



Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών

ΗΥ252 – Αντικειμενοστρεφής Προγραμματισμός

Εξάμηνο: Χειμερινό 2013-2014

Διδάσκων: Γιάννης Τζίτζικας

Βοηθοί: Βασίλης Ευθυμίου, Χριστίνα Λαντζάκη, Παναγιώτης Παπαδάκος, Μιχάλης Χόρτης

2^η Σειρά Ασκήσεων

Ανάθεση: Πέμπτη **7 Νοεμβρίου 2013**

Παράδοση: Πέμπτη **28 Νοεμβρίου 2013**

Εκπαιδευτικοί στόχοι

- Η 1^η άσκηση θα σας επιτρέψει να εξοικειωθείτε με τη δημιουργία αντικειμένων, με τις υποκλάσεις, την υποσκέλιση μεθόδων (overriding), την επισκίαση πεδίων (shadowing) και τα στατικά (static) μέλη κλάσεων.
- Η 2^η άσκηση θα σας επιτρέψει να εξοικειωθείτε με τις διεπαφές (interfaces), τις αφηρημένες (abstract) κλάσεις και μεθόδους.
- Η 3^η και η 4^η άσκηση θα σας επιτρέψουν να εξοικειωθείτε με τη σχεδίαση μιας κλάσης ή ενός αφαιρετικού τύπου δεδομένων, ήτοι:
 - πώς να υλοποιήσω μια κλάση ώστε να σέβεται το συμβόλαιό της;
 - πώς να τεκμηριώσω την κλάση (javadoc)
 - πώς να ελέγχω αυτόματα και περιοδικά την ορθότητα (γενική και ως προς το συμβόλαιο) της υλοποίησης (JUnit tests);

Σημειώσεις:

Θα γίνει και φροντιστήριο σχετικό με JavaDoc και JUnit. Τα generics θα διδαχθούν αναλυτικά αργότερα, στην παρούσα φάση απλά θα τα χρησιμοποιήσετε.



Άσκηση 1. Τμήματα και Σπουδαστές

Αξία: 20%

Δημιουργία πολλών συνδεδεμένων κλάσεων

1(α) [10%]

Ορίστε μία κλάση **Person** με τα εξής "πεδία": **όνομα** (String), **επώνυμο** (String) και **αριθμό ταυτότητας** (String). Ορίστε μία κλάση **Department** με τα εξής πεδία: **όνομα** (String), και **community** το οποίο θα είναι πίνακας που θα μπορεί να κρατάει αντικείμενα τύπου **Person**. Ορίστε μία κλάση **Student** που να εξειδικεύει την **Person** και έχει δύο επιπλέον πεδία: ένα πεδίο **AM** (String), και ένα **department** του οποίου η τιμή μπορεί να είναι ένα στιγμιότυπο της κλάσης **Department**. Ορίστε μία κλάση **GradStudent** που να εξειδικεύει την **Student** και να έχει ένα πεδίο **thesisArea** (String).

Λάβετε υπόψη σας τα εξής:

- Τα πεδία όλων των παραπάνω κλάσεων πρέπει να είναι **ιδιωτικά** (private).
- Όλες οι κλάσεις πρέπει να έχουν μία μέθοδο **toString()** η οποία να εκτυπώνει τα αντίστοιχα στοιχεία (π.χ. για έναν **GradStudent**, το **όνομα**, **επώνυμο**, **αριθμό ταυτότητας** (String), **AM**, το **όνομα του Τμήματος** στο οποίο σπουδάζει, και τη **θεματική περιοχή** της διατριβής του).
- Όλες οι κλάσεις πρέπει να έχουν μία **κατασκευάστρια μέθοδο** η οποία να παίρνει τις τιμές που χρειάζεται. Η κατασκευάστρια της **Department** θα λαμβάνει ως παράμετρο μόνο το **όνομα του Department**. Επίσης πρέπει να προσφέρει μεθόδους για την προσθήκη και διαγραφή σπουδαστών, συγκεκριμένα μία **add(Student)** και μία **delete(AM)**.

Για να δοκιμάσετε τα παραπάνω, ορίστε μια κλάση **UnivTester** της οποίας η **main()** θα δημιουργεί δύο στιγμιότυπα του **Department** και θα προσθέτει σε κάθε ένα από αυτά 200 προπτυχιακούς και 24 μεταπτυχιακούς (διαφορετικούς). Μετά θα καλεί την **toString()** του κάθε τμήματος αυτού η οποία θα πρέπει να εμφανίζει τα στοιχεία του τμήματος, δηλαδή το **όνομά του**, και τα στοιχεία των μελών του (συγκεκριμένα θα καλείται η **toString()** σε κάθε στοιχείο του πίνακα **community**).

1(β) [10%]

Επίσης κάντε ό,τι επέκταση σας φαίνεται λογική και χρήσιμη ώστε:

(α) να υπάρχει ένα φαινόμενο **shadowing** και

(β) να υπάρχει κάτι (πεδίο ή μέθοδος) **static**.

