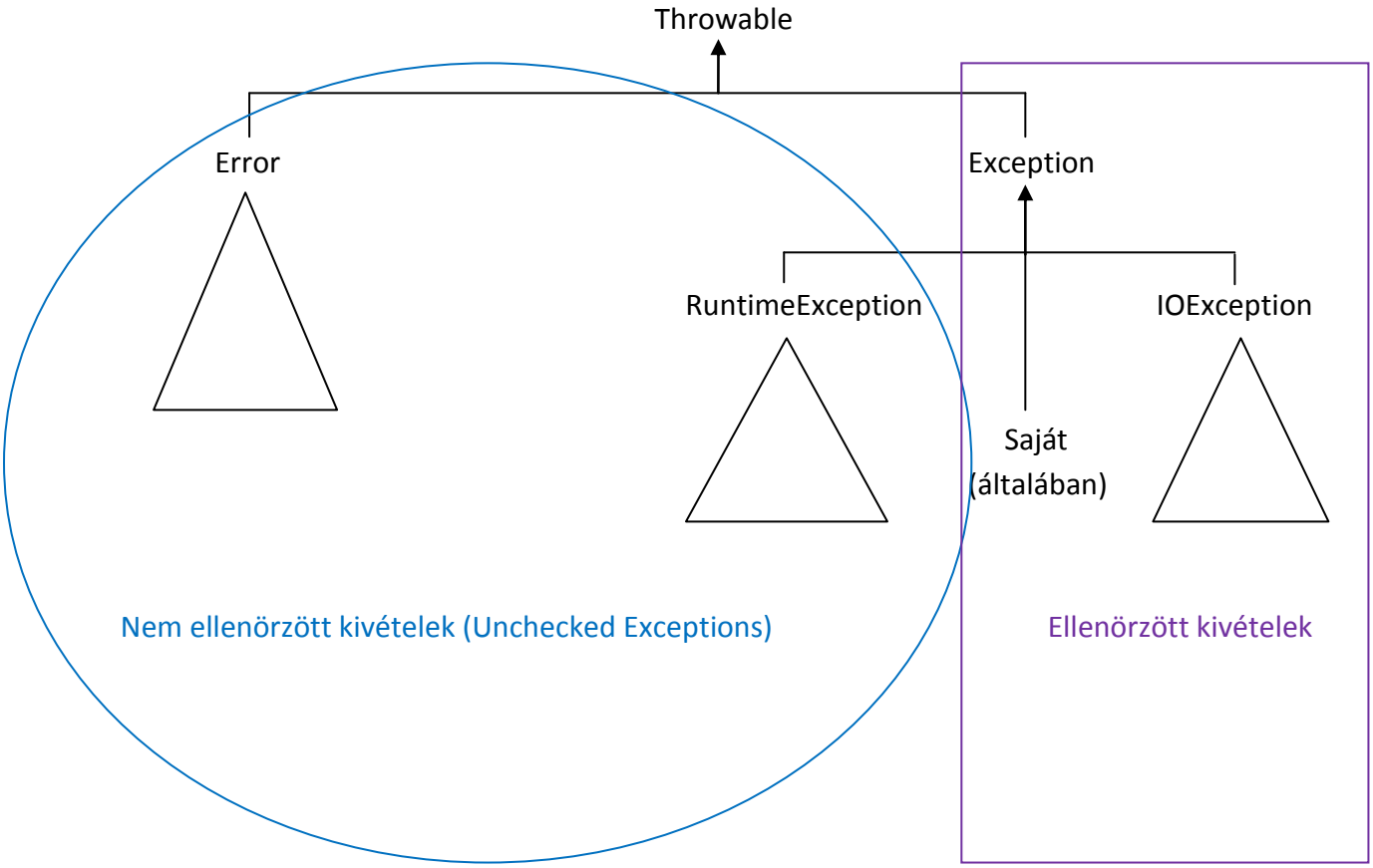
Programozási nyelvek Java

Kozsik Tamás előadása alapján
Készítette: Nagy Krisztián

13. előadás




```
class Stack<T> …{
     public void pop() throws StackException{…}
}

public class StackException extends Exception{}


public class StackException extends Exception{
     public StackException(String message){
          super(message);
     }

     public StackException(){}
}
```

Előfeltétel ellenörzés:

- nyilvános metódusok
- IllegalArgumentException

Belül: assert

- ki-be kapcsolható

Object osztály

```
public class Object{
      public String toString(){…}
      public int hashCode(){…}
      public boolean equals(Object other){…}
      public Class getClass(){…}
      public void finalize(){…}
      protected Object clone() throws CloneNotSupportedException
}

public class Date{

      private int y,m,d;
      …
      //public String toString(){ return y+"/"+m+"/"+d;}
      //Kevésbé hatékony, mivel minden egyes + művelet után új
      //String objektum keletkezik, ovábbá az összefűzésekből is új
      //String objektum lesz és az összefűzést megvalósításához is új
      //objektumok keletkeznek.
      @Override
      public String toString(){
            StringBuilder sb = new StringBuilder();
      return sb.append(Integer.toString(y)).append("/"). …
      .append(Integer.toString(d));}

      @Override
      public boolean equals(Object other){
            if(other == this) return true;
            if(other == null) return false;
            if(getClass().equals(other.getClass())){
                  Date date = (Date) other;
                  return y == date.y && m == date.m && d == date.d;
            } else return false;
      }
```

```
	@Override
	public int hashCode(){
		return d+32*m+512*y;
	}

}
```

