Programowanie komputerów I

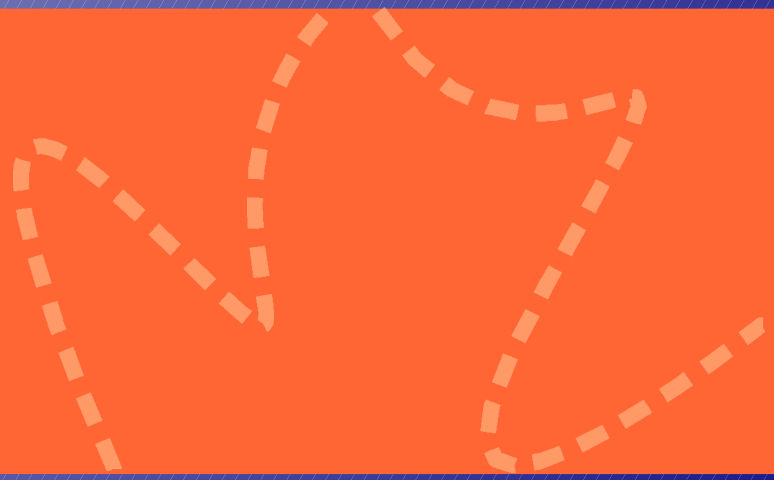
Tablice i kontenery

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

Agenda

- Cel(e) wykładu
- Odświeżenie i przekąski
- Tablice
- Kontenery
- The Java™ Collections Framework
- Podsumowanie

Cel(e) wykładu



Przegląd wykładu

Podstawowe pojęcia

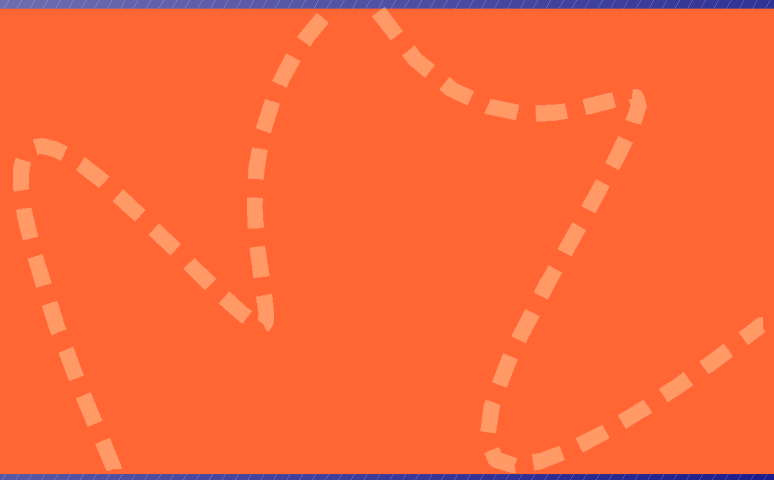
- 1: Wprowadzenie
- 2: Podstawowe struktury danych & instrukcje
- 3: Programowanie obiektowe I
- 4: Programowanie obiektowe II
- 5: Programowanie obiektowe III
- 6: Zaawansowane struktury danych
- 7: Wątki & Wyjątki

CELE WYKŁADU

Cel na dziś

Wprowadzić
kontenery do
przechowywania obiektów

Odświeżenie i przekąski



Przykład interfejsu

```
class IZwierzę{
        int zwróćWagę();
        String zwróćNazwę();
        void wydajGłos();
        void jedz();
        void jedz(int ilość);
}
```

Przykład klasy

```
class Kot implements IZwierzę{
        int _waga;
        String _nazwa;
        Kot(int waga, String nazwa){
            _waga = waga;
            _nazwa = nazwa;
        }
       ...
}
Kot mójKot = new Kot(1200, "Felix");
```

ODŚWIŻENIE

Przykład klasy

```
class Kot implements IZwierzę{
        int _waga;
        String _nazwa;
        int zwróćWagę(){
            return _waga;
        }
        String zwróćNazwę(){
            return _nazwa;
        }
        ...
}
```

Przykład klasy

```
class Kot implements IZwierzę{
      ...
      void wydajGłos(){
         System.out.println("Miau");
      }
      void jedz(){
          _waga += 200;
       }
       void jedz(int ilość){
          _waga += ilość;
       }
}

System.out.prinln(mójKot.zwróćNazwę()+ " mówił:");
mójKot.wydajGłos());
```