Σημειώστε το τι κάνατε για αυτό το θέμα μέσα στον κώδικα με σχόλια.



Άσκηση 2. Απλός Πυρήνας για Παιχνίδια Ρόλων (core for Role Playing Games)

Αξία: 35%

Interfaces, Inheritance, Subclasses, Collaborating Objects

Η διαφορά με την προηγούμενη άσκηση είναι ότι εδώ θα χρησιμοποιήσετε και *interfaces* και *abstract* κλάσεις.

Σε αυτή την άσκηση καλείστε να μοντελοποιήσετε ένα τμήμα του περιβάλλοντος ενός παιχνιδιού ρόλων. Σε αυτό το παιχνίδι εμφανίζονται διάφορες οντότητες, όπως **μάγοι** (Wizards), **ξωτικά** (Elves), **ορκς** (Orcs) και **τρολς** (Trolls).

Μάγοι (wizards)	Ξωτικά (Elves)	Ορκς (Orcs)	Τρολς (Trolls)
			

Όλες οι οντότητες μοιράζονται κάποια κοινά χαρακτηριστικά, όπως **όνομα**, **δύναμη**, **υγεία** και έχουν τη δυνατότητα να φέρουν κάποιο **όπλο**, το οποίο αυξάνει τη δύναμή τους. Ωστόσο κάθε οντότητα έχει και δικά της, μοναδικά χαρακτηριστικά, όπως π.χ. οι **μάγοι**, που μπορούν να κάνουν κάποιο **ξόρκι**, και τα **ξωτικά** που μπορούν να γίνουν **αόρατα**. Κάθε **ορκ** βρίσκεται υπό την ηγεσία κάποιου **τρολ**.

Οι **παίχτες** έχουν τη δυνατότητα να χειριστούν μόνο δύο είδη οντοτήτων, τους **μάγους** και τα **ξωτικά**, οι οποίες υλοποιούν το interface Playable και δίνει τη δυνατότητα στους παίχτες να επιτεθούν και να αμυνθούν.

ENTITY

Μία **οντότητα** (**Entity**) - abstract class - μπορεί να περιγραφεί από τα εξής χαρακτηριστικά:

- Όνομα (name) – Δε μπορεί να είναι κενό
- Υγεία (health) – Παίρνει τιμές ανάμεσα στο 0 και στο 10 (ποτέ η υγεία δεν είναι μικρότερη του 0)
- Δύναμη (strength) – Παίρνει τιμές ανάμεσα στο 1 και στο 10
- Όπλο (weapon) – Το όπλο το οποίο φέρει η οντότητα

Το συμβόλαιο του ΑΤΔ της **οντότητας** σας δίνεται στη συνέχεια:

Όνομα μεθόδου	Περιγραφή μεθόδου
Entity (String name)	Δημιουργία μίας οντότητας με όνομα name. Όταν δημιουργείται μία οντότητα δεν έχει όπλο και η υγεία της είναι 10. Η δύναμή της καθορίζεται ανάλογα με τον συγκεκριμένο τύπο της οντότητας.*

boolean isAlive()	Επιστρέφει αν η οντότητα είναι ζωντανή ή όχι
void setHealth(int health) void setStrength(int strength) void setWeapon(Weapon weapon)	Ορίζει μία νέα τιμή για την υγεία της οντότητας Ορίζει μία νέα τιμή για τη δύναμη της οντότητας Αναθέτει ένα νέο όπλο στην οντότητα
String getName() int getHealth() int getStrength() Weapon getWeapon()	Επιστρέφει το όνομα της οντότητας Επιστρέφει την υγεία της οντότητας Επιστρέφει τη δύναμη της οντότητας Επιστρέφει το όπλο που φέρει η οντότητα
abstract String toString()	Επιστρέφει την αναπαράσταση της οντότητας ως συμβολοσειρά
boolean wonBattle (Entity adversary)	Επιστρέφει αν η οντότητα νίκησε μάχη εναντίον μίας άλλης οντότητας (adversary) ή όχι.**
boolean isStrongerThan (Entity adversary)	Επιστρέφει αν η οντότητα είναι πιο δυνατή από μια άλλη οντότητα (adversary) ή όχι.

*Οι μάγοι έχουν αρχική δύναμη 4. Τα ξωτικά έχουν αρχική δύναμη 6. Τα όρκες έχουν αρχική δύναμη 3 και τα τρολς έχουν αρχική δύναμη 8.

**Μία οντότητα έχει κερδίσει τη μάχη εναντίον μίας άλλης οντότητας, όταν η άλλη οντότητα είναι νεκρή.

PLAYABLE

Η διεπαφή (interface) **Playable** σας δίνεται παρακάτω:

Όνομα μεθόδου	Περιγραφή μεθόδου
void attack (Entity adversary)	Διατάζει επίθεση εναντίον της οντότητας adversary. Η υγεία του αντιπάλου (adversary) μειώνεται τόσο, όσο είναι η δύναμη της επίθεσης του αντιπάλου. Ο αντίπαλος δεν μπορεί να είναι Playable. Τόσο ο επιτιθέμενος (this) όσο και ο αντίπαλος (adversary) πρέπει να είναι ζωντανοί.
void defend (Entity adversary)	Αμύνεται από την επίθεση της οντότητας adversary. Η υγεία της οντότητας μειώνεται, ανάλογα με τη δύναμη της επίθεσης. Ο adversary δε μπορεί να είναι Playable. Τόσο ο επιτιθέμενος (adversary) όσο και ο δεχόμενος επίθεση (this) πρέπει να είναι ζωντανοί.

