Kurs programowania

Wykład 9

Wojciech Macyna

28 kwiecień 2016

Java Collections Framework (w C++ Standard Template Library)

Kolekcja (kontener)

Obiekt grupujący/przechowujący jakieś elementy (obiekty lub wartości). Przykładami kolekcji są zbiór, lista czy wektor.

Składniki

- Interfejsy
- Implementacje (klasy)
- Algorytmy
- Iteratory

Przykład - Zbiory w Javie

Interfejs `java.util.Set` (`java.util.SortedSet`)

Zbiór elementów bez powtórzeń.

Przykładowe Implementacje

- `java.util.HashSet`
- `java.util.LinkedHashSet`
- `java.util.TreeSet` (automatycznie posortowane)

Klasa `java.util.Collections`

Zawiera pomocnicze metody statyczne do operowania na kolekcjach (zobacz także klasę `java.util.Arrays` dla tablic). Szczegółowy spis metod w dokumentacji.

Przykład użycia - zbior.java

```
1  import java.util.Collections;
2  import java.util.Set;
3  import java.util.HashSet;
4
5  public class zbior {
6
7    public static void main(String args[]) {
8      Set<String> nazwiska = new HashSet<String>();
9      nazwiska.add("Nowak"); nazwiska.add("Kowalski");
10     nazwiska.add("Bielecki"); nazwiska.add("Adamski");
11     nazwiska.add("Kowalski");
12
13     for(String s : nazwiska) System.out.println("-> "+s);
14
15     System.out.println(Collections.min(nazwiska));
16     System.out.println(Collections.max(nazwiska));
17
18     nazwiska.remove("Kowalski"); System.out.println(nazwiska);
19     System.out.println(nazwiska.contains("Kowalski"));
20   }
21 }
```

Przykład - Listy w Javie

Interfejs `java.util.List`

Uporządkowany zbiór elementów, z dostępem przez indeks, można używać jak tablicy.

Przykładowe Implementacje

- `java.util.ArrayList`
- `java.util.LinkedList`
- `java.util.Vector`

Przykład użycia - lista.java

```
1  import java.util.Collections;
2  import java.util.List;
3  import java.util.ArrayList;
4  public class lista {
5    public static void main(String args[]) {
6      List<String> nazwiska = new ArrayList<String>();
7      nazwiska.add("Nowak"); nazwiska.add("Kowalski");
8      nazwiska.add("Bielecki"); nazwiska.add("Adamski");
9      nazwiska.add("Kowalski");
10     for(String s : nazwiska) System.out.println("-> "+s);
11     Collections.sort(nazwiska); System.out.println(nazwiska);
12     Collections.shuffle(nazwiska); System.out.println(nazwiska);
13     Collections.reverse(nazwiska); System.out.println(nazwiska);
14     System.out.println(Collections.min(nazwiska));
15     System.out.println(Collections.max(nazwiska));
16
17     System.out.println(nazwiska.get(2));
18     nazwiska.remove(2); System.out.println(nazwiska);
19     nazwiska.remove("Adamski"); System.out.println(nazwiska);
20     System.out.println(nazwiska.contains("Kowalski"));
21   }
22 }
```

Podobny przykład w C++ – lista.cpp

```
1  #include<iostream>
2  #include<list>
3  #include<string>
4  using namespace std;
5  int main(int argc, char* argv[]) {
6    list<string> *nazwiska = new list<string>();
7    nazwiska->push_back("Nowak"); nazwiska->push_back("Kowalski");
8    nazwiska->push_back("Bielecki"); nazwiska->push_back("Adamski");
9    nazwiska->push_back("Kowalski");
10   list<string>::iterator it;
11   for(it=nazwiska->begin(); it!=nazwiska->end(); it++)
12     cout << "␣-␣" << *it << endl;
13   nazwiska->sort();
14   for(it=nazwiska->begin(); it!=nazwiska->end(); it++)
15     cout << "␣+␣" << *it << endl;
16   nazwiska->reverse();
17   for(it=nazwiska->begin(); it!=nazwiska->end(); it++)
18     cout << "␣-␣" << *it << endl;
19   nazwiska->remove("Kowalski");
20   for(it=nazwiska->begin(); it!=nazwiska->end(); it++)
21     cout << "␣+␣" << *it << endl;
22   delete(nazwiska);
23 }
```

Przykład - Mapy w Javie

Interfejs `java.util.Map` (`java.util.SortedMap`)

Kolekcja przechowująca pary *klucz-wartość*, inaczej tablica asocjacyjna.

