Java Util labor

Melléklet: *PQueue sablondefiníciójának elemzése*

A *Util* labor 5. feladatában a *PQueue* osztály sablonparaméter <*T extends Comparable*> típusú volt.

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

Felmerül a kérdés, hogy ha a *Comparable* maga is sablon osztály, akkor miért nem kap sablonparamétert? Gond-e ez? Szintaktikailag helyes-e? Ha szeretnénk paraméteressé tenni, mi a helyes kifejezés?

A válaszhoz kicsit bele kell ásnunk a Java sablonkezelésébe és a sablonkezelés kialakulásába. A Java eredetileg nem támogatta a sablonokat. Ha példaként megnézzük a kollekció keretrendszer interfészeit és osztályait, akkor azt látjuk, hogy a Java 4. verziójában a *List* interfész néhány metódusa az alábbi módon nézett ki:

```
void add(Object e)
Object get(int index)
```

A sablonosítás után a *List<E>* interfészben ugyanezek már így néznek ki:

```
void add(E e)
E get(int index)
```

A kérdés az volt, hogy hogyan lehet egyszerre megőrizni a kompatibilitást és bevezetni a sablonokat. A megoldás az lett, hogy ha nem írunk sablon paramétert (ú.n. *raw type*), akkor azt a fordító *Object*-nek fogja tekinteni, és akkor az új, sablonos változatból automatikusan előáll a korábbi, *Object-alapú* változat. Ez az oka annak, hogy az 5. feladat *PQueue* osztályában a sablonparaméter típusa lehet *T extends Comparable* anélkül, hogy a *Comparable*-nek sablon-paramétert definiálnánk. [1]

Mit jelent ez a gyakorlatban? Milyen *T*-ket kaphat paraméterül a *PQueue*? A válasz, hogy bármit, ami megvalósítja a raw *Comparable*-t, vagyis van

```
int compareTo(Object o)
```

metódusa. Elvileg ez a metódus nem csak egy másik *T*-vel, hanem bármely másik osztály példányával össze tudja hasonlítani *this*-t. Ez azonban futási idejű hibához (pl. *ClassCastException*) vezethet, ha a paraméterként kapott objektum osztálya nekünk nem jó.

[1] Valójában a *List*, *ArrayList*, stb bytecode-jában már nem is találunk *T*-t, mindenhol az *Object*-es változat szerepel. A fordító kitörli a sablon-elemeket (ez az ú.n. *type erasure*), és *Object*-tel vagy más típussal helyettesíti őket (pl. ha a sablon paramétere *<T extends Cloneable>*, akkor a *T*-k helyére a fordító *Cloneable*-t helyettesít a sablon bytecode-jában). Ahol használjuk a sablont, oda pedig a fordító szükség szerint *cast*-okat tesz, illetve fordítási idejű típusellenőrzést végez.

Mi a teendő, ha azt szeretnénk, hogy *PQueue*-ba csak olyan elemek kerülhessenek, amik csak saját osztályukba tartozó elemekkel hasonlíthatók össze? Definiálni kell a *Comparable* paraméterét:

```
public class PQueue2 <T extends Comparable<T>>
```

Ekkor a megvalósítandó metódus paramétere *T* lesz, így fordítási idejű ellenőrzés segít abban, hogy csak a megfelelő típusú objektummal végezzünk összehasonlítást.

Van-e ennek a megoldásnak hátránya? Sajnos van. Képzeljük el az alábbi osztályokat:

```
class C1 implements Comparable<C1> {
      protected int q;
      @Override
      public int compareTo(C1 arg0) {
            return q-arg0.q;
      }
}
class C2 extends C1 {}
```

A *C1* osztály egy *natural ordering*-gel rendelkező osztály, másik *C1* objektummal bármikor összevethető. Akármi mással pedig nem, mert a *compareTo* metódus paramétere (*C1 arg0*) ezt nem engedi meg, fordítási idejű hiba lép fel. A *C2* pedig *C1* leszármazottja, és ezért megörökli a *natural ordering*-et, másik *C2* objektummal összevethető, de igazából bármelyik *C1* objektummal vagy *C1* leszármazottjával is, ami az öröklés miatt rendben is van.

Melyik osztállyal lehet a *PQueue2*-t paraméterezni?

```
PQueue2<C1> p21 = new PQueue2<C1>(); // OK
PQueue2<C2> p22 = new PQueue2<C2>(); // fordítási hiba
```

A *p22*-es objektum azért nem jöhet létre, mert *C2 implements Comparable<C1>*, és nem *implements Comparable<C2>.* Márpedig a *PQueue2* sablonparaméter az utóbbit írná elő. A *Comparable<C1>* és a *Comparable<C2>* sablon-kifejezések azonban nem kompatibilisek egymással.

Hogyan tudjuk elérni, hogy a *p22* is jó legyen? Ehhez kell a *wildcard* kérdőjel és a *super* kulcsszó. Olyan osztállyal kell paraméterezhető legyen a *PQueue*, ami megvalósítja a *Comparable*-t, de ezt akár úgy is, hogy ősétől örökli, vagyis a *Comparable* egy őssel van paraméterezve:

```
public class PQueue3 <T extends Comparable<? super T>>
```

Ez a megoldás már illeszkedik mind *C1*-re, mind *C2*-re, és az elvárt módon is működik.

Végül, visszatérve magára a feladatra, felmerülhet, hogy miért nem a *PQueue3* definíciót alkalmaztuk egyből? Ennek az az oka, hogy úgy éreztük, a feladat megértését és megoldását indokolatlanul akadályozta volna a "? *super T*"-s kifejezés és a dupla kacsacsőr (>>). Ez az elemzés viszont lehetőséget adott arra, hogy rávilágítsunk a Java nyelv sablonkezelésének néhány sötétben hagyott pontjára.