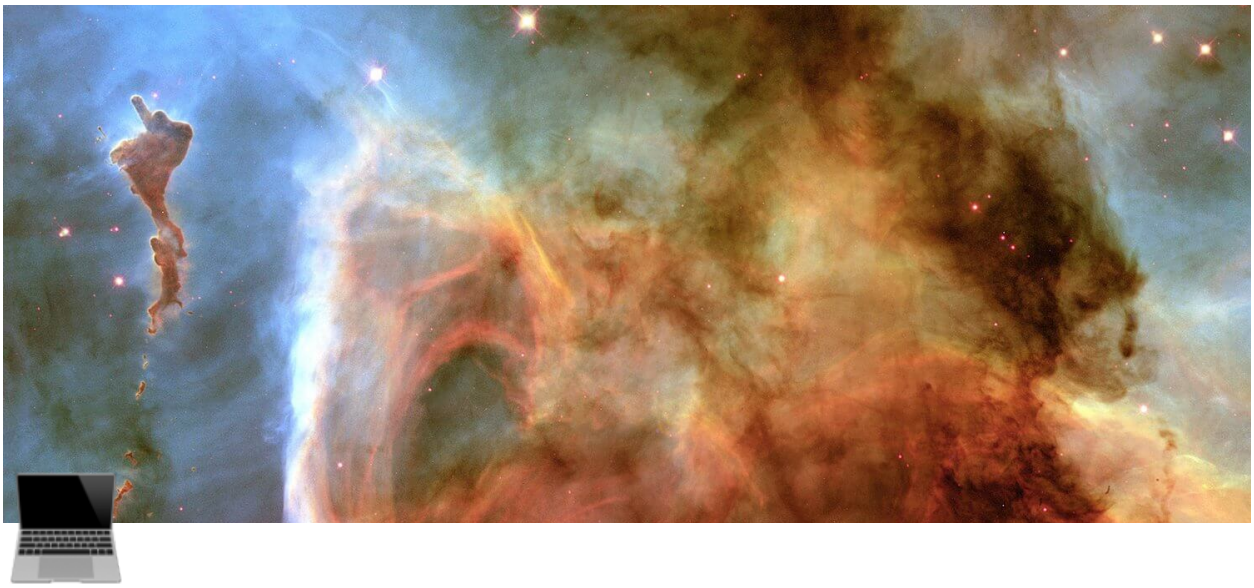


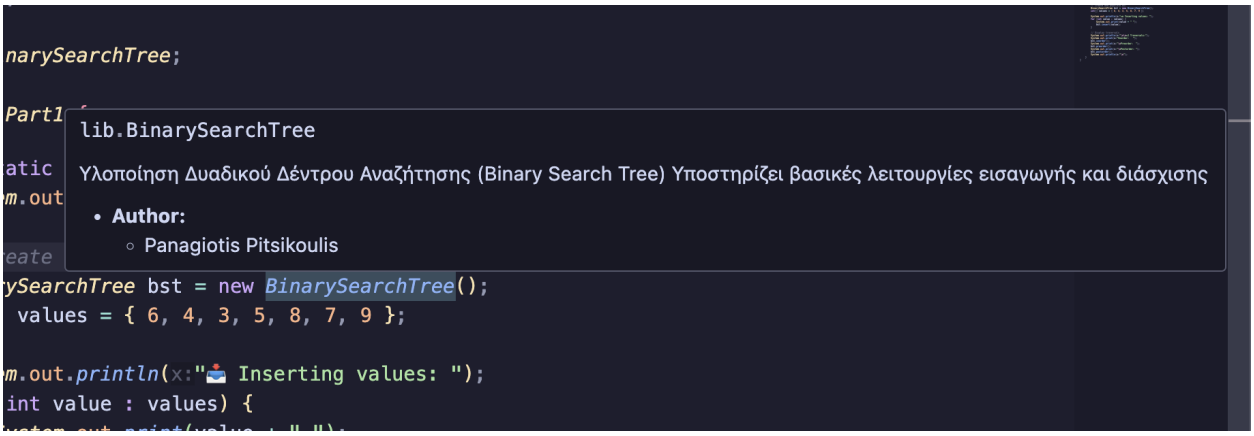
CN5005 Data Structures and Algorithms

Γενική Αρχιτεκτονική

Γίνεται χρήση αντικειμενοστρεφούς προγραμματισμού, με αυστηρή ιεραρχία. Το αρχείο ROOT του προγράμματος είναι το App.java, που συνδέεται με:

- Το part1
- Το part2
- Τα unit test
- Τις βιβλιοθήκες

Ταυτόχρονα, έχουν γραφτεί σχόλια υλοποίησης μεθόδων για κάθε μέθοδο με in-editor preview για κάθε μέθοδο:




```
    /**
     * Ελέγχει αν το person1 είναι παιδί του person2
     */
    private boolean isChild(String person1, String person2) {
        return parents.containsKey(person1) && parents.get(pers
    }

    /**
     * Ελέγχει αν τα δύο άτομα είναι αδέρφια
     * Αδέρφια θεωρούνται όταν έχουν τουλάχιστον έναν κοινό γον
     */
    private boolean isSibling(String person1, String person2) {
        if (!parents.containsKey(person1) || !parents.containsKe
            return false;
        return !Collections.disjoint(parents.get(person1), pare
    }

    /**
     * Ελέγχει αν τα δύο άτομα είναι σύζυγοι
     */
    private boolean isSpouse(String person1, String person2) {
        return spouses.containsKey(person1) && spouses.get(pers
    }

    /**
```

```
     * Ελέγχει αν το person1 είναι παππούς/γιαγιά του person2
     */
    private boolean isGrandparent(String person1, String person2
        if (!children.containsKey(person1))
            return false;
        for (String child : children.get(person1)) {
            if (isParent(child, person2))
                return true;
        }
        return false;
    }
```

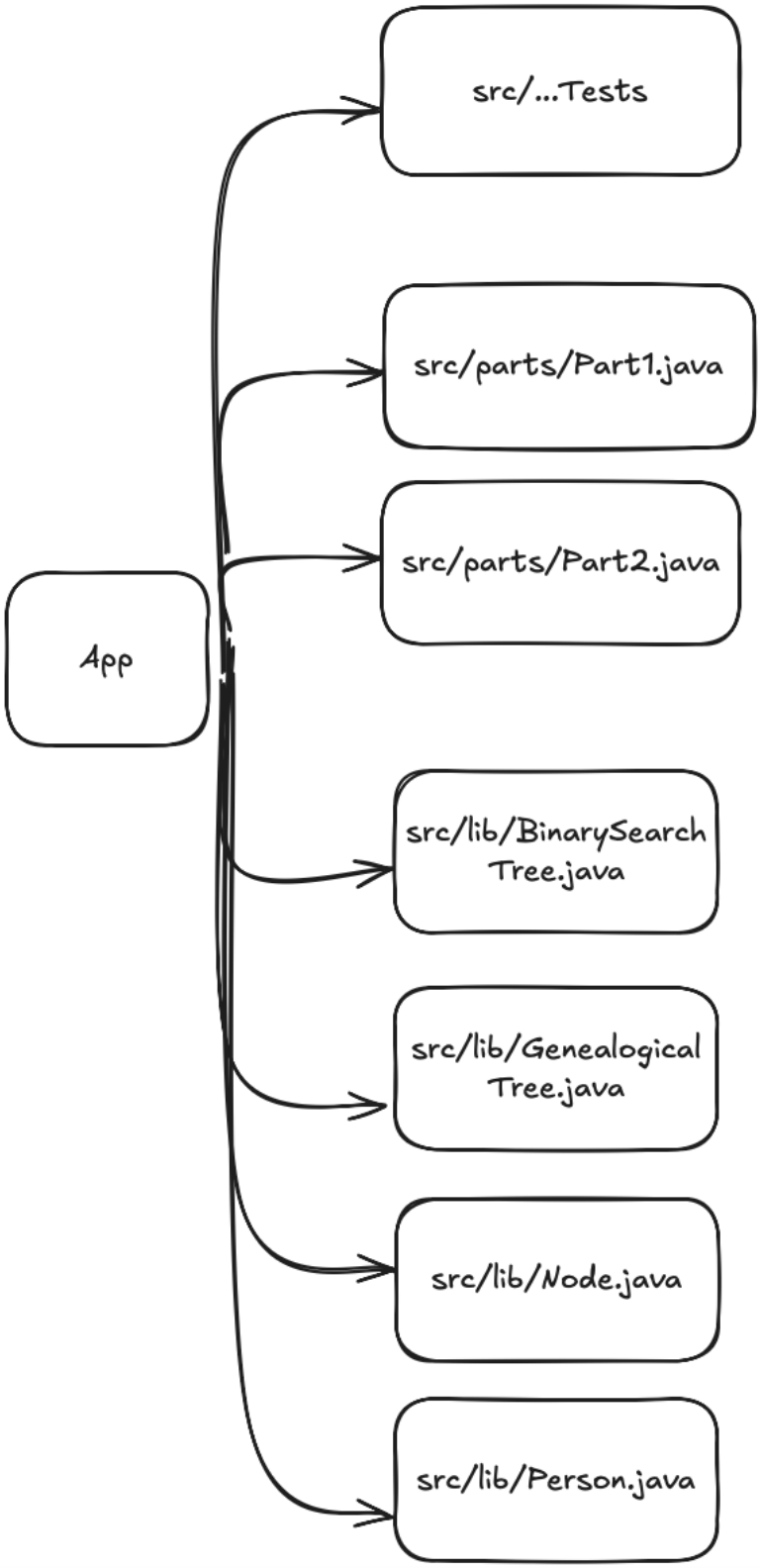
Επιπλέον, το root της εφαρμογής παρέχει μια CLI Interface με επιλογές για την εκτέλεση του προγράμματος, βελτιώνοντας την εμπειρία χρήστη.