Przykładowe Implementacje

- `java.util.HashMap`
- `java.util.HashTable`
- `java.util.SortedMap` (automatycznie posortowane)
- `java.util.TreeMap`

Przykład użycia - mapa.java

```
1  import java.util.Collections;
2  import java.util.Map;
3  import java.util.TreeMap;
4
5  public class mapa {
6
7    public static void main(String args[]) {
8      Map<String,Integer> nazwiska = new TreeMap<String,Integer>();
9      nazwiska.put("Nowak",1); nazwiska.put("Kowalski",2);
10     nazwiska.put("Bielecki",1); nazwiska.put("Adamski",2);
11     nazwiska.put("Kowalski",1);
12
13     for(String s : nazwiska.keySet())
14       System.out.println("-> "+s+" - "+nazwiska.get(s));
15     for(int s : nazwiska.values()) System.out.println("-> "+s);
16
17     System.out.println(nazwiska.get("Kowalski"));
18     nazwiska.remove("Kowalski"); System.out.println(nazwiska);
19     System.out.println(nazwiska.containsKey("Kowalski"));
20     System.out.println(nazwiska.containsValue(2));
21   }
22 }
```

Klasy parametryzowane typami

W językach obiektowych można definiować klasy uogólnione zawierające pola, których typy są parametrami, tzw. szablony klas (np. klasy z JCF czy STL).

Parametry typów podaje się w nawiasach (<>) po nazwie klasy, odzielając je od siebie przecinkami.

Typy uogólnione w Javie

W Javie typem uogólnionym może być tylko klasa (stąd typ `Integer` zamiast `int`). Jednak dzięki automatycznemu konwertowaniu nie jest to uciążliwe.

Pola typów uogólnionych przechowują tylko referencje do tych typów a nie kopie obiektów (w Javie nie ma niejawnego kopiowania).

Typem nie jest klasa parametryzowana ale dopiero jej ukonkretnienie z podanymi konkretnymi parametrami.

Kompilator zakłada, że typ podany jako parametr jest dowolną podklasą klasy `Object` - czyli dla obiektów klasy podanej jako parametr możemy używać tylko metod klasy `Object` (eventualnie możemy je nadpisać w klasie parametryzowanej).

Przykład - klasa implementująca stos - Stos.java

```
1  class ElemStosu<T> {
2    final T elem;
3    final ElemStosu<T> nast;
4    ElemStosu(T elem, ElemStosu<T> nast) {
5      this.elem = elem; this.nast = nast;
6    }
7  }
8  class PustyStos extends Exception{}
9  public class Stos<T> {
10   private ElemStosu<T> wierzch;
11   public Stos() { wierzch = null; }
12   public boolean empty() { return wierzch==null; }
13   public void push(T elem) {
14     wierzch = new ElemStosu<T>(elem,wierzch);
15   }
16   public T pop() throws PustyStos {
17     if( empty() ) throw new PustyStos();
18     T wynik = wierzch.elem;
19     wierzch = wierzch.nast;
20     return wynik;
21   }
22 }
```

Przykład - cd - StosTest.java

```
1  public class StosTest {
2    public static void main(String[] args) {
3      Stos<Integer> a = new Stos<Integer>();
4      Stos<String> b = new Stos<String>();
5      a.push(2); a.push(3);
6      try {
7        System.out.println(a.pop()+" "+a.pop());
8        System.out.println(a.pop()+" "+a.pop());
9      }
10     catch(PustyStos e) {
11       System.out.println("PustyStos!");
12     }
13     b.push("Marek"); b.push("Ala");
14     try {
15       while( !b.empty() )
16         System.out.println(b.pop());
17     }
18     catch(PustyStos e) {
19       System.out.println("PustyStos!");
20     }
21   }
22 }
```

Ten sam przykład w C++ – stos.cpp

```
1  template<typename T> class ElemStosu {
2    public:
3      T elem;
4      ElemStosu<T>* nast;
5      ElemStosu(T elem, ElemStosu<T>* nast) {
6        this->elem = elem;
7        this->nast = nast;
8      }
9  };
10 template<typename T> class Stos {
11   private:
12     ElemStosu<T>* wierzch;
13   public:
14     Stos() { wierzch = NULL; }
15     bool empty() { return wierzch==NULL; }
16     void push(T elem) { wierzch = new ElemStosu<T>(elem,wierzch);
17     T pop() {
18       if( empty() ) throw (string)"PustyStos!";
19       T wynik = wierzch->elem;
20       wierzch = wierzch->nast;
21       return wynik;
22     }
23 };
```