Σημείωση: Η διεπαφή (interface) **Playable** υλοποιείται από **Entities**. Αυτό σημαίνει ότι όποια κλάση υλοποιεί την **Playable**, θα πρέπει να υλοποιεί τις μεθόδους και της **Playable** και τις αφηρημένες (abstract) μεθόδους της **Entity**. (πχ class Wizard extends Entity implements Playable)



Ένας **μάγος** μπορεί να περιγραφεί επιπλέον από το χαρακτηριστικό `spellStrength`, που παίρνει τιμές ανάμεσα στο 1 και το 5.

Το συμβόλαιο του ΑΤΔ ενός **μάγου (wizard)** είναι:

Όνομα μεθόδου	Περιγραφή μεθόδου
Wizard (String name)	Δημιουργεί ένα νέο μάγο με όνομα <code>name</code> , αρχική δύναμη 4 και <code>spellStrength</code> έναν τυχαίο αριθμό από 1 ως 5.
void castSpell (Entity adversary)	Ο μάγος επιτίθεται στον <code>adversary</code> με ξόρκι. Η ζωή του αντιπάλου μειώνεται όσο το άθροισμα των <code>spellStrength</code> , της δύναμης του μάγου και της δύναμης του ραβδιού (αν έχει).
void attack (Entity adversary)	Αν ο μάγος έχει ως όπλο του ραβδί (<code>wand</code>)*, τότε καλείται η <code>castSpell()</code> . Διαφορετικά, η δύναμη της επίθεσης είναι το άθροισμα της δύναμης του μάγου και της δύναμης του όπλου του.
void defend (Entity adversary)	Αμύνεται από την επίθεση της οντότητας <code>adversary</code> . Η υγεία του μάγου μειώνεται τόσο, όσο το άθροισμα της δύναμης του <code>adversary</code> και του όπλου του <code>adversary</code> .
int getSpellStrength()	Επιστρέφει την τιμή του <code>spellStrength</code> .

*Ο έλεγχος αυτός μπορεί να γίνει απλά επιστρέφοντας `true` αν το όνομα του όπλου είναι "Wand". Ο έλεγχος αυτός μπορεί να γίνεται σε μία νέα μέθοδο `boolean hasWand()`.



Ένα **ξωτικό** χαρακτηρίζεται επίσης από το αν είναι ορατό ή αόρατο από τη boolean μεταβλητή `visible`.

Το συμβόλαιο του ΑΤΔ ενός **ξωτικού (elf)** είναι το εξής:

Όνομα μεθόδου	Περιγραφή μεθόδου
Elf (String name)	Δημιουργεί ένα νέο ξωτικό με όνομα <code>name</code> , που είναι ορατό και έχει αρχική δύναμη 6
void becomeVisible () void becomeInvisible ()	Το ξωτικό γίνεται ορατό. Το ξωτικό γίνεται αόρατο.
boolean isVisible ()	Επιστρέφει αν το ξωτικό είναι ορατό ή όχι.
void attack (Entity adversary)	Η ζημιά που επιφέρει στην υγεία του <code>adversary</code> είναι όση το άθροισμα της δύναμης του ξωτικού και της δύναμης του όπλου του (αν έχει).
void defend (Entity adversary)	Αν το ξωτικό είναι ορατό, τότε η υγεία του μειώνεται τόσο, όσο το άθροισμα της δύναμης του <code>adversary</code> και του όπλου του <code>adversary</code> . Αν το ξωτικό είναι αόρατο, τότε η υγεία του δε μειώνεται καθόλου. Το

	ξωτικό δε μπορεί να είναι άορατο σε δύο συνεχόμενες επιθέσεις κάποιου αντιπάλου.*
--	---

***Βοήθεια:** μπορείτε να καλέσετε τις `becomeVisible` και `becomeInvisible` στην αρχή της μεθόδου `defend()`, τηρώντας όμως τον περιορισμό.



Τα **τρολς** δεν έχουν κάποια επιπλέον χαρακτηριστικά. Θυμηθείτε ότι στον constructor του `Troll` πρέπει να αρχικοποιηθεί και η δύναμή του (8). Επίσης μην ξεχάσετε ότι και τα `Troll` είναι `Entity`, άρα πρέπει να υλοποιήσουν την abstract μέθοδο `String toString()`.