ODŚWIŻENIE

Nadpisanie metody w Javie

```
class Kot {
   void jedz(){
         _waga +=200;
   }
}

class KotPerski extends Kot {
   boolean _czyŚpi = false;

   void jedz(){
         super.jedz();
         zróbSiestę();
   }

   void zróbSiestę(){ _czyŚpi = true;}
}
```

System.out.println()

- W pakiecie `java.lang`
- `System` : klasa
- `out` : atrybut klasy `System`
- `System.out` : instancja klasy `java.io.PrintStream`
- `println()` : metoda klasy `java.io.PrintStream`

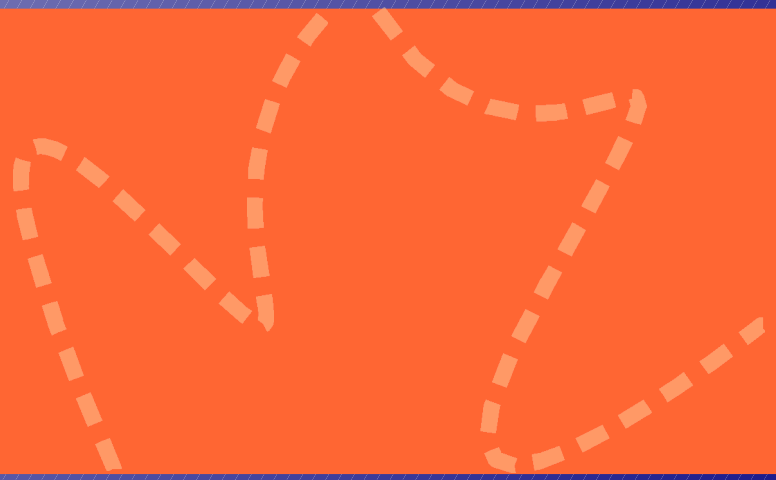
ODŚWIŻENIE

The API Specification

- Dokumentacja
  - Pakietów
  - Interfejsów
  - Klas
  - Dziedziczenia
  - Atrybutów
  - Metod

ODŚWIŻENIE

Tablice

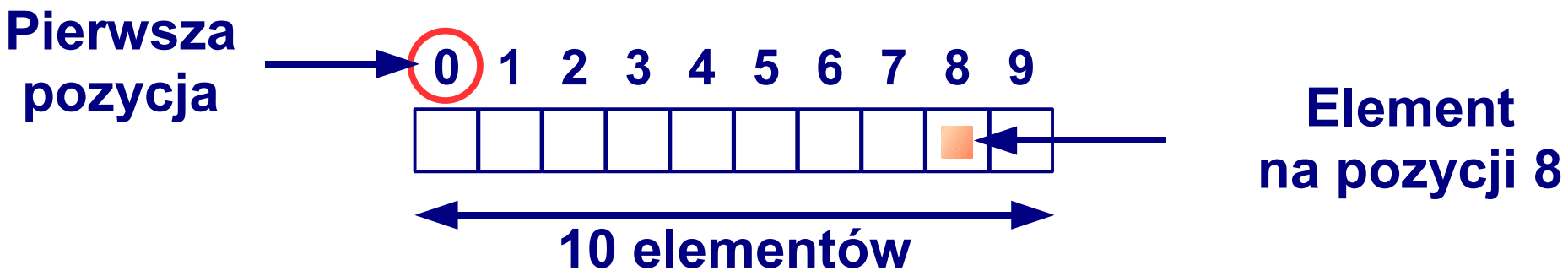


String[] args

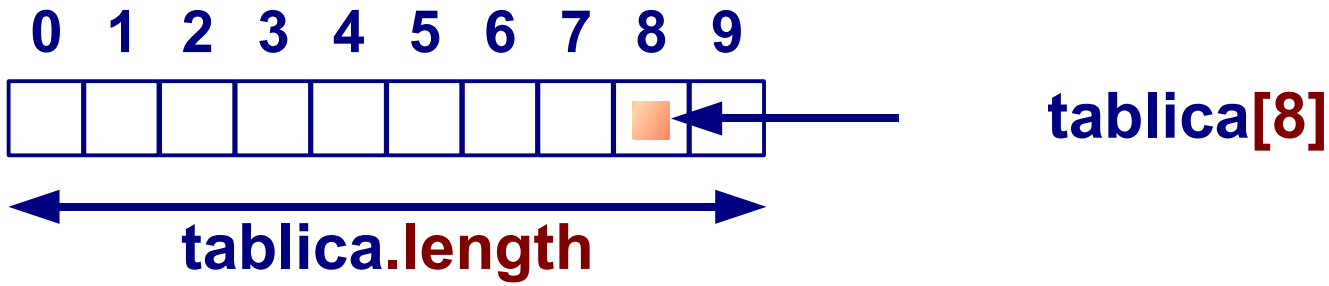
```
public class Kot{
    public static void main(String[] args){...}
}
```

