Αντικειμενοστρεφής Προγραμματισμός

Παναγιώτης Αδαμίδης
adamidis@ihu.gr

Σύνθεση, Κληρονομικότητα

Αναδρομικές κλάσεις

Σύνθεση (Composition)

- Σύνθεση έχουμε όταν κατά τη δημιουργία μιας νέας κλάσης χρησιμοποιούμε ως χαρακτηριστικά τις αναφορές σε αντικείμενα άλλων κλάσεων.
- Κατά τη σύνθεση δημιουργείται μία σχέση «Έχει-ένα» (has-a ή whole/part συσχέτιση).
- Όταν ένα χαρακτηριστικό μίας κλάσης είναι αναφορά σε αντικείμενο άλλης κλάσης, τότε τα αντικείμενα αυτής της κλάσης είναι σύνθετα αντικείμενα (composite objects).
- Αν ένα αντικείμενο Β περιέχεται σε ένα αντικείμενο Α τότε το αντικείμενο Α είναι υπεύθυνο για τη δημιουργία και καταστροφή του αντικειμένου Β.

Παράδειγμα Σύνθεσης - 1

1

```
class WaterSource {
   private String s;
   WaterSource(){
      System.out.println ("WaterSource()");
      s=new String("Constructed");
   }
   public String toString() {
      return s;
   }
}
```

Παράδειγμα Σύνθεσης - 1

2

```
public class SprinklerSystem {
    private String valve1, valve2, valve3, valve4;
    WaterSource source;
    int i;
    float f;
    void print() {
       System.out.println("valve1="+valve1);
       System.out.println("valve2="+valve2);
       System.out.println("valve3="+valve3);
       System.out.println("valve4="+valve4);
       System.out.println("i="+i);
       System.out.println("f="+f);
       System.out.println("source="+source);
    }
    public static void main (String[] args) {
       SprinklerSystem x=new SprinklerSystem();
       x.print();
    }
}
```

Σύνθεση: Παράδειγμα - Έξοδος

3

- Έξοδος

```
C:\Panos\Progs>java SprinklerSystem
valve1=null
valve2=null
valve3=null
valve4=null
i=0
f=0.0
source=null
```

- Εάν καλέσουμε μέθοδο αντικειμένου του οποίου η αναφορά είναι null τότε δημιουργείται εξαίρεση (εκτός της εμφάνισης).

Σύνθεση: Αρχικοποίηση αναφορών

- Είναι λογικό να μην δημιουργείται ένα προκαθορισμένο αντικείμενο κάθε αναφορά γιατί έτσι θα δημιουργούσε "overhead". Η απόδοση αρχικών τιμών στις αναφορές μπορεί να γίνει:
- Στο σημείο ορισμού των αντικειμένων. Αυτό σημαίνει ότι θα αρχικοποιούνται πάντα πριν την κλήση του δομητή.
- Στον δομητή της κλάσης.
- Ακριβώς πριν την χρήση του αντικειμένου (lazy initialization). Μπορεί να μειώσει το "overhead" σε καταστάσεις όπου το αντικείμενο δεν χρειάζεται να δημιουργείται κάθε φορά.

Σύνθεση: Αρχικοποίηση αναφορών Παράδειγμα (1)

```
class Soap {
   private String s;
   Soap() {

  System.out.println("Soap()");
       s=new
  String("Constructed");
    }
    public String toString() {
       return s;
    }
}
```

Σύνθεση: Αρχικοποίηση αναφορών Παράδειγμα (2)

```
public class Bath {
  private String
    //Initializing at point of definition:
    s1=new String("Happy"),
    s2="Happy",  s3, s4;
  Soap castille;
  int i;
  float toy;
  Bath() {
      System.out.println("Inside Bath()");
      s3=new String("Joy");
      i=47;
      toy=3.14f;
      castille=new Soap();
  }
```

Σύνθεση: Αρχικοποίηση αναφορών Παράδειγμα (3)

```
void print() {
      //Delayed initialization:
      if (s4==null)
          s4=new String("Joy");
      System.out.println("s1="+s1);
      System.out.println("s2="+s2);
      System.out.println("s3="+s3);
      System.out.println("s4="+s4);
      System.out.println("i="+i);
      System.out.println("toy="+toy);
      System.out.println("castille="+castille);
  }
  public static void main(String[] args) {
      Bath b=new Bath();
      b.print();
  }
}
```