Fontos kiegészítések:

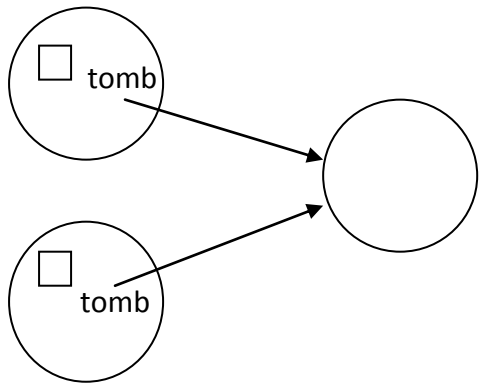
- Ha egy általunk létrehozott objektumban felüldefiniáljuk az equals(…) metódust, akkor kötelező felüldefiniálnunk az objektumunk hashCode() metódusát is.
- Az object osztály finalize() függvényét nem szoktuk használni. Ez a függvény lényegében destruktor szerű szerepet tölt be, viszont nem tudjuk pontosan, hogy a garbage-collector mikor fogja felszabadítani az allokált objektumunkat. Ignoráljuk!
- Az equals(…) metódus alapértelmezett definíciója: return this == other;
- Két adat egyenlő, ha a hashCode() egyenlő
- hashCode() fontos szerepet játszik az adatszerkezeteknél is. Például: HashSet, HashMap

protected Object clone() throws CloneNotSupportedException

if(this instanceof Clonable){DO_MAGIC}
else throw new CloneNotSupportedException();

DO_MAGIC: Alapértelmezetten ***sekély másolás (Shallow copy)***
Clonable: üres interfész, melyre szükség van a clone() használatához
Minden osztály, amely implementálja a Clonable interfészt másolható lesz a clone() segítségével.

Azonban mivel sekély másolást biztosít, így a referencia típusú mezők esetében a másolást követően nem keletkezik új objektum, hanem ugyan arra a helyre fog mutatni a referencia, mint ahova a lemásolandó példányban mutat.



Első sorban primitív típusú mezőket tartalmazó objektumok esetén hasznos, vagy ha azt szeretnénk, hogy a másolt objaktumban levő referenciák is ugyan oda mutassanak, ahova az eredeti objektumban mutattak.

Az előző minta kibővítve::

public class Date implements Clonable{…} –be:

```
@Override public Date clone(){

        try{
            return (Date) super.clone();}
        catch(CloneNotSupportedException e){
            assert false; // ide sose fog ráfutni a program.
            return null;
        }

}
```

Újra deklarálás:
Eredetileg protected Object clone() volt a metódusunk.
Amit tudni kell:

- a láthatóságot szabad bővíteni, de szűkíteni nem!
- a visszatérési érték csak specializálódhat! (Object-ből származik a Date!)
- a paraméterlista típusilag nem változhat, de a formális paraméter nevek, amennyiben vannak változhatnak!
- szerepelhet kevesebb ellenörzött kivétel, de több nem!

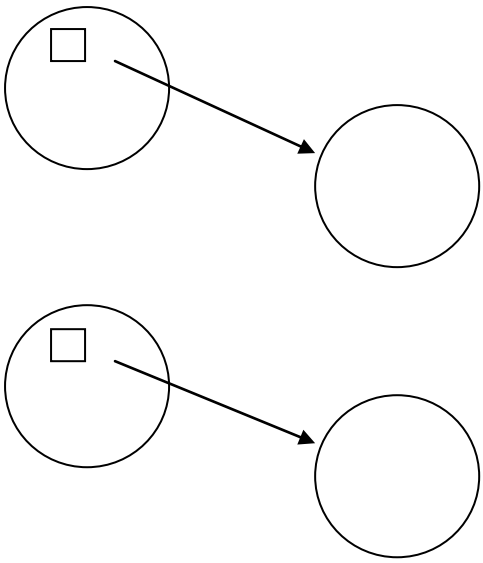
Miért? Azért, mert nem sérti meg a helyettesítési elvet!

Mély másolás (Deep copy):

```
public class Date implements Clonable{

private int[] values;
public Date(int y, int m, int d){

values = {y,m,d};
}
@Override
public Date clone(){
    try{
        Date other = (Date) super.clone();
        other.values = { values[0], values[1], values[2]};
    } catch …
```



Döntés:
Amennyiben asszociációs kapcsolat áll fenn két objektum között, úgy sekély másolást használunk.
Amennyiben aggregáció kapcsolat áll fenn két objektum között, úgy mély másolást, használunk.