- Uruchomiona przez `java Kot`
- Argumenty
  - `String[] args`
  - Liczba argumentów: `args.length`
- Przykład
  - `java Kot "Felix" "1200"`
  - `args[0] = "Felix", args[1] = "1200"`

Tablica



Tablice w Javie



Tworzenie tablic w Javie

- Składnia
  - Z późną deklaracją rozmiaru
    - `<typ>[] <nazwa>;`
    - `<nazwa> = new <typ>[<rozmiar>];`
  - „Dwa w jednym”
    - `<typ>[] <nazwa> = new <typ>[<rozmiar>];`
- Przykład
  - `int[] mojeLiczbe;`
  - `mojeLiczbe = new int[3];`
  - `int[] mojeLiczbe = new int[3];`

Używanie tablic w Javie

- Składnia
  - `<nazwa>[indeks] = <nowaWartość>;`
- Przykład
  - `mojeLiczbe[2] = 0;`
- Rozmiar tablicy
  - `<nazwa>.length`
- Przykład
  - `mojeLiczbe.length`

Zaawansowane tablice w Javie

- Tablice obiektów
  - `Kot[] mojeKoty = new Kot[2];`
  - `Kot felix = mojeKoty[0];`
- Tablice tablic
  - `int[][] mojeLiczbe = new int[2][];`
  - `mojeLiczbe[0] = new int[3];`
  - `mojeLiczbe[1] = new int[1];`
  - `int[] dodatneLiczbe = mojeLiczbe[0];`
  - `dodatneLiczbe[2] = 13;`
  - `mojeLiczbe[0][2] = 13;`

