



Programozási technológia I.

Generikus osztályok, gyűjtemények és algoritmusok

Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

Dr. Szendrei Rudolf
Informatikai Kar
Eötvös Loránd Tudományegyetem



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

Tartalom

1 Generikus osztályok

2 Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények megvalósítása

A Java gyűjteményei

A Java beépített algoritmusai



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

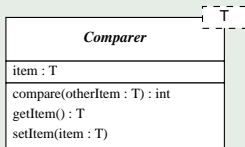
A Java gyűjteményei

A Java beépített
algoritmusai

Generikus osztályok

Generikus osztályok Java-ban és UML-ben

- Az UML-beli paraméteres osztályok a Java nyelvben generikus (sablon) osztályok segítségével valósíthatók meg
- Az UML-ben megjelölt paraméterek Javában a generikus paraméterek
- Java-ban a generikus paraméterek osztálynevek lehetnek, melyek segítségével a generikus osztály definíciójában paraméterezhető típusok adhatóak meg



```
public class Comparer<T>{  
    private T    item;  
    public T    getItem() { ... }  
    public void setItem(T item) { ... }  
    public int  compare(T otherItem) { ... }  
}
```



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

Generikus osztályok

Generikus osztályok Java-ban és UML-ben

- A generikus osztályok használatakor meg kell adni a generikus paraméterek konkrét értékeit (a konkrét osztályneveket). Ekkor a generikus osztályban a generikus paraméterekkel jelölt típusok helyére a kapott konkrét típusok helyettesítődnek be. Ettől fogva már példányosíthatjuk a konkrét osztályt.

```
public class Comparer<T>{  
    private T    item;  
    public T    getItem() {...}  
    public void setItem(T item) {...}  
    public int  compare(T otherItem) {...}  
}  
  
...  
  
Comparer<String> comparer = new Comparer<>();  
comparer.setItem("szöveg");  
int compared = comparer.compare("valami");
```



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

Generikus osztályok

Generikus osztályok Java-ban és UML-ben

- A generikus osztályok használata is egyfajta absztrakció, de az absztrakt osztályokkal ellentétben itt nem az elvégzendő műveletek megvalósítása az ismeretlen, hanem az adatok típusa (legalább részben), amelyeken a műveleteket végezzük.

```
public class Comparer<T>{  
    private T    item;  
    ...  
}  
  
public abstract class GeometricShape{  
    public abstract double getArea();  
}
```

- A két absztrakciót akár együtt is alkalmazhatjuk

```
public abstract class ItemProcessor<T>{  
    public abstract T getProcessedItem();  
    ...  
}
```



Gyűjtemények

Bevezetés

- A gyűjtemény egy absztrakt adatszerkezet
 - tetszőleges mennyiségű adat csoportosítását végzi
 - az adatok az adott probléma megoldása szempontjából egyformán fontosak
 - az adatokon szabályozott módon lehet műveleteket végezni
- A tárolt adatok általában egyforma típusúak, vagy legalábbis ugyanabból a típusból vannak származtatva
- A tömböket nem tekintjük gyűjteményeknek, mert rögzített mérettel rendelkeznek. Igaz, a gyűjtemények megvalósításához gyakran használunk tömböket.



Gyűjtemények

Példa

```
public class SampleCollection<E> implements Collection<E> {

    @Override // tárolt elemek "száma" (lehetne összméret is...)
    public int size() {...}

    @Override // üres-e?
    public boolean isEmpty() {...}

    @Override // tartalmazza az adott objektumot?
    public boolean contains(Object o) {...}

    @Override // a bejárás iterátora
    public Iterator<E> iterator() {...}

    @Override // hozzáadja a megadott elemet
    public boolean add(E e) {...}

    @Override // eltávolítja a megadott elemet
    public boolean remove(Object o) {...}

    ...
}
```



