Java IX. – Öröklődés

Öröklődésnek nevezzük azt amikor egy osztályt egy másik osztály felhasználásával definiálunk, másszóval egy másik osztályból származtatunk. Így a létrejövő osztályban csak azt kell megfogalmaznunk, miben tér el az eredeti osztálytól.

class A extends B { ... }

ahol A-t, B gyermekosztályának (child class), alosztályának (subclass) vagy származtatott osztályának (derived class) nevezzük. Hasonlóan B, A szülőosztálya (parent class) vagy bázisosztálya (superclass/baseclass). Nyilvánvaló, hogy az öröklődés egy tranzitív folyamat, tehát egy alosztály alosztálya ugyan úgy örökli a bázis osztály minden tulajdonságát.

Vegyünk egy korábbi példát:

public class Time extends java.lang.Object {

private int hour, minute; // initialized to 00:00

public int getHour(){ ... }

public int getMinute(){ ... }

public void setHour( int hour ){ ... }

public void setMinute( int minute ){ ... }

public void aMinutePassed(){ ... }

}

Származtassunk ebből az osztályból egy időt pontosabban kifejező osztályt:

public class ExactTime extends Time {

private int second; // initialized to 00

public int getSecond(){ ... }

public void setSecond( int second ){ ... }

public boolean earlierThan( ExactTime that ){ ... }

}

Érdekesség, hogy minden osztály a java.lang.Object osztályból származik. Például esetünkben:

public class Time extends java.lang.Object {

## Részlet a java.lang.Object-ből:

package java.lang;

public class Object {

public Object(){ ... }

public String toString(){ ... }

public int hashCode(){ ... }

public boolean equals( Object that ){ ... }

...

}

# Konstruktorok:

A konstruktorok nem öröklődnek, kötelesek vagyunk egy új konstruktort írni az alosztályban. Ehhez viszont fel tudjuk használni a szülőosztály konstruktorát.

public ExactTime( int hour, int minute, int second ){

super(hour,minute); // meghívandó a szülő konstruktora

if( second < 0 || second > 59 ) throw new IllegalArgumentException();

this.second = second;

}

A super() egy speciális kulcsszó, mellyel az gyermekosztályból hozzáférhetünk a szülőosztály konstruktorához. Minden esetben a legelső utasításnak kell lennie.

A super-en keresztül minden objektum esetén meghívásra kerül a szülőosztály konstruktora. Még akkor is, ha nem írjuk ki explicit módon akkor implicit módon meghívásra kerül a szülőosztály üres konstruktora.

public class Time extends Object {

private int hour, minute;

public Time( int hour, int minute ){

super();

if( hour < 0 || hour > 23 || minute < 0 || minute > 59 )

throw new IllegalArgumentException();

this.hour = hour;

this.minute = minute;

}

...

}

Tehát itt opcionális az extends Object és a super() is.

Egy konstruktor törzse minden esetben a következők egyike:

1. Más osztálybeli konstruktor explicit meghívása (this-hívás)
2. Explicit super-hívás
3. Implicit super-hívás (paraméter nélküli)

amit a többi utasítás (ami nem this-hívás vagy super-hívás) követ.

# Felüldefiniálás(overriding):

A szülőosztályból megörökölt példányszintű metódusok felüldefiniálhatók. Gyakran felüldefiniált az objektum szöveges reprezentációját előállító toString metódus és a *hashcode* kiszámításáért felelős hashCode metódus. Ezeket minden osztály megörökli a java.lang.Object osztályból.

public class Time {

...

public String toString(){

return hour + ":" + minute; // 8:5

}

}

A felüldefiniálás nem kötelező, de dokumentációs okokból ajánlott az @Override annotáció használata.

public class Time {

...

@Override

public String toString(){ // 8:05

return String.format("%1$d:%2$02d", hour, minute);

}

}

A super referencián keresztül a szülőosztály példánymetódusaihoz is hozzáférhetünk, így például egy származtatott osztályban felhasználható a szülő osztály azon metódusa, melyet az alosztályban éppen felül akarunk írni. Például:

public class ExactTime extends Time {

...

@Override

public String toString(){ // 8:05:17

return super.toString() + String.format(":%1$02d", second);

}

}