Σύνθεση: Αρχικοποίηση αναφορών Παράδειγμα (4)

- Έξοδος:

```
C:\Progs>java Bath
Inside Bath()
Soap()
s1=Happy
s2=Happy
s3=Joy
s4=Joy
i=47
toy=3.14
castille=Constructed
```

Παράδειγμα Σύνθεσης: Name 1

```
class Name{                    // Object represent names of people
    private String first;
    private String middle;
    private String last;
     Name(){} //default constructor
    Name(String first, String last){
        this.first = first;        this.last = last;
    }
    Name(String first, String middle, String last){
        this(first,last);          this.middle = middle;
    }
    String first() { return first; }
    String middle() { return middle; }
    String last() { return last; }
```

Παράδειγμα Σύνθεσης: Name 2

```
    void setFirst(String first) { this.first  = first; }
    void setMiddle(String middle) { this.middle =middle;
    }
    void setLast(String last) { this.last = last; }
    public String toString() {
        String s=new String();
        if (first != null) s += first + " ";
        if (middle != null) s += middle + " ";
        if (last != null) s += last + " ";
        return s.trim();
    }
}
```

Παράδειγμα Σύνθεσης: Name 3

```
class TestName{
   //Test driver for Name class
   public static void main(String[] args){
      Name tr=new Name("Werner", "Heisenberg");
      Name fc=new Name("Francis", "Harry Compton", "Crick");
      System.out.println(fc + "won the 1962 Nobel in Physiology.");
      System.out.println("His first name was " + fc.first());
      System.out.println(tr + " won the 1932 Nobel in Physics.");
      System.out.println("His middle name was " + tr.middle());
   }
}
```

Παράδειγμα Σύνθεσης: Name 4

- Έξοδος:

```
run:
Francis Harry Compton Crick won the 1962 Nobel in Physiology.
His first name was Francis
Werner  Heisenberg won the 1932 Nobel in Physics.
His middle name was null
```

Παράδειγμα Σύνθεσης: Person 1

```
class Person{              //objects represent people
   protected Name name;
   protected char sex;     //  ´M´ or ´F´
   protected String id;
   Person() {}
   Person(Name name, char sex){
      this.name = name;
      this.sex = sex;
   }
   Person(Name name, char sex , String id){
      this.name  = name;
      this.sex = sex;
      this.id = id;
   }
```

Γιατί protected;
Πότε χρειάζεται;

Θα μπορούσαμε να έχουμε άλλους δομητές (constructors);

Παράδειγμα Σύνθεσης: Person 2

```
Name name(){ return name; }
char sex(){ return sex; }
String id(){ return id; }
void setId(String id){ this.id = id; }
public String toString(){
   String s = new String(name + " (sex: " + sex);
   if (id != null) s += "; id: " + id;
   s += ")";
   return s;
}
}
```

Ποιες μέθοδοι απουσιάζουν;

Παράδειγμα Σύνθεσης: Person 3

```
class TestPerson{
   // Test driver for the Person class:
   public static void main(String[] args){
      Name georgeName = new Name("Georgios", "Seferis");
      Person george = new Person(georgeName, 'M');
      System.out.println("george: " + george);
      george.name.setMiddle("Stelios");
      System.out.println ("george: " + george);
      Person marika = new Person(new Name ("Marika", "Kotopouli"), 'F');
      System.out.println("marika: " + marika);
      marika.setId("A101010");
      System.out.println("marika: " + marika);    }
}
```

Παράδειγμα Σύνθεσης: Person 4

- Έξοδος:

```
run:
george: Georgios Seferis (sex: M)
george: Georgios Stelios Seferis (sex: M)
marika: Marika Kotopouli (sex: F)
marika: Marika Kotopouli (sex: F; id: A101010)
```

Αναδρομικές Κλάσεις

- Αναδρομική μέθοδος είναι αυτή που καλεί τον εαυτό της. Αναδρομική κλάση είναι αυτή που συντίθεται χρησιμοποιώντας τον εαυτό της, δηλ. έχει **τουλάχιστον ένα** μέλος το οποίο είναι αναφορά σε αντικείμενο της κλάσης που ανήκει.
- Οι αναδρομικές κλάσεις παρέχουν μία ισχυρή τεχνική δημιουργίας συνδεδεμένων δομών οι οποίες μπορούν να αναπαραστήσουν αποτελεσματικά πολύπλοκες δομές..