Tworzenie tablicy inaczej

- Składnia
  - `<typ>[] <nazwa> = {<wartość1>, <wartość2>};`
- Przykłady
  - `int[] mojeLiczbe = { 3, 7, 13};`
  - ```
    IZwierzę[] zwierzęta = {
                        new Kot("Felix", "1200"),
                        new Kot("Garfield", "5000"),
                        new Pies("Scooby", "7000") };
    ```
    ```

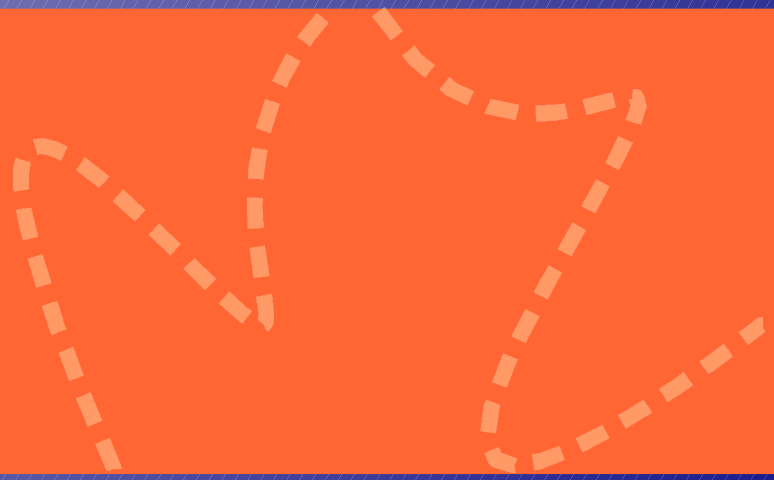
Przykład

Zwierzęta I

Ocena tablicy

- Za
  - Szybkie
- Przeciw
  - Stały rozmiar

Kontenery



Definicja kontenera

Kontener jest obiektem, który przechowuje potencjalnie wiele obiektów

Podstawowe typy kontenerów

- Zbiór (ang. *Set*)
  - np. kolekcja płyt
- Lista (ang. *List*)
  - np. dni tygodnia
- Mapa (ang. *Map*)
  - np. książka adresowa

Definicja zbioru

Zbiór jest nieuporządkowanym kontenerem, w którym dany obiekt występuje najwyżej jeden raz

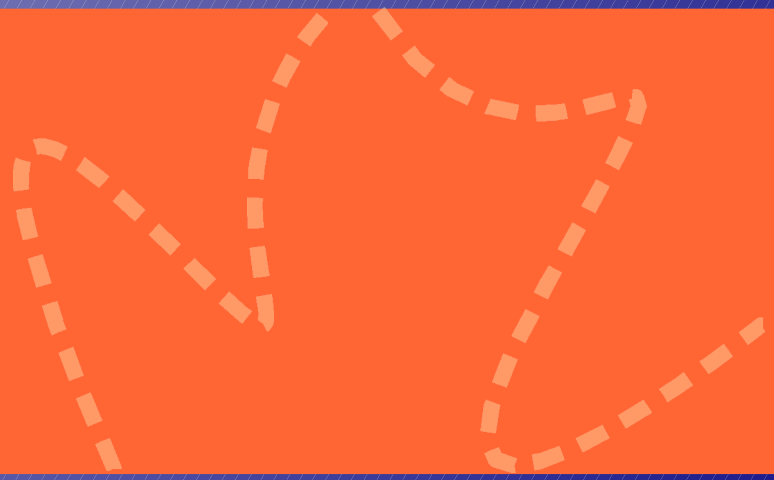
Definicja listy

Lista jest uporządkowanym kontenerem

Definicja mapy

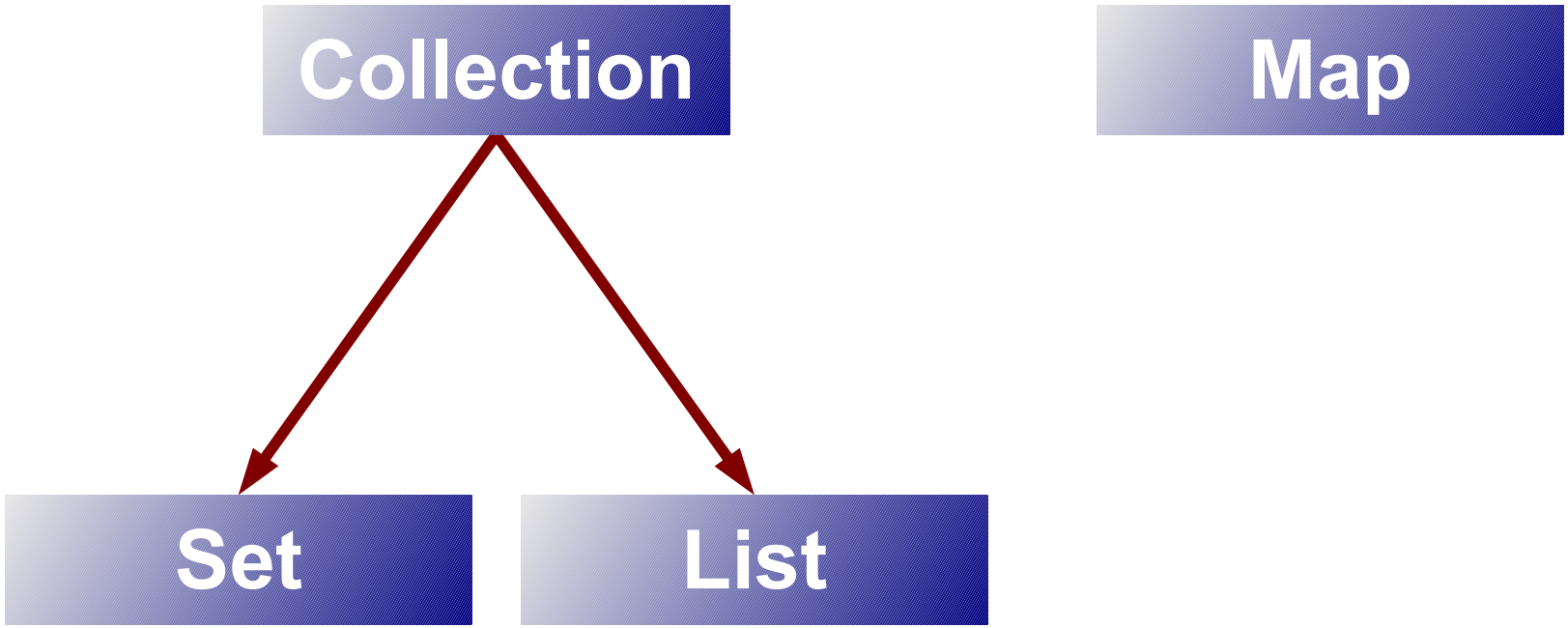
Mapa jest kontenerem, który przechowuje pary (klucz, wartość)

The Java™ Collections Framework



Zbiór interfejsów

- W pakiecie `java.util`



Interfejs *Collection*

- Operacje podstawowe

