Προγραμματισμός ΙΙ

29/4/2020

Εσωτερικές κλάσεις, γενικευμένες κλάσεις και μέθοδοι

Εσωτερικές κλάσεις

- Η Java δίνει τη δυνατότητα να ορίζουμε κλάσεις εσωτερικά μέσα σε άλλες κλάσεις.

- Οι **εσωτερικές κλάσεις** αποσκοπούν κυρίως στην κατασκευή βοηθητικών αντικειμένων που χρησιμοποιούνται για «μικροδουλίτσες» μέσα στις κλάσεις που περιέχουν.

- Έχουν πρόσβαση στα στοιχεία της εξωτερικής τους κλάσης και αντιστρόφως, ακόμα και στα ιδιωτικά (private).

- Μια βοηθητική εσωτερική κλάση δηλώνεται συνήθως ως private.

Εσωτερικές κλάσεις

- Σπανιότερα, μια εσωτερική κλάση μπορεί και να μη δηλωθεί ως private και τα στοιχεία της να είναι προσπελάσιμα και από άλλες κλάσεις.

- **Προσοχή! Οι εσωτερικές κλάσεις δεν είναι υποκλάσεις!**

- Στο παράδειγμα που ακολουθεί περιγράφεται μια κλάση που αναπαριστά έναν τραπεζικό λογαριασμό (bank account), όπως ένα βιβλιάριο τραπεζικού λογαριασμού.

- Ο τραπεζικός λογαριασμός μπορεί να περιλαμβάνει πολλές συναλλαγές (transactions).

Εσωτερικές κλάσεις

- Για μεγαλύτερη ακρίβεια στις συναλλαγές, αντί για τον τύπο double, του οποίου η ακρίβεια είναι πολύ περιορισμένη, χρησιμοποιείται ένας βοηθητικός τύπος που κατασκευάσαμε ειδικά για το συγκεκριμένο παράδειγμα.

- Ο τύπος αυτός περιγράφεται στην εσωτερική κλάση Accurate Number.

- Επιτρέπει την αναπαράσταση χρηματικών ποσών, χρησιμοποιώντας δύο μεγάλους ακέραιους (long).

- Η κλάση Accurate Number δηλώθηκε ως εσωτερική γιατί πρόκειται να χρησιμοποιηθεί μόνο από την Bank Account.

Bank account – κώδικας

```
public class BankAccount
{
        AccurateNumber[] transaction;
        int currentTransaction;

        public BankAccount(int numberOfTransactions)
        {
                transaction=new AccurateNumber[numberOfTransactions];
                currentTransaction=0;
        }
```

Bank account – κώδικας

```
public void deposit(long amountEuros, long amountCents)
{
        transaction[currentTransaction]=new AccurateNumber();
        transaction[currentTransaction].euros=amountEuros;
        transaction [currentTransaction].cents=amountCents;
        currentTransaction++;
}
```

Bank account – κώδικας

```
public String toString()
{
        String s="";
        for (int i=0;i<currentTransaction;i++)
        {       s+=transaction[i].euros+"."+transaction[i].cents+"\n";
        }
        return s;
}

//Inner class
private class AccurateNumber
{       long euros,cents;
}
```

Bank account – κώδικας

```
        public static void main(String[] args)
        {
                BankAccount b=new BankAccount(100);
                b.deposit(1000,20);
                b.deposit(100,21);

                System.out.println(b);
        }
}
```

Γενικευμένες κλάσεις

- Οι γενικευμένες κλάσεις δίνουν τη δυνατότητα να ορίζουμε ως παραμέτρους τους τύπους των μεταβλητών υπόστασής τους, ή/και των μεθόδων τους.

- Ο παραμετρικά οριζόμενος τύπος εισάγεται δίπλα από το όνομα της κλάσης με χρήση οξυγώνιων παρενθέσεων.

- Στο παράδειγμα που ακολουθεί, η κλάση GenericClass είναι τόσο γενική που μπορεί να χρησιμοποιηθεί εξίσου, είτε με ακεραίους, είτε με αλφαριθμητικά στη θέση του τύπου T.

Γενικευμένες κλάσεις

- Ο τύπος Τ πρέπει να είναι τύπος κλάσης. Αν επιθυμούμε να έχουμε πρωταρχικό τύπο, πχ int, θα πρέπει να χρησιμοποιήσουμε την αντίστοιχη περιβάλλουσα κλάση, πχ την κλάση Integer αντί για τον τύπο int.

- Μια γενικευμένη κλάση μπορεί να έχει περισσότερες από μια παραμέτρους, πχ GenericClass<T1,T2>.

