Αντικειμενοστρεφής Προγραμματισμός

Παναγιώτης Αδαμίδης
adamidis@it.teithe.gr

Γενικεύσεις (Generics)

Γενικεύσεις – Generics (1/2)

"Generics – An enhancement to the type system that supports operations on objects of various types while providing compile-time type."

The Java Tutorial, Oracle

- Πρώτη εμφάνιση: Java 5, 2004
- Δυνατότητα ορισμού κλάσεων, διασυνδέσεων και μεθόδων που έχουν σαν παραμέτρους τύπους δεδομένων οι οποίοι λειτουργούν με αντικείμενα διαφορετικών τύπων και παρέχουν σταθερότητα στον κώδικα προσφέροντας τη δυνατότητα αναγνώρισης περισσοτέρων λαθών κατά την **μεταγλώττιση**.

Μεταβλητές/Παράμετροι Τύπων

- Ειδικές μεταβλητές με τιμή κάποιο τύπο **αναφοράς**.
- Δεν αποτελούν μέρος του ονόματος της κλάσης στην οποία δηλώνονται.
- Τα αναγνωριστικά μεταβλητών τύπων πρέπει να περιέχουν μόνο ένα κεφαλαίο χαρακτήρα.

Αναγνωριστικό	Αναπαριστά...
T	Type ...κάποιος τύπος
E	Element .... Java Collections Framework
K	Key ...τύπος κλειδιών απεικονίσεων
V	Value ...τύπος τιμών απεικονίσεων
X	τύποι εξαιρέσεων
N	αριθμητικοί τύποι (πχ. Java.lang.number)
S, U, V ... ή T1, T2, T3 ...	2ος, 3ος, 4ος τύπος

Γενικές/Παραμετρικές Κλάσεις
Generic Classes

- Ορίζονται όπως και οι μη γενικές κλάσεις

```
class Onoma <T1[, T2[, ..., Tn]]> {
        σώμα κλάσης
}
```

- Αποθηκεύονται σε αρχεία όπως και οι μη γενικές κλάσεις
- Ο τύπος της παραμέτρου πρέπει να καθοριστεί πριν τη χρήση της κλάσης
- Η μεταγλώττιση με την οδηγία `-Xlint` παρέχει περισσότερες πληροφορίες για πιθανά προβλήματα του κώδικα

```
javac -Xlint Sample.java
```

Παράδειγμα (χωρίς Generics)

- Μια κλάση περιγραφής ενός κελιού πίνακα ...οιουδήποτε τύπου.
- Ποιος θα έπρεπε να είναι ο τύπος του, αφού θέλουμε να περιγράφει οιοδήποτε τύπο;

```
public class ArrayCell {
    private Object cellContent;
    public Object read() { return cellContent; }
    public void write (Object obj) {
        cellContent = obj;
    }
}
```

Παράδειγμα (με Generics)

- Τι αλλάζει ....;

```
public class ArrayCell<AnyType> {
    private AnyType cellContent;
    public AnyType read() {
        return cellContent;
    }
    public void write (AnyType obj) {
        cellContent = obj;
    }
}
```

Διαφορά χρήσης

- Χωρίς generics, απαιτείται μετατροπή τύπου (cast):

```
ArrayCell cell = new ArrayCell ();
cell.write("hello");
String s = (String) cell.read();
```

- Με generics, δεν χρειάζεται μετατροπή τύπου :

```
ArrayCell<String> cell = new ArrayCell<String> ();
cell.write("hello");
String s = cell.read();
```

Απλή κλάση Box

- Η κλάση περιέχει αντικείμενα …οιουδήποτε τύπου.
- Ποιος θα έπρεπε να είναι ο τύπος του;

```
public class Box {
    private Object obj;
    public Object get() {
        return obj;
    }
    public void set (Object o) {
        obj = o;
    }
}
```

Κλάση Box με Generics

```
public class Box <T> {
    private T obj;
    public T get() {
        return obj;
    }
    public void set (T  o) {
        obj = o;
    }
}
```

The Diamond <>

- Από τη Java 7 αλλά και … μετά
- Κατά την κλήση του δομητή το όρισμα του τύπου μιας γενικής κλάσης μπορεί να είναι κενό (<>) εάν ο μεταγλωττιστής μπορεί να αποφασίσει τον τύπο από το περιεχόμενο. Πχ.

```
ArrayCell<String> cell = new ArrayCell<> ();
Box<Integer> integerBox = new Box<>();
Box<String> textBox = new Box<>();
```

Κληρονομικότητα

- Μια γενική κλάση μπορεί να επεκτείνει μια γενική κλάση
- Μια γενική κλάση μπορεί να επεκτείνει μια μη γενική κλάση
- Μια μη γενική κλάση μπορεί να επεκτείνει μια γενική κλάση
- Μια γενική κλάση **δεν** μπορεί να επεκτείνει (είτε άμεσα, είτε έμμεσα) την κλάση `Throwable`

Διασυνδέσεις / Interfaces

- Μια γενική διασύνδεση μπορεί να επεκτείνει μια γενική διασύνδεση
- Μια γενική διασύνδεση μπορεί να επεκτείνει μια μη γενική διασύνδεση κλάση
- Μια μη γενική διασύνδεση μπορεί να επεκτείνει μια γενική διασύνδεση
- Μια γενική κλάση μπορεί να υλοποιήσει μια γενική διασύνδεση
- Μια μη γενική κλάση μπορεί να υλοποιήσει μια γενική διασύνδεση

Παράδειγμα με 2 παραμέτρους (1/2)

```
public interface Pair<K, V> {
        public K getKey();
        public V getValue();
}
public class OrderedPair<K, V> implements Pair<K, V> {
        private K key;
        private V value;
        public OrderedPair(K key, V value) {
                this.key = key;
                this.value = value;
        }
        public K getKey()      { return key; }
        public V getValue() { return value; }
}
```

Παράδειγμα με 2 παραμέτρους (2/2)

- Pair<String, String> p2 =
new OrderedPair<String, String>("hello", "world");
Η τιμή και των δύο παραμέτρων είναι String.

