Programozás III

GENERIKUSOK és EGYEBEK

JAVA – GENERIKUSOK

A **generikus** lehetőséget ad osztályok más típussal való paraméterezésére.
Pl.:

java.util

Interface List<E>

Type Parameters:

E - the type of elements in this list

A generikusan (általánosan) definiált List aktuális típus-paramétere az lehet, hogy a lista milyen konkrét típusú adatokat tartalmazzon – pl. List<Paros>

```
private List<Paros> adatok = new ArrayList<>();
```

Mi ez?

JAVA – GENERIKUS PÉLDA

De sok más helyen is használhatjuk a generikust.

Pl. készíthetünk olyan saját generikus (általános) számológép típust, amely ugyanúgy teszi a dolgát: összead, szoroz, de egyszer egészekkel, máskor törtekkel, attól függően, hogy Integer vagy a Double típussal paraméterezve konkretizáltuk-e.

PÉLDA SAJÁT GENERIKUS OSZTÁLYRA

```
public class GenerikusOsszeg <T extends Number> {

   T egyik;
   T masik;

   GenerikusOsszeg(T egyik, T masik){
       this.egyik = egyik;
       this.masik = masik;
   }

   public double osszeg(){
       return (egyik.doubleValue() + masik.doubleValue());
   }

}
```

MÁSIK PÉLDA SAJÁT GENERIKUS OSZTÁLYRA

```
public static void main(String[] args) {
    Integer a = new Integer(1);
    Integer b = new Integer(2);
    Integer c = nagyobb(a,b);
    Double ad = new Double(2.5);
    Double bd = new Double(1.25);
    Double cd = nagyobb(ad,bd);
    System.out.println("c = " + c + " cd = " + cd);
}

private static <E extends Comparable <E>> E nagyobb(E a, E b) {
    if(a != null && b != null){
        return (a.compareTo(b) > 0)? a : b;
    }else{
        return null;
    }
}
```

```
run:
c = 2 cd = 2.5
```

PÉLDA SAJÁT GENERIKUS OSZTÁLYRA

```
public class GenerikusProba{

   public static void main(String args[]){
       int a=2;
       int b=3;
       GenerikusOsszeg<Integer> egy =
                       new GenerikusOsszeg<Integer>(a,b);
       System.out.println("Az összeg: " + egy.osszeg());

       float x = (float)2.4;
       float y = (float)3.2;

       GenerikusOsszeg<Float> ketto =
                       new GenerikusOsszeg<Float>(x,y);
       System.out.println("Az összeg: " + ketto.osszeg());
   }
}
```

HARMADIK PÉLDA GENERIKUS HASZNÁLATÁRA

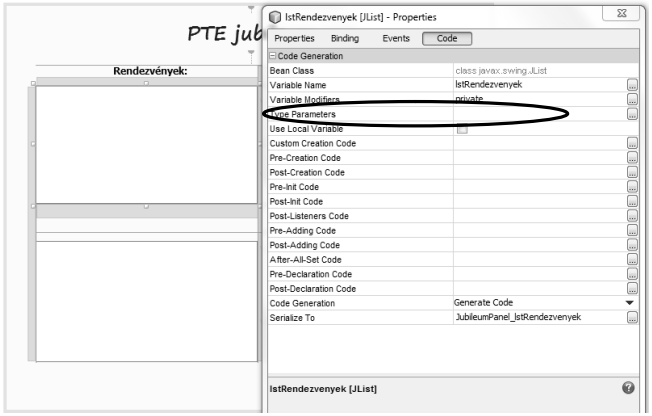
Emlékeztető: a 3. gyakorlaton vett jubileumi feladat.