Τα **ορκς** έχουν ένα επιπλέον χαρακτηριστικό, το `Troll leader`, που αναφέρεται σε κάποιο `Troll`. Το συμβόλαιο ΑΤΔ ενός **ορκ (orc)** είναι το εξής:

Όνομα μεθόδου	Περιγραφή μεθόδου
<code>Orc (String name)</code>	Δημιουργεί ένα νέο <code>orc</code> με όνομα <code>name</code> , χωρίς ηγέτη, με δύναμη 3.
<code>Orc (String name, Troll leader)</code>	Δημιουργεί ένα νέο <code>orc</code> με όνομα <code>name</code> , ηγέτη τον <code>leader</code> και δύναμη 3.
<code>void setLeader (Troll leader)</code> <code>Troll getLeader ()</code>	Ο <code>leader</code> ορίζεται ως ηγέτης του <code>ορκ</code> . Επιστρέφει τον ηγέτη του <code>ορκ</code> .



Ένα **όπλο** (`weapon`) μπορεί να περιγραφεί από τα εξής χαρακτηριστικά:

- Όνομα (`name`) – Δε μπορεί να είναι κενό
- Δύναμη (`strength`) – Παίρνει τιμές ανάμεσα στο 1 και στο 3

Το συμβόλαιο ΑΤΔ ενός **όπλου (weapon)** είναι το εξής:

Όνομα μεθόδου	Περιγραφή μεθόδου
<code>Weapon (String name)</code> <code>Weapon (String name, int)</code>	Δημιουργεί ένα νέο όπλο με όνομα <code>name</code> και δύναμη μία τυχαία τιμή από 1 έως 3.

strength)	Δημιουργεί ένα νέο όπλο με όνομα name και δύναμη strength.
int getStrength () String getName ()	Επιστρέφει τη δύναμη του όπλου. Επιστρέφει το όνομα του όπλου.

Βοήθεια: Πριν καλέσετε τις μεθόδους getStrength() και getName() βεβαιωθείτε ότι η οντότητα που εξετάζετε έχει κάποιο όπλο. π.χ. if (this.getWeapon() != null) { ... }.

Παρακάτω ακολουθεί ο κώδικας μια κλάσης Battle της οποίας η main() καλεί την createPlayers() η οποία δημιουργεί στιγμιότυπα των κλάσεων που θα έχετε ορίσει. Κατόπιν δίνει ένα παράδειγμα μάχης μεταξύ δύο οντοτήτων.

Μπορείτε να χρησιμοποιήσετε την κλάση αυτή για να ελέγξετε κάποιες από τις λειτουργίες που υλοποιήσατε. Πρέπει να τρέχει και με τη δική σας υλοποίηση και να δίνει την έξοδο που δίδεται πιο κάτω.

```
public class Battle {
    private static List<Entity> createPlayers() {
        List<Entity> players = new ArrayList<>();

        Wizard wiz = new Wizard("Pallando");
        players.add(wiz);

        Elf elf = new Elf("Orodreth");
        Weapon sword = new Weapon("sword");
        elf.setWeapon(sword);
        players.add(elf);
        Troll troll = new Troll("Rogash");
        players.add(troll);

        Orc orc = new Orc("Bolg");
        Weapon sword2 = new Weapon("sword");
        orc.setWeapon(sword2);
        orc.setLeader(troll);
        players.add(orc);

        Entity wiz2 = new Wizard("Alatar");
        Weapon wand = new Weapon("wand");
        wiz2.setWeapon(wand);
        players.add(wiz2);
        return players;
    }

    public static void main (String[] args){
        List<Entity> players = createPlayers();
        System.out.println("Created the following players:\n");
        for (Entity entity : players) {
            System.out.print(entity.toString());
        }

        Wizard e0 = (Wizard) players.get(0);
        Elf e1 = (Elf) players.get(1);

        if (e0.isStrongerThan(e1)) {
            System.out.println(e0.getName()+" is stronger than "+e1.getName());
        }
    }
}
```

```

    } else {
        System.out.println(e0.getName()+" is not stronger than "+e1.getName());
    }

    Orc e3 = (Orc) players.get(3);
    Wizard e4 = (Wizard) players.get(4);
    Troll leader = e3.getLeader();

    System.out.println("\nBattle between "+e4.getName()+" and "+e3.getName()+"!");
    while (!e4.wonBattle(e3) && !e3.wonBattle(e4)) {
        System.out.print(e4.getName()+"'s health:" +e4.getHealth()+" ");
        System.out.println(e3.getName()+"'s health:" +e3.getHealth());
        try {
            e4.attack(e3);
            System.out.println(e4.getName()+" just attacked "+e3.getName());
            System.out.println(e3.getName()+"'s health:" +e3.getHealth());
            e4.defend(e3);
            System.out.println(e3.getName()+" just attacked "+e4.getName());
            System.out.println(e4.getName()+"'s health:" +e4.getHealth());

            if (leader != null) {
                e4.defend(leader);
                System.out.println(leader.getName()+" just attacked "+e4.getName());
                System.out.println(e4.getName()+"'s health:" +e4.getHealth());
                e1.defend(leader);
                System.out.println(leader.getName()+" just attacked "+e1.getName());
                if (e1.isVisible()) {
                    System.out.println(e1.getName() + " was visible!");
                } else {
                    System.out.println(e1.getName() + " was invisible!");
                }
                System.out.println(e1.getName()+"'s health:" +e1.getHealth());
            }
        } catch (Exception ex) {
            System.err.println(ex.toString());
        }
    }
    if (e4.wonBattle(e3)) {
        System.out.println(e4.getName() + " won the battle against "+e3.getName());
    } else {
        System.out.println(e4.getName() + " lost the battle against "+e3.getName());
    }
}
}