GenericClass– κώδικας

```
public class GenericClass<T>
{
        T x;
        public void print()
        {
                System.out.println(x);
        }
}
```

GenericClass– κώδικας

```
public class GenericTest
{
        public static void main(String[] args)
        {
                GenericClass<Integer> a=new GenericClass<Integer>();
                a.x=1230;
                a.print();

                GenericClass<String> b=new GenericClass<String>();
                b.x="Hello Generics";
                b.print();
        }
}
```

Λίστες

- Δομές δεδομένων, όπως οι πίνακες, που επιτρέπουν όμως την επέκτασή τους με νέα δεδομένα ή την περικοπή τους, ανά πάσα στιγμή, χωρίς να χάνονται τα δεδομένα τους (όπως συμβαίνει με τους πίνακες).

- Στη Java οι λίστες υλοποιούνται μέσω της γενικευμένης κλάσης ArrayList ή της κλάσης Vector, του πακέτου java.util.

- Στο παράδειγμα που ακολουθεί, η χρήση της κλάσης ArrayList αντιπαραβάλλεται με αυτή των πινάκων.

- Οι πίνακες μπορούν και αυτοί να αλλάζουν μέγεθος δυναμικά με χρήση της new, αλλά έτσι χάνονται τα περιεχόμενά τους.

Λίστες

- Ενδεικτικές μέθοδοι:

  - add(element)-προσθέτει ένα νέο στοιχείο στο τέλος της λίστας
  - add(pos,element)-προσθέτει ένα νέο στοιχείο στη θέση pos της λίστας
  - set(pos,element)-θέτει νέο στοιχείο στην υπάρχουσα θέση pos της λίστας
  - remove(pos)-διαγράφει όποιο στοιχείο βρίσκεται στη θέση pos
  - remove(element)-διαγράφει το στοιχείο element (την πρώτη εμφάνισή του)

ArrayList– κώδικας

```
import java.util.ArrayList;

class TestArrayList
{
        public static void main(String[] args)
        {
                //How it is done with arrays
                int[] a={5,1,2};

                //We can change length later (dynamic allocation)
                a=new int[10];

                //But the contents of a are lost!
```

ArrayList– κώδικας

```
//How it is done with lists
ArrayList<String> list=new ArrayList<String>(1);

//Add new elements to the list
list.add(“Maria”);
list.add(“George”);
list.add(“John”);
list.add(“Costas”);

System.out.println(“Before…”);
```

ArrayList– κώδικας

```
//Classic ‘for’ loop to see what is in the list
for (int i=0;i<list.size();i++)
{
        System.out.println(list.get(i));
}

//Change the second element to ‘Anna’
list.set(2,“Anna”);

//Change the first element to ‘Ioanna’
list.add(1,“Ioanna”);
```

ArrayList– κώδικας

```
//Remove the second element
list.remove(2);

//Remove the element ‘Costas’
list.remove(“Costas”);

System.out.println(“After…”);

//New loop to see contents of the list
for (String element.list)
{
        System.out.println(element);
}}}
```

Εισαγωγή στην κληρονομικότητα

Υποκλάσεις

- Έστω ότι επιθυμούμε να κατασκευάσουμε μια νέα κλάση Β η οποία έχει τις ίδιες μεταβλητές υπόστασης και μεθόδους με μια κλάση Α, αλλά έχει και κάποιες επιπλέον μεταβλητές υπόστασης και μεθόδους.

- Μπορούμε να δηλώσουμε ότι η Β «επεκτείνει» (extends) την Α, ώστε αυτομάτως να κληρονομήσει (inherit) τα στοιχεία της Α.

- Έτσι δε χρειάζεται να υλοποιήσουμε ξανά τα στοιχεία της Α.

- Στο παράδειγμα που ακολουθεί, η κλάση Date αναπαριστά μια ημερομηνία και η DateTime επεκτείνει την κλάση Date με την ώρα.

Υποκλάσεις

- Η DateTime ονομάζεται υποκλάση (subclass) της κλάσης Date.
- Η κλάση Date ονομάζεται υπερκλάση (superclass) της DateTime.
- Μια κλάση στη Java μπορεί να έχει μόνο μια υπερκλάση (μονή κληρονομικότητα)
- Σε άλλες αντικειμενοστρεφείς γλώσσες, όπως η C++, υποστηρίζεται και πολλαπλή κληρονομικότητα (δηλαδή μια κλάση μπορεί να έχει πολλές υπερκλάσεις).
- **Η DateTime είναι και Date. Η Date δεν είναι και DateTime.**

Date– κώδικας

```
class Date
{
        int day,month,year;
}

class DateTime extends Date
{
        int hour,minute,second;

        public static void main(String[] args)
        {
                DateTime dt=new DateTime();
```

Date– κώδικας

```
//DateTime inherits day,month,year
dt.day=20;
dt.month=10;
dt.year=2010;

//DateTime has also hour,minute,second
dt.hour=8;
dt.minute=12;
dt.second=25;
```

Date– κώδικας

```
System.out.println(dt.day+“/”+dt.month+“/”+dt.year+“ ”
+dt.hour+“:”+dt.minute+“:”+dt.second);
```

Σημείωση: Οι ιδιωτικές (private) μεταβλητές υπόστασης και οι μέθοδοι μιας κλάσης δεν είναι άμεσα προσπελάσιμες από τις υποκλάσεις της. Είναι προσπελάσιμες μόνο μέσω μεθόδων προσπέλασης και μεταλλαγής.