Τέλος, έχουν συνταχθεί αρχεία markdown με οδηγείες χρήσης και σημειώσεις για κάθε directory.

Ο κώδικας υλοποίησης των data structures έχει τοποθετηθεί στον φάκελο lib σε υποκλάσεις, έτσι επιτυγχάνεται η αρχή encapsulation του OOP. Για παράδειγμα:

```
package lib;

/**
 * Κλάση που αναπαριστά ένα άτομο στο γενεαλογικό δέντρο
 *
 * @author Panagiotis Pitsikoulis
 */
public class Person {
    /** Το όνομα του ατόμου */
    private String name;
    /** Το φύλο του ατόμου (man/woman) */
    private String gender;

    /**
     * Δημιουργεί ένα νέο άτομο
     *
```

```
     * @param name   το όνομα του ατόμου
     * @param gender το φύλο του ατόμου
     */
    public Person(String name, String gender) {
        this.name = name;
        this.gender = gender;
    }

    /**
     * Επιστρέφει το όνομα του ατόμου
     *
     * @return το όνομα
     */
    public String getName() {
        return name;
    }

    /**
     * Επιστρέφει το φύλο του ατόμου
     *
     * @return το φύλο
     */
    public String getGender() {
        return gender;
    }
}
```


src/...Tests
src/parts/Part1.java
src/parts/Part2.java
App
src/lib/BinarySearch Tree.java
src/lib/Genealogical Tree.java
src/lib/Node.java
src/lib/Person.java

Διάγραμμα αρχιτεκτονικής


├── CREDITS.md
├── README.md
├── bin
│   ├── App.class
│   ├── lib
│   │   ├── BinarySearchTree.class
│   │   ├── GenealogicalTree.class
│   │   ├── Node.class
│   │   ├── Person.class
│   │   └── README.md
│   ├── parts
│   │   ├── Part1.class
│   │   ├── Part2.class
│   │   └── README.md
│   ├── public
│   │   ├── README.md
│   │   ├── input.csv
│   │   └── test_family.csv
│   └── tests
│   ├── BinarySearchTreeTest.class
│   ├── GenealogicalTreeTest.class
│   ├── NodeTest.class
│   ├── PersonTest.class
│   ├── README.md
│   └── TestRunner.class
├── lib
│   ├── hamcrest-core-1.3.jar
│   └── junit-4.13.2.jar
├── public
│   ├── input.csv
│   └── test_family.csv
└── src
├── App.java
├── lib
│   ├── BinarySearchTree.java
│   ├── GenealogicalTree.java
│   ├── Node.java
│   ├── Person.java
│   └── README.md
├── parts
│   ├── Part1.java
│   ├── Part2.java
│   └── README.md
├── public
│   ├── README.md
│   ├── input.csv
│   └── test_family.csv
└── tests
├── BinarySearchTreeTest.java
├── GenealogicalTreeTest.java
├── NodeTest.java
├── PersonTest.java
├── README.md
└── TestRunner.java