```
int size();
boolean isEmpty();
boolean contains(Object element);
boolean add(Object element);
boolean remove(Object element);
Iterator iterator();
```

- Operacje hurtowe

```
boolean containsAll(Collection c);
boolean addAll(Collection c);
boolean removeAll(Collection c);
boolean retainAll(Collection c);
void clear();
```

- Operacje na tablicach

```
Object[] toArray();
Object[] toArray(Object a[]);
```

Interfejs *Iterator*

```
public interface Iterator {
        boolean hasNext();
        Object next();
        void remove();
}
```

- Przykład

```
Collection zwierzęta;
for (Iterator i = zwierzęta.iterator();i.hasNext();){
   IZwierzę zwierzę = (IZwierzę) i.next();
   System.out.println("Znalezione "+
                      zwierzę.zwróćNazwę());
}
```

Interfejs *Set*

- Operacje podstawowe

```
int size();
boolean isEmpty();
boolean contains(Object element);
boolean add(Object element);
boolean remove(Object element);
Iterator iterator();
```

- Operacje hurtowe

```
boolean containsAll(Collection c);
boolean addAll(Collection c);
boolean removeAll(Collection c);
boolean retainAll(Collection c);
void clear();
```

- Operacje na tablicach

```
Object[] toArray();
Object[] toArray(Object a[]);
```

Interfejs *List*

- Manipulacja i dostęp

```
Object get(int index);
Object set(int index, Object element);
void add(int index, Object element);
Object remove(int index);
abstract boolean addAll(int index, Collection c);
```

- Wyszukiwanie

```
int indexOf(Object o);
int lastIndexOf(Object o);
```

- Iteracje

```
ListIterator listIterator();
ListIterator listIterator(int index);
```

- Podlista

```
List subList(int from, int to);
```

Interfejs *ListIterator*

```
public interface ListIterator
        extends Iterator{
    boolean hasNext();
    Object next();
    boolean hasPrevious();
    Object previous();
    int nextIndex();
    int previousIndex();
    void remove();
    void set(Object o);
    void add(Object o);
}
```

Interfejs *Map*

- Operacje podstawowe

```
Object put(Object key, Object value);
Object get(Object key);
Object remove(Object key);
boolean containsKey(Object key);
boolean containsValue(Object value);
int size();
boolean isEmpty();
```

- Operacje hurtowe

```
void putAll(Map t);
void clear();
```

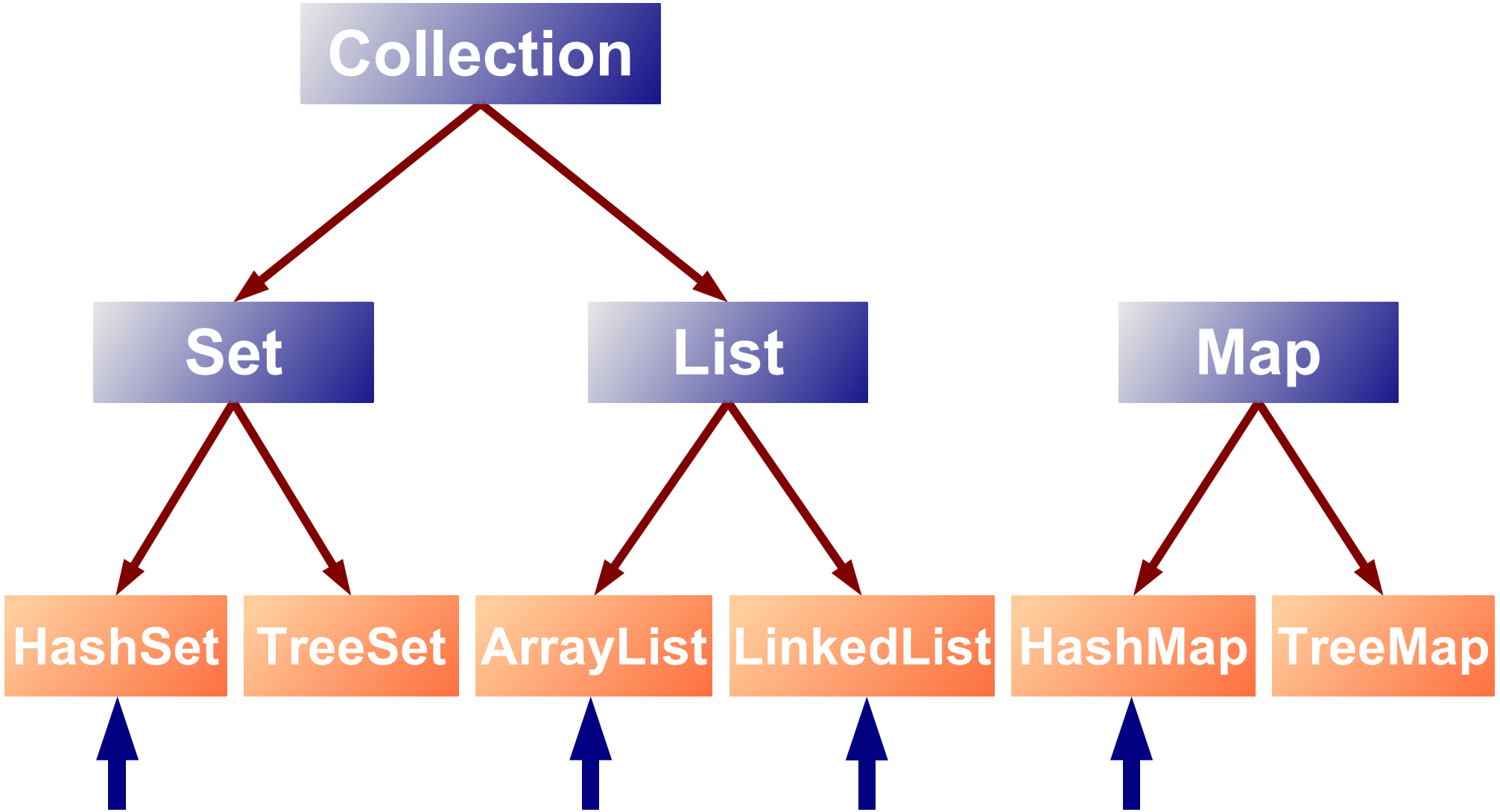
- Dostęp do podkontenerów