Μέθοδοι δημιουργοί υποκλάσεων

- Στις μεθόδους δημιουργούς υποκλάσεων πρέπει πάντα στην πρώτη γραμμή τους να καλείται η μέθοδος δημιουργός της υπερκλάσης τους.

- Αυτό γίνεται με τη χρήση της έκφρασης **super**, η οποία είναι ανάλογη της έκφρασης **this** και αναφέρεται στην υπερκλάση.

- Η **super** ακολουθούμενη από τελεία, πχ **super.day**, αναπαριστά ένα αντικείμενο της υπερκλάσης της κλάσης μας, ενώ ακολουθούμενη από παρενθέσεις, πχ **super()**, αναπαριστά μια μέθοδο δημιουργό της υπερκλάσης της κλάσης μας.

- Ακολουθεί το προηγούμενο παράδειγμα, εμπλουτισμένο με μεθόδους δημιουργούς.

Date– κώδικας

```
class Date
{
        int day,month,year;

        Date(int d, int m, int y)
        {
                day=d;
                month=m;
                year=y;
        }
}
```

Date– κώδικας

```
class DateTime extends Date
{
        int hour,minute,second;

        DateTime(int d,int mo, int y, int h, int m, int s)
        {
                super(d,mo,y);//should always be first

                hour=h;
                minute=m;
                second=s;
        }
```

Date– κώδικας

```
        public static void main(String[] args)
        {
                DateTime dt=new DateTime(29,4,2020,6,19,0);
                System.out.println(dt.day+“/”+dt.month+“/”+dt.year+“ ”
                +dt.hour+“:”+dt.minute+“:”+dt.second);
        }
}
```

Σημείωση: αν δεν προστεθεί η **super** μέσα στη μέθοδο δημιουργό DateTime, τότε η Java καλεί αυτομάτως τη **super()**, δηλαδή τη μέθοδο δημιουργό της Date που δεν έχει ορίσματα. Στο παραπάνω παράδειγμα όμως, δεν υπάρχει τέτοια μέθοδος δημιουργός, με αποτέλεσμα σε αυτήν την περίπτωση να μη γίνεται μεταγλώττιση.

Υπέρβαση μεθόδων

- Μια υποκλάση κληρονομεί όλες τις μεθόδους της υπερκλάσης της.

- Αν έχει μια μέθοδο με την ίδια επικεφαλίδα με μια μέθοδο της υπερκλάσης της, τότε η μέθοδος αυτή **υπερβαίνει (overrides)** τη μέθοδο της υπερκλάσης.

- Αυτό σημαίνει ότι για την υποκλάση καταργείται η αντίστοιχη μέθοδος της υπερκλάσης και θα ισχύει μόνο η δική της, η οποία ορίστηκε τελευταία.

Υπέρβαση μεθόδων

- Ακολουθεί το προηγούμενο παράδειγμα, εμπλουτισμένο με τη μέθοδο toString. Η μέθοδος toString της DateTime υπερβαίνει τη μέθοδο toString της Date. Έτσι για την DateTime θα ισχύει η δική της toString, και όχι η toString της Date.

- Ωστόσο, μπορούμε να καλούμε τη μέθοδο toString της Date μέσα από την DateTime. Αυτό επιτυγχάνεται με τη χρήση της έκφρασης **super**, ως αντικείμενο κλήσης, δηλαδή super.toString().

Date– κώδικας

```
class Date
{
        int day,month,year;

        Date(int d, int m, int y)
        {
                day=d;
                month=m;
                year=y;
        }

        public String toString()
        {return day+“/”+month+“/”+year;} }
```

Date– κώδικας

```
class DateTime extends Date
{
        int hour,minute,second;

        DateTime(int d,int mo, int y, int h, int m, int s)
        {
                super(d,mo,y);//should always be first

                hour=h;
                minute=m;
                second=s;
        }
```

Date– κώδικας

```
    public String toString()
    {
        //The following calls toString of Date
        return super.toString()+“ ”+hour+“:”+minute+“:”+second;
    }

    public static void main(String[] args)
    {
        DateTime dt=new DateTime(29,4,2020,6,19,0);
        System.out.println(dt.day+“/”+dt.month+“/”+dt.year+“ ”
        +dt.hour+“:”+dt.minute+“:”+dt.second);
    }
}
```