Binary Tree

Το δέντρο έχει κατασκευαστεί πλήρως και σωστά με όλους τους κόμβους στις σωστές θέσεις σύμφωνα με τους κανόνες του δυαδικού δέντρου αναζήτησης.

```
package parts;

import lib.BinarySearchTree;

public class Part1 {
    public static void main(String[] args) {
        System.out.println("\n🌳 Binary Search Tree Demo\n");

        // Create and populate BST
        BinarySearchTree bst = new BinarySearchTree();
        int[] values = { 6, 4, 3, 5, 8, 7, 9 };

        System.out.println("📥 Inserting values: ");
        for (int value : values) {
            System.out.print(value + " ");
            bst.insert(value);
        }

        // Display traversals
        System.out.println("\n\n📊 Traversals:");
        System.out.print("Inorder:   ");
        bst.inorder();
        System.out.print("\nPreorder:  ");
        bst.preorder();
        System.out.print("\nPostorder: ");
        bst.postorder();
        System.out.println("\n");
```

```
    }
}
```

1. **Προδιατεταγμένη, Ενδοδιατεταγμένη και Μεταδιατεταγμένη Αναπαράσταση**

   Όλες οι αναπαραστάσεις (preorder, inorder, postorder) υλοποιούνται σωστά και παρουσιάζονται όπως αναμένεται.

```
package lib;

/**
 * Υλοποίηση Δυαδικού Δέντρου Αναζήτησης (Binary Search Tree)
 * Υποστηρίζει βασικές λειτουργίες εισαγωγής και διάσχισης
 *
 * @author Panagiotis Pitsikoulis
 */
public class BinarySearchTree {
    /** Η ρίζα του δέντρου */
    private Node root;

    /**
     * Εισάγει μια νέα τιμή στο δέντρο διατηρώντας την ιδιότη
     *
     * @param value η τιμή που θα εισαχθεί στο δέντρο
     */
    public void insert(int value) {
        root = insertRec(root, value);
    }

    /**
     * Αναδρομική βοηθητική μέθοδος για την εισαγωγή στοιχείο
     *
     * @param root  η τρέχουσα ρίζα του υποδέντρου
     * @param value η τιμή προς εισαγωγή
     * @return ο νέος κόμβος ή το ενημερωμένο υποδέντρο
```

```
     */
    private Node insertRec(Node root, int value) {
        if (root == null) {
            root = new Node(value);
            return root;
        }
        if (value < root.value) {
            root.left = insertRec(root.left, value);
        } else if (value > root.value) {
            root.right = insertRec(root.right, value);
        }
        return root;
    }

    /**
     * Εκτελεί ενδοδιατεταγμένη διάσχιση (inorder traversal)
     * Επισκέπτεται τους κόμβους με σειρά: αριστερό παιδί ->
     */
    public void inorder() {
        inorderRec(root);
    }

    /**
     * Αναδρομική βοηθητική μέθοδος για την ενδοδιατεταγμένη
     *
     * @param root η ρίζα του τρέχοντος υποδέντρου
     */
    private void inorderRec(Node root) {
        if (root != null) {
            inorderRec(root.left);
            System.out.print(root.value + " ");
            inorderRec(root.right);
        }
    }

    /**
```

```
     * Εκτελεί προδιατεταγμένη διάσχιση (preorder traversal)
     * Επισκέπτεται τους κόμβους με σειρά: ρίζα -> αριστερό π
     */
    public void preorder() {
        preorderRec(root);
    }

    /**
     * Αναδρομική βοηθητική μέθοδος για την προδιατεταγμένη δ
     *
     * @param root η ρίζα του τρέχοντος υποδέντρου
     */
    private void preorderRec(Node root) {
        if (root != null) {
            System.out.print(root.value + " ");
            preorderRec(root.left);
            preorderRec(root.right);
        }
    }

    /**
     * Εκτελεί μεταδιατεταγμένη διάσχιση (postorder traversal
     * Επισκέπτεται τους κόμβους με σειρά: αριστερό παιδί ->
     */
    public void postorder() {
        postorderRec(root);
    }

    /**
     * Αναδρομική βοηθητική μέθοδος για την μεταδιατεταγμένη
     *
     * @param root η ρίζα του τρέχοντος υποδέντρου
     */
    private void postorderRec(Node root) {
        if (root != null) {
            postorderRec(root.left);
```

```
            postorderRec(root.right);
            System.out.print(root.value + " ");
        }
    }
}
```

```
🌳 Binary Search Tree Demo

📥 Inserting values:
6 4 3 5 8 7 9

📊 Traversals:
Inorder:   3 4 5 6 7 8 9
Preorder:  6 4 3 5 8 7 9
Postorder: 3 5 4 7 9 8 6
```