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

Gyűjtemények

Példa - folyt.

```
public class SampleCollection<E> implements Collection<E> {  
  
    ...  
  
    @Override // benne van-e a megadott gyűjtemény minden eleme?  
    public boolean containsAll(Collection<?> c) {...}  
  
    @Override // hozzáadja a megadott gyűjtemény elemeit  
    public boolean addAll(Collection<? extends E> c) {...}  
  
    @Override // eltávolítja a megadott gyűjtemény elemeit  
    public boolean removeAll(Collection<?> c) {...}  
  
    @Override // meghagyja a megadott gyűjtemény elemeit  
    public boolean retainAll(Collection<?> c) {...}  
  
    @Override // eltávolítja az összes elemet  
    public void clear() {...}  
  
    @Override // tömbbé konvertálja  
    public Object[] toArray() {...}  
  
    @Override // tömbbé konvertálja  
    public <T> T[] toArray(T[] a) {...}  
}
```




Gyűjtemények

Gyűjtemények bejárása

- Egy gyűjtemény bejárásakor a gyűjtemény minden elemét sorra vesszük, és minden elemmel elvégezzük egy adott műveletet
- Általában a gyűjtemények nem indexelhetőek, ezért egy úgynevezett *iterátor* segítségével járhatóak be

```
Collection<Double> doubles = Arrays.asList(2.72, 3.14, 42.0);

double sum = 0.0;
for (Iterator<Double> it = doubles.iterator(); it.hasNext(); )
{
    Double d = it.next();
    sum += d;
}

System.out.println(sum);
```

- Megjegyzés: a `Double` a `double` adattípus „beburkoló” osztálya, amelyre most azért van szükségünk, mert generikus paraméter csak osztály lehet.



Gyűjtemények

Gyűjtemények bejárása

```
Collection<Double> doubles = Arrays.asList(2.72, 3.14, 42.0);

double sum = 0.0;
for (Iterator<Double> it = doubles.iterator(); it.hasNext();)
{
    Double d = it.next();
    sum += d;
}

System.out.println(sum);
```

- A gyűjtemények egyszerűbben is bejárhatók, ha a *foreach* ciklust használjuk

```
Collection<Double> doubles = Arrays.asList(2.72, 3.14, 42.0);

double sum = 0.0;
for (Double d : doubles){
    sum += d;
}

System.out.println(sum);
```



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

Gyűjtemények

Gyűjtemények megvalósítása

- A `Collection<E>` interfészt, vagy akár valamelyik speciálisabb interfészét kell megvalósítani
- Érdeemes a megvalósított gyűjteménynek is generikus osztálynak lennie, hogy tetszőleges típusú adat tárolására alkalmas legyen
- A gyűjtemény műveleteinek absztrakt formái a megvalósítandó interfészben már adottak, így csak a tárolt adatok reprezentációjával és a műveletek függvénytorzseinek meghatározásával kell törődnünk
- Az `AbstractCollection<E>` osztály már tartalmazza a szokásos gyűjteményi viselkedést, így érdemes abból származtatni a gyűjteményünket, és csak a lényegre koncentrálni



Generikus osztályok

Gyűjtemények

Bevezetés

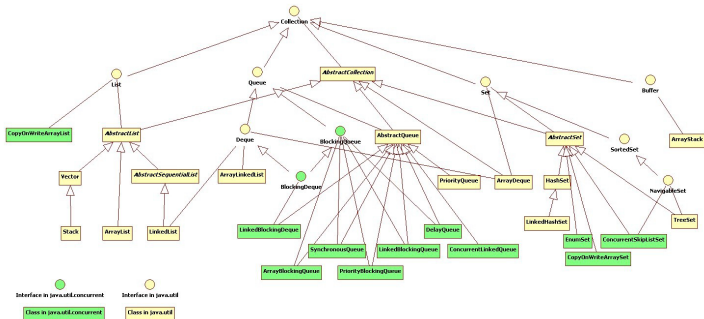
Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai



Dr. Szendrei Rudolf



Generikus osztályok

Gyűjtemények

Bevezetés

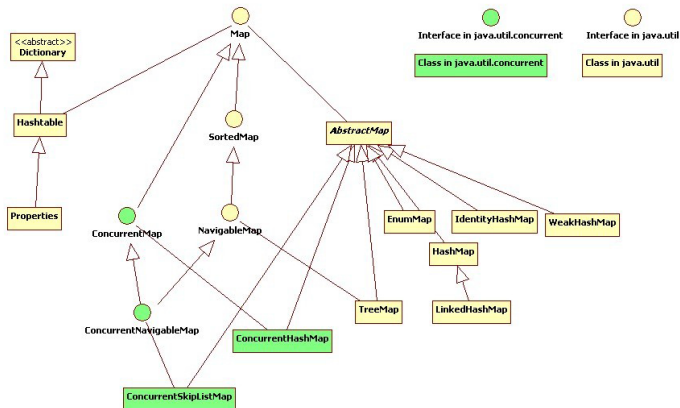
Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai





Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

A java gyűjteményei

Gyűjtemények, mint adatszerkezetek

- A gyűjteményekben tárolt adatok elérésének, módosításának, törlésének ideje nagyban függ a választott adatszerkezettől (lásd Algoritmusok és adatszerkezetek tárgy)
- Ökölszabályok kezdőknek:
 - ❶ kritikus futási idő alapján választunk adatszerkezetet (pl. keresés és beszúrás);
 - ❷ adatok feldolgozási módja szerint választunk adatszerkezetet (pl. rendezés);
 - ❸ csekély elemszám esetén kényelmesen dolgozhassuk fel az adatokat;
 - ❹ speciális feladat esetén készítsünk saját adatszerkezetet (pl. térinformatikai feladatok esetén).



A java gyűjteményei

Gyűjtemények, mint adatszerkezetek

- A Java gyűjteményeit csoportosíthatjuk a felhasznált adatszerkezet típusa alapján:
 - közvetlen elérésű, indexelhető;
 - láncolt listás;
 - fa adatszerkezetű;
 - hasító függvényt alkalmazó.
- Bizonyos esetekben a fentiek nem eléggé illeszkednek egy adott feladathoz, ezért elképzelhető, hogy saját hibrid adatszerkezetet készítünk (pl. fa leveleit két irányú láncolt listába fűzzük).



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

A java gyűjteményei

Közvetlen elérésű, indexelhető gyűjtemények

- Főbb jellemzők:
 - konstans elem elérési idő,
 - lassú beszúrás,
 - könnyű rendezés
- Gyakran használt Java osztályok:
 - ArrayList,
 - ArrayLinkedList,
 - Vector,
 - Stack...



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

A java gyűjteményei

Láncolt listás adatszerkezetű gyűjtemények

- Főbb jellemzők:
 - Első/utolsó listaelem azonnal elérhető, a köztes
 - elemek elérési ideje viszont lassú lehet,
 - könnyen bővíthető
- Gyakran használt Java osztályok:
 - Queue,
 - DeQueue,
 - PriorityQueue,
 - LinkedList...



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

A java gyűjteményei

Fa adatszerkezetű gyűjtemények

- Főbb jellemzők:
 - Logaritmikus idejű elem elérés, módosítás és törlés,
 - könnyen bővíthető,
 - az elemek indexelése (n-edik elem kiválasztása)
„megvalósítható”,
 - jól alkalmazhatóak asszociatív tárolókhoz
- Gyakran használt Java osztályok:
 - TreeSet,
 - TreeMap...



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

A java gyűjteményei

Hasító függvényt alkalmazó gyűjtemények

- Főbb jellemzők:
 - Gyors elérési, módosítási és törlési lehetőség,
 - kis elemszám esetén vagy jó hasító függvény esetén nagyon gyors lehet,
 - az elemek nem indexelhetők
 - az elemek rendezése nem lehetséges
 - jól alkalmazhatóak asszociatív tárolókhoz
- Gyakran használt Java osztályok:
 - HashSet,
 - LinkedHashSet,
 - HashTable,
 - HashMap,
 - LinkedHashMap...



Generikus osztályok

Gyűjtemények

Bevezetés

Példa

Gyűjtemények bejárása

Gyűjtemények
megvalósítása

A Java gyűjteményei

A Java beépített
algoritmusai

A Java beépített algoritmusai

Algoritmusok gyűjteményeken - `java.util.Collections`

- <http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html>

Algoritmusok tömbökön - `java.util.Arrays`

- <http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html>