```
public Set keySet();
public Collection values();
public Set entrySet();
```

Interfejs *Map.Entry*

```
public interface Entry {
    Object getKey();
    Object getValue();
    Object setValue(Object value);
}
```

Implementacje



Od tablic do kontenerów

- Klasa `Arrays`
  - Metoda `List Arrays(Object[])`
- Przykład

```
List listaZwierząt = Arrays.asList(zwierzęta);
for (int i = 0; i< listaZwierząt.size(); i++){
   Object obj = listaZwierząt.get(i);
   IZwierzę zwierzę = (IZwierzę) obj;
   System.out.println("Witam "+
                        zwierzę.zwróćNazwę());
}
```

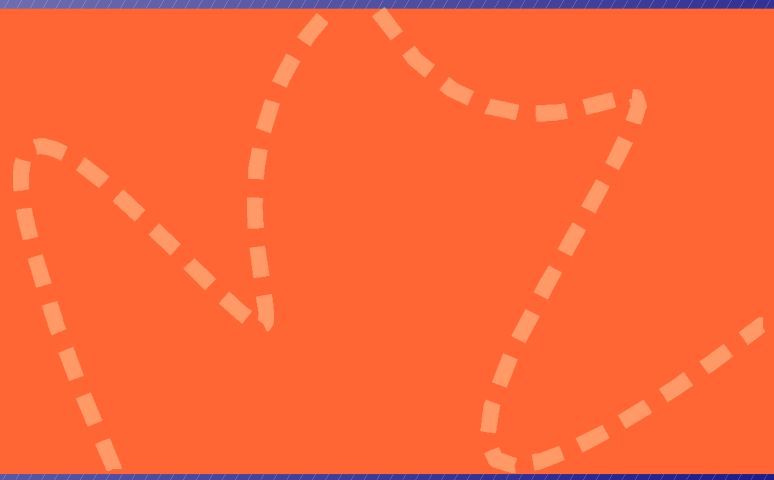
Klasa *Collections*

- Zbiór przydatnych funkcji
- Tasowanie (ang. *Shuffle*)
- Odwrócenie (ang. *Reverse*)
- Posortowanie (ang. *Sorting*)
  - Metoda `sort(List)`
  - Metoda `sort(List, Comparator)`
- Dwie możliwości
  - Interfejs `Comparable`
  - Interfejs `Comparator`

Przykład kontenerów

Animals II

Podsumowanie



Złote reguły

- Reguła 1
  - Używaj interfejsy
- Reguła 2
  - Używaj interfejsy
- Reguła 3
  - Używaj interfejsy

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