```

Ενδεικτική έξοδος:

Created the following players:

Wizard: Pallando

Spell strength:4

Health: 10

Strength: 4

No weapon

Elf: Orodreth

Visible:true

Health: 10

Strength: 6

Weapon: name: sword, strength: 1

Troll: Rogash

*Health: 10
Strength: 8
No weapon*


*Orc: Bolg
Troll leader: Rogash
Health: 10
Strength: 3
Weapon: name: sword, strength: 2*

*Wizard: Alatar
Spell strength: 1
Health: 10
Strength: 4
Weapon: name: wand, strength: 3*

Pallando is not stronger than Orodreth

*Battle between Alatar and Bolg!
Alatar's health: 10, Bolg's health: 10
Alatar just attacked Bolg
Bolg's health: 2
Bolg just attacked Alatar
Alatar's health: 5
Rogash just attacked Alatar
Alatar's health: 0
Rogash just attacked Orodreth
Orodreth was invisible!
Orodreth's health: 10
Alatar lost the battle against Bolg*

Παραδοτέα: Entity.java, Playable.java, Wizard.java, Elf.java, Orc.java, Troll.java, Weapon.java

	Άσκηση 3. Person again Αξία: 15% Τεκμηρίωση και έλεγχος ADTs (Javadoc, Junit). <i>Η διαφορά με τις προηγούμενες άσκησης είναι ότι εδώ θα χρησιμοποιήσετε και JavaDoc και JUNIT tests</i>
---	---

Κάντε ένα αντίγραφο της κλάσης **Person** που κάνατε στην 1^η άσκηση που είχε τα ιδιωτικά μέλη **όνομα** (String), **επώνυμο** (String) και **αριθμό ταυτότητας** (String). Προσθέστε ένα στατικό πεδίο **numberOfPersons** το οποίο πρέπει να κρατά τον αριθμό των προσώπων που έχουν δημιουργηθεί.

Ορίστε ένα πλήρες σύνολο μεθόδων (**constructors, accessors, transformers**) λαμβάνοντας υπόψη σας και τα εξής:


- τα ονόματα να έχουν μόνο έγκυρους (για ονόματα ανθρώπων) χαρακτήρες
- ο αριθμός ταυτότητας να έχει πάντα 8 χαρακτήρες: 2 γράμματα στην αρχή και κατόπιν έξι ψηφία.
- ο αριθμός numberOfPersons πρέπει να έχει την ορθή τιμή.

Σε περίπτωση που κληθεί μία δημόσια μέθοδος με παράμετρο που δεν πληροί τους παραπάνω κανόνες τότε ο κώδικας σας να εγείρει ένα `IllegalArgumentException`.

(i) [5 μονάδες] Δώστε την υλοποίηση σε Java

(ii) [5 μονάδες] Τεκμηριώστε πλήρως την υλοποίηση σας με χρήση Javadoc. Πλοηγηθείτε με έναν ιστοπλοηγητή για να δείτε η παραχθείσα τεκμηρίωση είναι εντάξει.

(iii) [5 μονάδες] Δημιουργείστε JUnit tests που να ελέγχουν ότι η υλοποίηση σας είναι σωστή.

	<p>Άσκηση 4 – Βιβλιοθήκη Αξία: 30%</p> <p>[ADTs, overloading, pre/postconditions-invariants, Javadocs, JUnit Tests]</p> <p><i>Η άσκηση αυτή σε σχέση με τις προηγούμενες επικεντρώνεται στην τεκμηρίωση και στα συμβόλαια.</i></p>
---	---

Σε αυτή την άσκηση καλείστε να υλοποιήσετε ένα πολύ απλό σύστημα διαχείρισης βιβλιοθήκης. Θα χρειαστεί να υλοποιήσετε τέσσερεις Αφαιρετικούς Τύπους Δεδομένων (ΑΤΔ):

1. **Βιβλίο** (Book),
2. **Αντίτυπο** (Copy),
3. **Βιβλιοθήκη** (Library) και
4. **Μέλος** (Member).

οι οποίοι προφανώς συνδέονται μεταξύ τους (ένα βιβλίο έχει αντίτυπα τα οποία μία βιβλιοθήκη δανείζει στα μέλη της), και θα περιγραφούν αναλυτικά παρακάτω.