Ten sam przykład w C++ – stos.cpp (cd)

```
24  int main(int argc, char* argv[]) {
25    Stos<int> a;
26    Stos<string> b;
27    a.push(2); a.push(3);
28    try {
29      cout << a.pop() << " " << a.pop() << endl;
30      cout << a.pop() << " " << a.pop() << endl;
31    }
32    catch(string e) {
33      cout << e << endl;
34    }
35    b.push("Marek"); b.push("Ala");
36    try {
37      while( !b.empty() )
38        cout << b.pop() << endl;
39    }
40    catch(string e) {
41      cout << e << endl;
42    }
43  }
```

Dodanie specjalnych własności typu podanego jako parametr

Jeśli potrzebujemy aby nasza klasa podawana jako parametr posiadała dodatkowe metody możemy to w Javie uzyskać dodając jaką klasę (interfejs) nasza klasa ma dziedziczyć.

Na przykład jeśli klasa użyta jako parametr powinna mieć porządek liniowy to powinna dziedziczyć interfejs `Comparable` z metodą `compareTo`. Deklaracja takiego interfejsu (jest już w języku Java) wygląda nastąpująco:

```
1 public interface Comparable<T>{
2   public int compareTo(T o);
3 }
```

Przykład - drzewo binarne - `Drzewo.java` (1/2)

```
1  class ElemDrzewa<T extends Comparable<T>> {
2    final T elem;
3    ElemDrzewa<T> lewy;
4    ElemDrzewa<T> prawy;
5    ElemDrzewa(T elem)
6    {
7      this.elem = elem;
8      lewy = null;
9      prawy = null;
10   }
11   public String toString() { return elem.toString(); }
12 }
13 public class Drzewo<T extends Comparable<T>> {
14   private ElemDrzewa<T> korzen;
15   public Drzewo() { korzen = null; }
16   public void insert(T elem) { korzen = ins(elem, korzen); }
17   public boolean isElement(T elem) {
18     return isElem(elem,korzen); }
19   public String toString() { return toS(korzen); }
```

Przykład - drzewo binarne - Drzewo.java (2/2)

```
21    private ElemDrzewa<T> ins(T elem, ElemDrzewa<T> w) {
22      if( w==null ) return new ElemDrzewa<T>(elem);
23      if( elem.compareTo(w.elem)<0 )
24        w.lewy = ins(elem, w.lewy);
25      else if( elem.compareTo(w.elem)>0)
26        w.prawy = ins(elem, w.prawy);
27      return w;
28    }
29    private boolean isElem(T elem, ElemDrzewa<T> w) {
30      if( w==null ) return false;
31      if( elem.compareTo(w.elem)==0 ) return true;
32      if( elem.compareTo(w.elem)<0)
33        return isElem(elem, w.lewy);
34      else
35        return isElem(elem, w.prawy);
36    }
37    private String toS(ElemDrzewa<T> w) {
38      if( w!=null )
39        return "("+w.elem+":"+toS(w.lewy)+":"+toS(w.prawy)+")";
40      return "()";
41    }
42  }
```

Przykład - drzewo binarne - DrzewoTest.java

```
1  public class DrzewoTest
2  {
3    public static void main(String[] args)
4    {
5      Drzewo<String> d = new Drzewo<String>();
6
7      d.insert("Marek"); d.insert("Ala"); d.insert("Kot");
8      System.out.println(d.isElement("Ala"));
9      System.out.println(d.isElement("ma"));
10     System.out.println(d);
11   }
12 }
```

Podsumowanie

Realizacja mechanizmu typów uogólnionych w Javie wystarcza do większości zastosować – pewne niedogodności wynikają z braku tego mechanizmu w pierwszych wersjach języka.

Rozwiązania przyjęte w C++ są bardziej elastyczne (umożliwiają posługiwanie się niezdefiniowanymi metodami przy pisaniu schematów czego poprawność jest sprawdzana dopiero podczas kompilacji konkretnej instancji szablonu). W Javie trzeba jawnie podać wymagania danej klasy będącej parametrem.