Παράδειγμα Σύνθεσης: Person (Recursive) 1

```
class Person{              //objects represent people
    private Name name;
    private Person mother;
    private Person father;
    private char sex;      //  'M' or 'F'
    private String id;
    private static final String twoBlanks = "  ";
    private static String tab = "" ;

    // Constructors
    Person() {}
    Person(Name name) { this.name = name; }
    Person(char sex) { this.sex = sex; }
    Person(String id) { this.id = id; }
```

Παράδειγμα Σύνθεσης: Person (Recursive) 2

```
Person(Name name, char sex) {
   this.name = name;        this.sex = sex;
}
Person(Name name, String id) {
   this.name  = name;       this.id = id;
}
Person(char sex , String id) {
    this.sex = sex;         this.id = id;
}
Person(Name name, char sex , String id) {
   this.name  = name;
   this.sex = sex;
   this.id = id;
}
```

Παράδειγμα Σύνθεσης: Person (Recursive) 3

```
public Name getName() { return name; }
public void setName (Name name) { this.name = name; }
public Person getMother() { return mother; }
public void setMother (Person m) { mother = m; }
public Person getFather() { return father; }
public void setFather (Person f) { father = f; }
public char getSex() { return sex; }
public void setSex (char s) { sex = s; }
public String getId() { return id; }
public void setId(String id) { this.id = id; }
```

Παράδειγμα Σύνθεσης: Person (Recursive) 4

```
public String toString(){
    String s = new String(name + " (" + sex + ")");
    if (id != null) s += "; id: " + id;
    s += "\n";
    if (mother != null ){
        tab += twoBlanks;           // adds two banks
        s += tab + "mother: " + mother;
        tab = tab.substring(2);     // removes two blanks
    }
    if(father != null){
        tab += twoBlanks;       // adds two blanks
        s += tab + "father: " + father;
        tab = tab.substring(2); // removes two blanks
    }
    return s;
  }
}
```

Παράδειγμα Σύνθεσης: Person (Recursive) 5

```
class TestPersonRec {
// Test driver for the Person class:
  public static void main(String[] args){
    Person ww = new Person(new Name("William", "Windsor"), 'M');
    Person cw = new Person(new Name("Charles", "Windsor"), 'M');
    Person ds = new Person(new Name("Diana", "Spencer"), 'F');
    Person es = new Person(new Name("Edward", "Spencer"), 'M');
    Person ew = new Person(new Name("Elizabeth", "Windsor"), 'F');
    Person pm = new Person(new Name("Phillip", "Mountbatten"), 'M');
    Person eb = new Person(new Name("Elizabeth","Bowes-Lyon"), 'F');
    Person gw = new Person(new Name("George", "Windsor"), 'M');
```

Παράδειγμα Σύνθεσης: Person (Recursive) 6

```
        ww.setFather(cw);
        ww.setMother(ds);
        ds.setFather(es);
        cw.setMother(ew);
        cw.setFather(pm);
        ew.setMother(eb);
        ew.setFather(gw);
        System.out.println(ww);
      }
    }
```

Παράδειγμα Σύνθεσης: Person (Recursive) 7

- Έξοδος:

```
run:
William Windsor (M)
  mother: Diana Spencer (F)
    father: Edward Spencer (M)
  father: Charles Windsor (M)
    mother: Elizabeth Windsor (F)
      mother: Elizabeth Bowes-Lyon (F)
      father: George Windsor (M)
    father: Phillip Mountbatten (M)
```

Συνδυασμός Σύνθεσης & Κληρονομικότητας 1

```
class Plate {
   Plate(int i) { System.out.println("Plate constructor: "+i); }
}
class DinnerPlate extends Plate {
   DinnerPlate(int i) {
      super(i);
      System.out.println("DinnerPlate constructor: "+i);
   }
}
class Utensil {
   Utensil(int i) { System.out.println("Utensil constructor: "+i);
   }
}
```

