Παραμετρικοί Τύποι (Generics)

Γιώργος Θάνος
gthanos@uth.gr
Γραφείο: Γ5/8, 3ος όροφος
Γκλαβάνη 37

Όταν θέλουμε να δημιουργήσουμε κλάσεις που μπορούν να αποθηκεύσουν αντικείμενα οποιασδήποτε κλάσης (π.χ. εάν θέλουμε να δημιουργήσουμε μία διασυνδεδεμένη λίστα ή μία στοίβα) τότε θα πρέπει τις κλάσεις αυτές να τις κάνουμε τόσο γενικές ώστε να μπορούν να αποθηκεύουν οποιονδήποτε τύπο αντικειμένων.

Η υιοθέτηση γενικών κλάσεων έχει το πλεονέκτημα ότι μπορεί να αποθηκεύσει αντικείμενα οποιασδήποτε κλάσης, όμως απαιτεί αρκετές μετατροπές τύπων (typecasts). Οι μετατροπές τύπων όταν γίνονται από τον προγραμματιστή ενέχουν κινδύνους ως προς την ορθή μετατροπή.

Παράδειγμα χρήσης γενικής κλάσης για την αποθήκευση δεδομένων

```
public class Box {
  private Object object;
  public void set(Object object) {
    this.object = object;
  }
  public Object get() {
    return object;
  }
}
```

```
public class BoxUsage {
   public static void main(
                    String args[]) {
     Box b = new Box();
     Integer n = new Integer(5);
     b.set(n);
     Integer s = (Integer)b.get();
   }
}
```

Παράδειγμα λανθασμένης χρήσης γενικής κλάσης για την αποθήκευση δεδομένων

Μπορούμε να περάσουμε ως όρισμα στη μέθοδο *set* οποιοδήποτε τύπο δεδομένων ή

Μπορούμε να αναθέσουμε την επιστρεφόμενη τιμή της μεθόδου *get* σε οποιοδήποτε τύπο δεδομένων.

```
public class BoxUsage {
   public static void main(
                        String args[])
 {
      Box b = new Box();
      Integer n = new Integer(5);
      b.set(n);
      String s = (String)b.get();
   }
}
```

Παραμετρικοί Τύποι

Προκειμένου να αποφύγουμε τα προβλήματα της προηγούμενης διαφάνειας επιλέγουμε να χρησιμοποιήσουμε παραμετρικούς τύπους δεδομένων (Generics)

Οι παραμετρικοί τύποι προφυλάσσουν τον προγραμματιστή από την λανθασμένη χρήση τύπων δεδομένων κατά τη μεταγλώττιση του προγράμματος.

Παράδειγμα χρήσης παραμετρικής κλάσης

```
public class Box<T> {
  // T stands for "Type"
  private T t;

  public void set(T t) {
    this.t = t;
  }
  public T get() { return t;
 }
}
```

Η δήλωση της κλάσης **Box<T>** σημαίνει ότι κατά τον ορισμό αντικειμένων της κλάσης αυτά θα πρέπει να προσδιορίζεται ανάμεσα στους χαρακτήρες '<', ΄>' ένας επιπλέον αναφορικός τύπος δεδομένων

```
Box<String> b1 = new Box<String>();
Box<Integer> b2  = new Box<Integer>();
Box<Student> b3 = new Box<Student>();
```

Δημιουργία και χρήση αντικειμένων

```
Box<Integer> integerBox = new Box<Integer>();
```

ή

```
Box<Integer> integerBox = new Box<>();
```

Προσοχή: το παραπάνω δεν είναι ίδιο με το παρακάτω.

```
Box<Integer> integerBox = new Box();
```

Εδώ ο *compiler* θα εκδώσει το παρακάτω warning, καθώς επιχειρούμε να αναθέσουμε σε μία μεταβλητή παραμετρικού τύπου ένα αντικείμενο που δεν είναι αυτού του τύπου.

```
Note: BoxUsage.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Interfaces ως παραμετρικοί τύποι δεδομένων

Μη παραμετρικός τύπος

```
public interface Stack {
  public int size();
  public void push(Object o);
  public Object pop();
  public Object top();
}
```

Παραμετρικός τύπος

```
interface Stack <T> {
  public int size();
  public void push(T obj);
  public T pop();
  public T top();
}
```

Παραμετρικοί τύποι δεδομένων με πολλές παραμέτρους

```
public interface Pair<K, V> {
  public K getKey();
  public V getValue();
  public void setKey(K key);
  public void setValue(V
val);
}
```

```
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 void setKey(K key) { this.key = key; }
    public void setValue(V value) { this.value = value;
 }
    public K getKey()   { return key; }
    public V getValue() { return value; }
}
```

```
public class OrderedPairUsage {
  public static void main(String args[]) {
    Pair<String, Integer> p1 = new OrderedPair<String, Integer>("Even", 8);
    Pair<String, String>  p2 = new OrderedPair<String, String>("hello", "world");
    OrderedPair<String, Box<Integer>> p = new OrderedPair<>("primes", new Box<Integer>());
    // the following is not allowed
    Pair<String, Integer> p1 = new OrderedPair<>("hello", "world");
  }
}
```

Απλοί παραμετρικοί τύποι δεδομένων (Raw Generic Types)

Για την κλάση Box που ορίσαμε προηγούμενα μπορούμε να ορίσουμε και μη παραμετρικούς τύπους δεδομένων, όπως παρακάτω

```
class BoxUsage {
   public static void main(
                    String args[]) {
     Box b = new Box();
     b.set(5);
   }
}
```

Η χρήση απλών παραμετρικών τύπων δεδομένων παράγει warnings κατά τη μεταγλώττιση.

```
javac -Xlint BoxUsage.java
BoxUsage.java:4: warning: [rawtypes] found raw
 type: Box
      Box b = new Box();
      ^
  missing type arguments for generic class
Box<T>
  where T is a type-variable:
    T extends Object declared in class Box
```

ΔΕΝ συνιστάται σε καμία περίπτωση η χρήση παραμετρικών τύπων (generic types) ως απλών.

Παραμετρικές Μέθοδοι


```
class Util {
  public static <K, V> boolean isEqual(Pair<K, V> p1, Pair<K, V> p2) {
    return p1.getKey().equals(p2.getKey()) &&
      p1.getValue().equals(p2.getValue());
  }
}
public class OrderedPairUsage {
  public static void main(String args[]) {
    Pair<Integer, String> p1 = new OrderedPair<>(1, "Black");
    Pair<Integer, String> p2 = new OrderedPair<>(1, "Red");
    boolean same = Util.<Integer, String>isEqual(p1, p2);
    if( same ) { System.out.println("p1 is equal to p2"); }
    else { System.out.println("p1 is NOT equal to p2"); }
  }
}
```

Μέθοδοι οι οποίες λαμβάνουν παραμετρικούς τύπους δεδομένων.

Οι κλάσεις στις οποίες ανήκουν δεν είναι παραμετρικές.