Fordításkor:

```
Compiling 6 source files to C:\temp\gyak3\build\classes
C:\temp\gyak3\src\feluletek\JubileumPanel.java:42: warning: [unchecked] unchecked call to
        setModel(ListModel<E>) as a member of the raw type JList
        lstRendezvenyek.setModel(rendezvenyModel);
  where E is a type-variable:
    E extends Object declared in class JList
C:\temp\gyak3\src\feluletek\JubileumPanel.java:219: warning: [unchecked] unchecked conversion
        List<Rendezveny> valasztottak = lstRendezvenyek.getSelectedValuesList();
  required: List<Rendezveny>
  found:    List
2 warnings
```

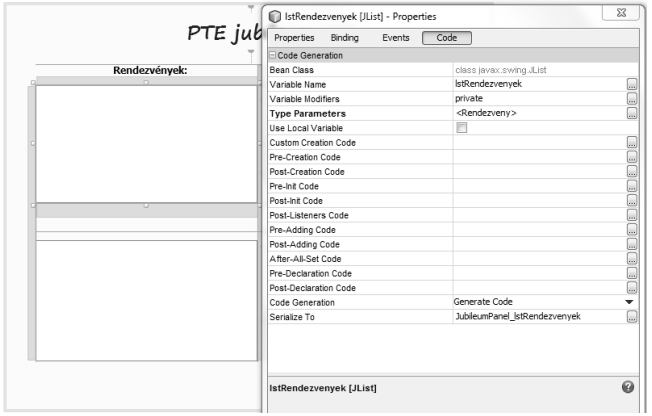
HARMADIK PÉLDA GENERIKUS HASZNÁLATÁRA

Oka:



HARMADIK PÉLDA GENERIKUS HASZNÁLATÁRA

Megoldás:



MÉG EGY FOGALOM

Enumerátor:
Az enum fix konstans értékek létrehozására használható.
Mivel típusos, így biztonságosabb mint egy int konstans.

PI: public static final int HETFO = 1

– nem tudjuk, hogy az 1-es mit takar, és kezelni kell az érvénytelen értéket

public enum Nap { HETFO, KEDD,…

– típusos, így nem kell foglalkozni az érvénytelen értékekkel

Lehet konstruktora, és használhatunk benne final és nem final mezőket.

Enumerátor példa 1.

```
public class EnumPelda {

    public enum Nap{
        HETFO, KEDD, SZERDA, CSUTORTOK,
        PENTEK, SZOMBAT, VASARNAP
    }

    private Nap nap;

    public EnumPelda(Nap nap) {
        this.nap = nap;
    }

    public void milyenNap(){
        switch(nap){
            case KEDD: {
                System.out.println(nap + " a legjobb nap, mert Java előadás.\n");
                break;
            }
            case SZERDA:
            case CSUTORTOK :{
                System.out.println(nap + " is jó, mert Java gyakorlat.\n");
                break;
            }default: System.out.println(nap + ". Mit ér a nap Java nélkül?\n ");
        }
    }
```

MÉG EGY FOGALOM

```
public static void main(String[] args){
    EnumPelda kedd = new EnumPelda(Nap.KEDD);
    kedd.milyenNap();
    EnumPelda csutortok = new EnumPelda(Nap.CSUTORTOK);
    csutortok.milyenNap();
    EnumPelda pentek = new EnumPelda(Nap.PENTEK);
    pentek.milyenNap();
}
```

```
run:
KEDD a legjobb nap, mert Java előadás.

CSUTORTOK is jó, mert Java gyakorlat.

PENTEK. Mit ér a nap Java nélkül?
```

MÉG EGY FOGALOM

```
class indito{

    public static void main(String[] args){
        Nap[] napok = Nap.values();
        System.out.println("A hét napjai: ");
        for(Nap nap: napok){
            System.out.println(nap + " tulajdonsága: " +
                                nap.getTulajdonsag());
        }

        Nap nap = Nap.KEDD;
        nap.setTulajdonsag("akkor Belgium");
        System.out.println(nap + " tulajdonsága: "
                + nap.getTulajdonsag());
    }
}
```

Enumerátor példa 2.

```
public enum Nap {

    HETFO("Nehezen indul"), KEDD("Végre Java"),
    SZERDA("Hurrá, Java"), CSUTORTOK("Fárasztó"),
    PENTEK("Jön a hétvége :)"),
    SZOMBAT("Hét vége :))"), VASARNAP("Jaj, mindjárt hétfő");

    private String tulajdonsag;

    private Nap(String tulajdonsag) {
        this.tulajdonsag = tulajdonsag;
    }

    public String getTulajdonsag() {
        return tulajdonsag;
    }

    public void setTulajdonsag(String tulajdonsag) {
        this.tulajdonsag = tulajdonsag;
    }
}
```