- Pair<String, Integer> p1 =
new OrderedPair<String, Integer>("Even", 8);
Η τιμή του K είναι String και του V είναι Integer (autoboxing).

- Με σημειολογία διαμαντιού:

OrderedPair<String, Integer> p1 = new OrderedPair**<>**("Even", 8);

OrderedPair<String, String> p2 =
new OrderedPair**<>**("hello", "world");

Περιορισμοί παραμέτρων τύπων

- Δεν μπορεί να χρησιμοποιηθεί ως όνομα δομητή ή όπως ένας δομητής για δημιουργία αντικειμένων. Πχ. Τα παρακάτω **δεν** ισχύουν:

```
T object = new T();
T[] a = new T[10];
```

- Επίσης η δημιουργία πινάκων με παράμετρο τύπων όπως παρακάτω είναι λάθος:

```
Box<String>[] a = new Box<String>[10];
```

Φράγματα/Όρια (Bounds) (1/2)

- Μερικές φορές θέλουμε να περιορίσουμε τους πιθανούς τύπους που μπορούμε να δώσουμε σε μια παράμετρο τύπων. Πχ. Μπορεί να θέλουμε να χρησιμοποιήσουμε ως τύπο μόνο κλάσεις που υλοποιούν το interface Comparable:

```
public class BoundClass<T extends
                        Comparable>
```

  - Το "extends Comparable" περιορίζει τις τιμές του T στις κλάσεις που υλοποιούν την Comparable.
  - Σε αυτή τη περίπτωση, η χρήση κλάσεων που δεν υλοποιούν το Comparable προκαλεί λάθος κατά τη μεταγλώττιση.

Φράγματα/Όρια (Bounds) (2/2)

- Το όριο ενός τύπου μπορεί να είναι όνομα κλάσης οπότε μόνο κλάσεις που την επεκτείνουν μπορούν να μπουν στη θέση της παραμέτρου του τύπου, πχ.

```
public class ExClass<T extends Class1>
```

- Τα όρια μπορούν να περιλαμβάνουν πολλές διασυνδέσεις και μία μόνο κλάση.
- Για πολλές παραμέτρους τύπων η σύνταξη είναι:

```
public class Sample<T1 extends Class1,
            T2 extends Class2 &
            Comparable>
```

interface Comparable

- Πριν τη Java 5

```
package java.lang;
public inteface Comparable {
        public int compareTo (Object o);
}
```

- Από τη Java 5 …και μετά

```
package java.lang;
public inteface Comparable <T>{
        public int compareTo (T  o);
}
```

Wildcards (Μπαλαντερ)

- Περιέχει το χαρακτήρα «?» και χρησιμοποιείται για να αναπαραστήσει οιονδήποτε τύπο.
- Μπορούμε να δηλώσουμε φραγμένους μπαλαντέρ
- Άνω φράγμα (upper bound):
  - ? extends T
  - πχ. Box<? extends Number> box1;
- Κάτω φράγμα (lower bound):
  - ? super T
  - πχ. Box<? super Integer> box2;

Συμβατότητα εκχώρησης

- Έγκυρες:
- Box<Integer> bi = new Box<>(1);
- Box<String> bs = new Box<>("Δευτέρα")
- Box<Float> bf = new Box<>(3.15.f);
- Box<?> b = bi;
- b = bs;
- b = bf;
- Box<? extends Number> bu = bi;
- bu = bf;
- Box<? super Integer> bl = bi;

Γενικές Μέθοδοι

- Σύνταξη:

[πρόσβαση] <παράμετροι τύπων> <τύπος_επιστροφής> <αναγνωριστικό_μεθόδου>(<παράμετροι>) [εξαιρέσεις]

  - Μπορούν να δηλωθούν τόσο μέσα σε γενικές όσο και σε μη γενικές κλάσεις
  - Η εμβέλεια των μεταβλητών τύπων που δηλώνει μια μέθοδος περιλαμβάνει όλη τη δήλωση της μεθόδου.
- Κλήση:
- Αναφορά_ή_τύπος.<ορίσματα_τύπων>αναγνωριστικό <ορίσματα_μεθόδου>

Παράδειγμα γενικής μεθόδου

```
public static <T extends Comparable<T>> T max
                        (T[] pinakas){
    T max = null;
    for (T elem : pinakas)
       if (max==null ||
              max.compareTo(elem)<0)
          max = elem;
    return max;
}
```

X.<Integer>max (new Integer[] {4,2,5,6});
X.<String>max(new String[] {"Νίκος", "Πέτρος", "Γιώργος"})