Επίσης καλείστε να:

- (a) τεκμηριώσετε πλήρως τον κώδικά σας με τη βοήθεια των **Javadocs**
- (b) υλοποιήσετε τις αμετάβλητες συνθήκες (invariants) των κλάσεων που θα υλοποιήσετε και τις εκ των προτέρων και εκ των υστέρων συνθήκες (**preconditions**, **postconditions**) των μεθόδων τους. Οι συνθήκες αυτές θα πρέπει να ορίζονται σαφώς στην τεκμηρίωση μέσω Javadocs για κάθε μέθοδο, καθώς επίσης τα preconditions θα πρέπει να ελέγχονται με χρήση **exceptions**.
- (c) να χρησιμοποιήσετε μηχανισμούς ελέγχου ορθότητας του κώδικά σας, μέσω **JUnit Tests** για τις μεθόδους των ΑΤΔ Book και Copy.

Ακολουθούν οι ιδιότητες και τα χαρακτηριστικά των ΑΤΔ που καλείστε να σχεδιάσετε και να υλοποιήσετε:

ΑΤΔ Book

Χαρακτηριστικά:

Τίτλος (title) – απαραίτητο

Κωδικός (code) – απαραίτητος, είναι μοναδικός για κάθε βιβλίο

Συγγραφείς (authors) – απαραίτητο, μπορεί να είναι παραπάνω από ένας

Εκδότης (publisher)

Έτος (year)

Κατηγορία (category) – παίρνει τιμές από ένα προκαθορισμένο σύνολο τιμών που ορίζεται παρακάτω (enum Category*)

Αντίτυπα (copies) – το σύνολο των κωδικών αντιτύπων του συγκεκριμένου τίτλου που υπάρχουν στη βιβλιοθήκη

Συμβόλαιο:

Όνομα Μεθόδου	Περιγραφή Μεθόδου
<code>Book(String title, int code, List<String> authors)</code> <code>Book(String title, int code, List<String> authors, String publisher)</code> <code>Book(String title, int code, List<String> authors, String publisher, int year)</code> <code>Book(String title, int code, List<String> authors, String publisher, int year, Category categ)</code>	Δημιουργία ενός βιβλίου με τίτλο <code>title</code> , κωδικό <code>code</code> και συγγραφείς <code>authors</code> . Τα υπόλοιπα χαρακτηριστικά παραμένουν κενά. Όπως παραπάνω, ορίζοντας επιπλέον και τον εκδότη <code>publisher</code> . Όπως παραπάνω, ορίζοντας επιπλέον και το έτος έκδοσης <code>year</code> . Όπως παραπάνω, ορίζοντας επιπλέον και την κατηγορία <code>categ</code> .
<code>String getTitle()</code> <code>int getCode()</code> <code>List<String> getAuthors()</code> <code>String getPublisher()</code> <code>int getYear()</code> <code>Category getCategory()</code> <code>List<Integer> getCopies()</code>	Επιστρέφει τον τίτλο του βιβλίου Επιστρέφει τον κωδικό Επιστρέφει τους συγγραφείς Επιστρέφει τον εκδότη Επιστρέφει το έτος έκδοσης Επιστρέφει την κατηγορία Επιστρέφει τη λίστα με τους κωδικούς αντιτύπων του βιβλίου
<code>void setTitle(String title)</code> <code>void setCode(int code)</code> <code>void setAuthors(List<String> authors)</code> <code>void setPublisher(String publisher)</code> <code>void setYear(int year)</code> <code>void setCategory(Category categ)</code> <code>void setCopies(List<Integer> copies)</code>	Ορίζει νέο τίτλο για το βιβλίο Ορίζει νέο κωδικό Ορίζει νέους συγγραφείς Ορίζει νέο εκδότη Ορίζει νέο έτος έκδοσης Ορίζει νέα κατηγορία Ορίζει νέα λίστα κωδικών αντιτύπων
<code>void addCopy(int copyCode)</code>	Προσθέτει έναν κωδικό αντιτύπου για το συγκεκριμένο βιβλίο
<code>void removeCopy(int copyCode)</code>	Αφαιρεί έναν κωδικό αντιτύπου
<code>void removeAllCopies()</code>	Αφαιρεί όλους τους κωδικούς από τη λίστα κωδικών αντιτύπων

* **protected enum Category { Action, Fiction, Drama, Romance, SciFi, Poems, Kids }**

ΑΤΔ Copy

Χαρακτηριστικά:

Κωδικός (`code`) – απαραίτητος, πρέπει να είναι μοναδικός για κάθε αντίτυπο

Κωδικός βιβλίου (`bookCode`) – απαραίτητος, είναι ο κωδικός του βιβλίου στο οποίο ανήκει το αντίτυπο

Σε δανεισμό (checkedOut) – περιγράφει αν είναι σε δανεισμό ή όχι
Κωδικός τελευταίου μέλους που το δανείστηκε (lastMember)

Συμβόλαιο:

Όνομα Μεθόδου	Περιγραφή Μεθόδου
Copy(int copyCode, int bookCode)	Δημιουργία ενός αντιτύπου με κωδικό copyCode. Με τη δημιουργία του, το αντίτυπο δεν είναι σε δανεισμό και δεν υπάρχει τελευταίο μέλος που το δανείστηκε.
Int getCode() int getBookCode() boolean getCheckedOut() int getLastMember()	Επιστρέφει τον κωδικό του αντιτύπου Επιστρέφει τον κωδικό βιβλίου του αντιτύπου Επιστρέφει αν το αντίτυπο είναι δανεισμένο ή όχι Επιστρέφει τον κωδικό του μέλους που δανείστηκε τελευταίο το αντίτυπο
void setCode(int copyCode) void setBookCode(int bookCode) void setCheckedOut(boolean chk) void setLastMember(int memberID)	Ορίζει νέο κωδικό για το αντίτυπο Ορίζει νέο κωδικό βιβλίου για το αντίτυπο Ορίζει αν είναι σε δανεισμό ή όχι το αντίτυπο Ορίζει τον κωδικό του τελευταίου μέλους που δανείστηκε το αντίτυπο

ΑΤΔ Member

Χαρακτηριστικά:

Όνομα (firstName) – απαραίτητο

Επώνυμο (lastName) – απαραίτητο

Κωδικός μέλους (memberID) – απαραίτητος, μοναδικό για κάθε μέλος

Αντίτυπα σε δανεισμό (checkedCopies) – το σύνολο των κωδικών των αντιτύπων που έχει σε δανεισμό το μέλος κάθε δεδομένη στιγμή

Ιστορικό δανεισμού (historyCopies) – το σύνολο των κωδικών όλων των αντιτύπων που έχει δανειστεί στο παρελθόν το μέλος (μαζί με τα τρέχοντα αντίτυπα)

Συμβόλαιο:

Όνομα Μεθόδου	Περιγραφή Μεθόδου
Member(String firstName, String lastName, int memberID)	Δημιουργία ενός μέλους με όνομα firstName, επώνυμο lastName και κωδικό μέλους memberID. Με τη δημιουργία του, το μέλος δεν έχει αντίτυπα σε δανεισμό, ούτε έχει δανειστεί αντίτυπα στο παρελθόν.
String getFirstName()	Επιστρέφει το όνομα του μέλους

String getLastName() int getMemberID() List<Integer> getCheckedCopies() List<Integer> getHistoryCopies()	Επιστρέφει το επώνυμο του μέλους Επιστρέφει τον κωδικό μέλους Επιστρέφει τα αντίτυπα σε δανεισμό του μέλους Επιστρέφει το ιστορικό δανεισμού
void setFirstName(String firstName) void setLastName(String firstName) void setMemberID(int memberID) void setCheckedCopies(List<Integer> checkedCopies) void setHistoryCopies(List<Integer> historyCopies)	Ορίζει νέο όνομα για το μέλος Ορίζει νέο επώνυμο για το μέλος Ορίζει νέο κωδικό μέλους Ορίζει νέο σύνολο για τα αντίτυπα σε δανεισμό Ορίζει νέο ιστορικό δανεισμού
void addCopyToCheckedCopies(int copyCode) void addCopyToHistoryCopies(int copyCode) void removeCopyFromCheckedCopies(int copyCode) void removeCopyFromHistoryCopies(int copyCode)	Προσθέτει τον κωδικό ενός αντιτύπου στο σύνολο με τα αντίτυπα υπό δανεισμό Προσθέτει τον κωδικό ενός αντιτύπου στο ιστορικό δανεισμού Αφαιρεί τον κωδικό ενός αντιτύπου από τα υπό δανεισμό αντίτυπα Αφαιρεί τον κωδικό ενός αντιτύπου από το ιστορικό δανεισμού

ΑΤΔ Library

Χαρακτηριστικά:

Όνομα (name) – απαραίτητο

Τίτλοι βιβλίων (books) – το σύνολο των βιβλίων που υπάρχει στη βιβλιοθήκη

Αντίτυπα (copies) – το σύνολο των αντιτύπων που υπάρχει στη βιβλιοθήκη

Μέλη (members) – το σύνολο των μελών της βιβλιοθήκης

Παρατήρηση: Η βιβλιοθήκη θα πρέπει να δημιουργεί, να διαχειρίζεται και να διατηρεί τη μοναδικότητα των κωδικών βιβλίου, αντιτύπου και μέλους.