MÉG EGY FOGALOM

```
run:
A hét napjai:
HETFO tulajdonsága: Nehezen indul
KEDD tulajdonsága: Végre Java
SZERDA tulajdonsága: Hurrá, Java
CSUTORTOK tulajdonsága: Fárasztó
PENTEK tulajdonsága: Jön a hétvége :)
SZOMBAT tulajdonsága: Hét vége :))
VASARNAP tulajdonsága: Jaj, mindjárt hétfő
KEDD tulajdonsága: akkor Belgium
```

ÖSSZETETTEBB ENUM PÉLDA

```
public enum Nap {

    HETFO(new Tulajdonsag("Nehezen indul", -1)),
    KEDD(new Tulajdonsag("Végre Java", 5)),
    SZERDA(new Tulajdonsag("Hurrá, Java", 4)),
    CSUTORTOK(new Tulajdonsag("Fárasztó", 2)),
    PENTEK(new Tulajdonsag("Jön a hétvége !", 3)),
    SZOMBAT(new Tulajdonsag("Hét vége :))", 5)),
    VASARNAP(new Tulajdonsag("Jaj, mindjárt hétfő", 1));

    private Tulajdonsag tulajdonsag;

    private Nap(Tulajdonsag tulajdonsag) {
        this.tulajdonsag = tulajdonsag;
    }

    public Tulajdonsag getTulajdonsag() {
        return tulajdonsag;
    }
}
```

ÖSSZETETTEBB ENUM PÉLDA

```
class indito3 {

    public static void main(String[] args) {
        Nap[] napok = Nap.values();
        System.out.println("A hét napjai: ");
        for (Nap nap : napok) {
            System.out.println(nap + " tulajdonsága: "
                    + nap.getTulajdonsag());
        }
        Arrays.sort(napok, new TulajdonsagSzerint());
        System.out.println("\nRendezve ");
        for (Nap nap : napok) {
            System.out.println(nap + " tulajdonsága: "
                    + nap.getTulajdonsag());
        }
    }
}
```

```
class Tulajdonsag{
    private String elnevezes;
    private int minosites;

    public Tulajdonsag(String elnevezes, int minosites) {
        this.elnevezes = elnevezes;
        this.minosites = minosites;
    }

    public String getElnevezes() {
        return elnevezes;
    }

    public int getMinosites() {
        return minosites;
    }

    @Override
    public String toString() {
        return elnevezes + ", minositese: " + minosites;
    }
}
```

ÖSSZETETTEBB ENUM PÉLDA

```
class TulajdonsagSzerint implements Comparator<Nap>{

    @Override
    public int compare(Nap o1, Nap o2) {
        return
            o2.getTulajdonsag().getMinosites() -
                    o1.getTulajdonsag().getMinosites();
    }
}
```

KITÉRŐ: ENUM vs ENUMERATION

enum: speciális adattípus

enumeration: egy interfész, az őt megvalósító osztály elemein „végig lehet menni”.

Pl.:

```
for (Enumeration<Rendezveny> e = rendezvenyModel.elements();
                                        e.hasMoreElements();) {
    System.out.println(e.nextElement());
}
```

https://stackoverflow.com/questions/19445183/difference-between-enum-and-enumeration

MÉG EGY, AMIT NEM ÁRT ISMÉT MEGBESZÉLNI

```
public class Diak  {

private List<Tantargy> tantargyak = new ArrayList<>();
```

Probléma:

Ha az eredeti listát adjuk vissza a getterben (vagy bármilyen más metódusban), akkor az kívülről módosítható lesz. ☹

JAVASLATOK AZ ENUM-HOZ

http://javarevisited.blogspot.hu/2011/08/enum-in-java-example-tutorial.html

http://blog.pengyifan.com/how-to-extend-enum-in-java/

MEGOLDÁSOK

Egy lehetséges megoldás:

```
public List<Tantargy> getTantargyak() {
    return new ArrayList<>(tantargyak);
}
```

Másik lehetőség:

```
public List<Tantargy> getTantargyak() {
    return Collections.unmodifiableList(tantargyak);
}
```

Mi a különbség?

MEGOLDÁSOK

Első változat:
A lista = getTantargyak() lista módosítható, de a getTantargyak() hívás eredménye mindig az eredeti lista.

Második változat:
A lista = getTantargyak() lista egyáltalán nem módosítható.