Συνδυασμός Σύνθεσης & Κληρονομικότητας 2

```
class Spoon extends Utensil {
  Spoon(int i) {
     super(i);
     System.out.println("Spoon constructor: "+i);
  }
}

class Fork extends Utensil {
  Fork(int i) {
    super(i);
    System.out.println("Fork constructor: "+i);
  }
}
```

Συνδυασμός Σύνθεσης & Κληρονομικότητας 3

```
class Knife extends Utensil {
  Knife(int i) {
    super(i);
    System.out.println("Knife constructor: "+i);
  }
}

class Custom {
  Custom(int i) {
    System.out.println("Custom constructor: "+i);
  }
}
```

Συνδυασμός Σύνθεσης & Κληρονομικότητας 4

```
public class PlaceSetting extends Custom {
    Spoon sp;       Fork frk;
    Knife kn;       DinnerPlate pl;
    PlaceSetting(int i) {
       super(i + 1);
       sp = new Spoon(i + 2);
       frk = new Fork(i + 3);
       kn = new Knife(i + 4);
       pl = new DinnerPlate(i + 5);
       System.out.println("PlaceSetting constructor: " + i);
    }

    public static void main(String[] args) {
        PlaceSetting x = new PlaceSetting(9);
    }
}
```

Συνδυασμός Σύνθεσης & Κληρονομικότητας 5

- Έξοδος:

```
run:
Custom constructor: 10
Utensil constructor: 11
Spoon constructor: 11
Utensil constructor: 12
Fork constructor: 12
Utensil constructor: 13
Knife constructor: 13
Plate constructor: 14
DinnerPlate constructor: 14
PlaceSetting constructor:9
```

Απόκρυψη Ονόματος 1

```
class Homer {
    char doh(char c) {
        System.out.println("doh(char)");
        return 'd';
    }
    float doh(float f) {
        System.out.println("doh(float)");
        return 1.0f;
    }
}

class Milhouse {}
```

Απόκρυψη Ονόματος 2

```
class Bart extends Homer {
    void doh(Milhouse m) {}
}
class Hide {
    public static void main(String[] args) {
        Bart b = new Bart();
        b.doh(1);
        b.doh('x');
        b.doh(1.0f);
        b.doh(new Milhouse());
    }
}
```

Έξοδος:
doh(float)
doh(char)
doh(float)

Σύνθεση με public μέλη 1

```
class Engine {
    public void start() {}
    public void rev() {}
    public void stop() {}
}
class Wheel { public void inflate(int psi) {} }
class Window {
    public void rollup() {}
    public void rolldown() {}
}
class Door {
    public Window window = new Window();
    public void open() {}
    public void close() {}
}
```

Σύνθεση με public μέλη 2

```
public class Car {
    public Engine engine = new Engine();
    public Wheel[] wheel = new Wheel[4];
    public Door left = new Door(),
        right = new Door(); // 2-door
    public Car() {
        for(int i = 0; i < 4; i++)
            wheel[i] = new Wheel();
    }
    public static void main(String[] args) {
        Car car = new Car();
        car.left.window.rollup();
        car.wheel[0].inflate(30);
    }
}
```

Σύνθεση με public μέλη: Γιατί;

- Επειδή η σύνθεση ενός αυτοκινήτου είναι μέρος της ανάλυσης του προβλήματος (και όχι απλά μέρος της σχεδίασης) κάνοντας τα μέλη **public** βοηθάει τον προγραμματιστή να κατανοήσει πώς να χρησιμοποιήσει την κλάση και απαιτεί μικρότερη πολυπλοκότητα του κώδικα της κλάσης. **Γενικά, θα πρέπει να δηλώνουμε τα πεδία/χαρακτηριστικά ως private**.
- Κατά την κληρονομικότητα, παίρνουμε μία υπάρχουσα κλάση και την εξειδικεύουμε με τη δημιουργία μίας ειδικής έκδοσή της. Γενικά, αυτό σημαίνει ότι παίρνετε μία κλάση γενικού σκοπού και την εξειδικεύετε σε μία συγκεκριμένη περίπτωση. Για παράδειγμα δεν έχει νόημα να συνθέσετε ένα αυτοκίνητο χρησιμοποιώντας ένα αντικείμενο τύπου όχημα. Το αυτοκίνητο δεν **περιέχει** ένα όχημα αλλά **είναι** ένα όχημα.

Ερωτήσεις;