case 0:
     System.out.println(liczba+" można dzielić przez 3");
     break;
   case 1:
     System.out.println(liczba+"%3=1");
     break;
   case 2:
     System.out.println(liczba+"%3=2");
     break;
}
```

Przykład *Break* z etykietą

```
int pierwsza, druga;
oblicz:
   for (int i = 0; i<10 ; i++){
      for (int j = 0; j< i; j++) {
         int suma = i+j;
         if ( suma > 15 ){
            pierwsza = i;
            druga = j;
            break oblicz;
         }
      }
   }
System.out.println("16="+pierwsza+"+"+druga);
```

Instrukcja *Continue*

- Dwie postaci
  - `continue;`
  - `continue <etykieta>;`
- Kończy iterację instrukcji
  - `while`
  - `do-while`
  - `for`

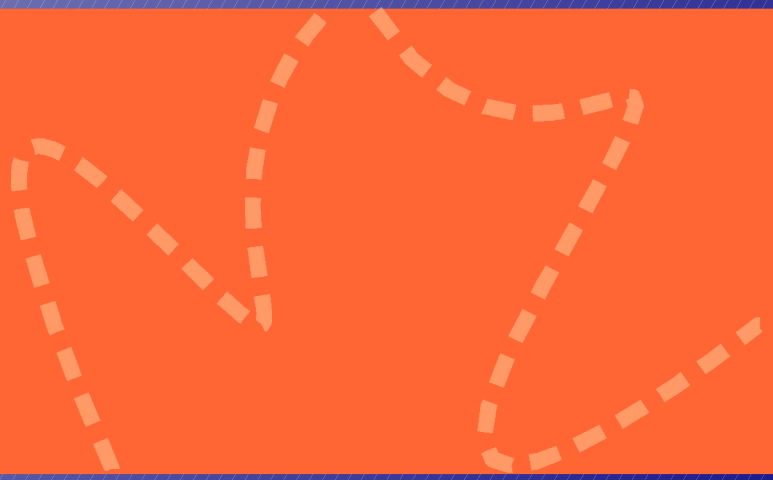
Przykład *Continue*

```
for (int i = 0; i<100 ; i++){
      if ( (i % 7) != 0 )
         continue;
      System.out.println(i+" można dzielić przez 7");
}
```

Przykład *Continue* z etykietą

```
oblicz:
   for (int i = 0; i<10 ; i++){
      for (int j = 0; j< i; i++) {
         int suma = i+j;
         if ( suma > 15 ){
            System.out.println("15="+i+"+"+j);
            continue oblicz;
         }
      }
   }
```

Podsumowanie



Podsumowanie

- Zmienne
- Podstawowe instrukcje
  - Wyrażenia
  - Bloki
- Instrukcje przepływu sterowania
  - Instrukcje warunkowe
  - Pętle
  - Instrukcje rozgałęzienia

Przykład

```
package pl.poznan.ae.compProg;

import java.util.*;

public class Sorter {
  private List _words;

  public void sort(String[] words){
     _words = Arrays.asList(words);
     Collections.sort(_words);
  }
  public String getSortedWords(){
     String sortedString = "";
     for (int i = 0; i< _words.size(); i++){
       sortedString += _words.get(i);
     }
     return sortedString;
  }
 public static void main(String[] args){
     Sorter sorter = new Sorter();
     sorter.sort(args);
     System.out.println(sorter.getSortedWords());
 }
}
```

Do zobaczenia za tydzień