Συμβόλαιο:

Όνομα Μεθόδου	Περιγραφή Μεθόδου
Library(String name)	Δημιουργία μίας βιβλιοθήκης με όνομα name. Με τη δημιουργία της, η βιβλιοθήκη δεν έχει τίτλους βιβλίων ούτε μέλη.
String getName() List<Book> getBooks() List<Book> getCopies() List<Member> getMembers()	Επιστρέφει το όνομα της βιβλιοθήκης Επιστρέφει το σύνολο των βιβλίων της βιβλιοθήκης Επιστρέφει το σύνολο των αντιτύπων της βιβλιοθήκης Επιστρέφει το σύνολο των μελών της βιβλιοθήκης
void setName(String name) void setBooks(List<Book> books)	Ορίζει νέο όνομα για τη βιβλιοθήκη Ορίζει νέο σύνολο βιβλίων της βιβλιοθήκης

void setCopies(List<Copy> copies) void setMembers(List<Member> members)	Ορίζει νέο σύνολο αντιτύπων της βιβλιοθήκης Ορίζει νέο σύνολο μελών της βιβλιοθήκης
int addBook(String title, List<String> authors) int addBook(String title, List<String> authors, String publisher) int addBook(String title, List<String> authors, String publisher, int year) int addBook(String title, List<String> authors, String publisher, int year, Category categ)	Προσθήκη ενός βιβλίου με τίτλο title και συγγραφείς authors. Τα υπόλοιπα χαρακτηριστικά παραμένουν κενά. Επιστρέφει τον κωδικό του βιβλίου το οποίο πρόσθεσε. Όπως παραπάνω, ορίζοντας επιπλέον και τον εκδότη publisher. Όπως παραπάνω, ορίζοντας επιπλέον και το έτος έκδοσης year. Όπως παραπάνω, ορίζοντας επιπλέον και την κατηγορία categ.
Book removeBook(int bookCode)	Αφαιρεί το βιβλίο με κωδικό bookCode από τη βιβλιοθήκη, μαζί με όλα τα αντίτυπά του. Επιστρέφει το βιβλίο το οποίο αφαίρεσε.
int addCopy(int bookCode)	Προσθέτει ένα αντίτυπο για το βιβλίο με κωδικό bookCode. Επιστρέφει τον κωδικό αντιτύπου το οποίο πρόσθεσε.
Copy removeCopy(int copyCode)	Αφαιρεί το αντίτυπο με κωδικό copyCode από τη βιβλιοθήκη. Επιστρέφει το αντίτυπο το οποίο αφαίρεσε.
int addMember(String firstName, String lastName)	Προσθέτει ένα μέλος με όνομα firstName και επώνυμο lastName. Επιστρέφει τον κωδικό του νέου μέλους.
Member removeMember(int memberID)	Αφαιρεί το μέλος με κωδικό μέλους memberID από τη βιβλιοθήκη. Επιστρέφει το μέλος το οποίο αφαίρεσε.
Book findBook(String title, List<String> authors) Book findBook(String title, List<String> authors, String publisher) Book findBook(String title, List<String> authors, String publisher, int year) Book findBook(String title, List<String> authors, String publisher, int year, Category categ)	Επιστρέφει το βιβλίο με τίτλο title και συγγραφείς authors. Όπως παραπάνω, ψάχνοντας επιπλέον και τον εκδότη publisher. Όπως παραπάνω, ψάχνοντας επιπλέον και το έτος έκδοσης year. Όπως παραπάνω, ψάχνοντας επιπλέον και την κατηγορία categ.
int rentBook(int memberID, String bookTitle, List<String> bookAuthors)	Δανείζει στο μέλος με κωδικό μέλους memberID ένα διαθέσιμο, αν υπάρχει, αντίτυπο του βιβλίου με τίτλο bookTitle και συγγραφείς bookAuthors. Επιστρέφει τον κωδικό αντιτύπου που δανείστηκε, αν ήταν επιτυχής ο δανεισμός.
void returnCopy(int copyCode)	Επιστρέφει στη βιβλιοθήκη το αντίτυπο με κωδικό

	copyCode.
int getMembersNum()	Επιστρέφει τον αριθμό μελών της βιβλιοθήκης
int getBooksNum()	Επιστρέφει τον αριθμό των διαφορετικών τίτλων της βιβλιοθήκης
int getCopiesNum()	Επιστρέφει τον αριθμό των διαφορετικών αντιτύπων της βιβλιοθήκης
int getCheckedCopiesNum()	Επιστρέφει τον αριθμό των αντιτύπων που είναι σε δανεισμό
String toString()	Επιστρέφει μία συμβολοσειρά με το όνομα και τα στατιστικά της βιβλιοθήκης (αριθμός μελών, τίτλων, αντιτύπων και αντιτύπων σε δανεισμό)

Παραδοτέα: Book.java, BookTest.java (JUnit test), Copy.java, CopyTest.java (JUnit test), Member.java, Library.java

ΟΔΗΓΙΕΣ ΠΑΡΑΔΟΣΗΣ

Φτιάξτε ένα φάκελο με όνομα A2_<<ΑριθμόςΜητρώου>>. Μέσα του δημιουργήστε έναν υποφάκελο για κάθε άσκηση (A2.1, A2.2, ...). Συμπιεστέ το φάκελο με όνομα A1_<<ΑριθμόςΜητρώου>>(Προσοχή όσες ασκήσεις παραδοθούν με διαφορετικό όνομα δεν θα γίνονται δεκτές), και παραδώστε τον μέσω του moodle

Απορίες σχετικά με την A1 θα απαντώνται μόνο μέσω του forum της A2 στο moodle.

Καλή εργασία