Ο ΣΗΜΑΝΣΗ (Ο NOTATION)

- Εισαγωγή (insert): O(log n) κατά μέσο όρο, O(n) στη χειρότερη περίπτωση (αν το δέντρο είναι εκφυλισμένο)
- Αναζήτηση (search): O(log n) κατά μέσο όρο, O(n) στη χειρότερη περίπτωση
- Διαγραφή (delete): O(log n) κατά μέσο όρο, O(n) στη χειρότερη περίπτωση

Οι διασχίσεις (inorder, preorder, postorder) έχουν πολυπλοκότητα O(n), καθώς κάθε κόμβος επισκέπτεται μία φορά.

ΒΙΒΛΙΟΘΗΚΕΣ

- lib.BinarySearchTree
  - Μέθοδος insert(): Εισαγωγή στοιχείων
  - Μέθοδος inorder(): Ενδοδιατεταγμένη διάσχιση
  - Μέθοδος preorder(): Προδιατεταγμένη διάσχιση
  - Μέθοδος postorder(): Μεταδιατεταγμένη διάσχιση

ΛΕΙΤΟΥΡΓΙΕΣ

1. Εισαγωγή στοιχείων στο δέντρο
2. Εμφάνιση διασχίσεων:
    - Ενδοδιατεταγμένη (Inorder)
    - Προδιατεταγμένη (Preorder)
    - Μεταδιατεταγμένη (Postorder)

Γενεαλογικό Δέντρο

1. **Αναπαράσταση Δεδομένων σε CSV Αρχείο**

    Το αρχείο CSV περιέχει όλα τα δεδομένα σωστά διαμορφωμένα (όνομα, φύλο, σχέσεις) και ακολουθεί τις προδιαγραφές.

    ```
    name,gender,relation,related_to
    Robert,man,father,Joffrey
    Cersei,woman,mother,Joffrey
    Robert,man,father,Tommen
    Cersei,woman,mother,Tommen
    Robert,man,father,Myrcella
    Cersei,woman,mother,Myrcella
    Joffrey,man,brother,Tommen
    Joffrey,man,brother,Myrcella
    Tommen,man,brother,Myrcella
    Tywin,man,father,Cersei
    Joanna,woman,mother,Cersei
    Tywin,man,father,Jaime
    ```

2. **Φόρτωση και Ανάγνωση Δεδομένων (20 μονάδες)**

    Η εφαρμογή διαβάζει πλήρως και σωστά το αρχείο CSV και εμφανίζει τα δεδομένα σωστά στη γραμμή εντολών (όνομα και φύλο).

```
// Χρήση του absolute path για το αρχείο
          String projectRoot = System.getProperty("user.di
          String csvPath = projectRoot + "/public/input.csv

          family.loadFromCSV(csvPath);
          System.out.println("\n👥 Μέλη Οικογενειακού Δέντρ
          System.out.println("----------------------------
          family.displayAll();
```

3. **Εύρεση Σχέσεων Μεταξύ Ατόμων**

   Η εφαρμογή εντοπίζει σωστά όλες τις σχέσεις μεταξύ των ατόμων (πατέρας, μητέρα, αδερφός, κ.λπ.), και κάθε σχέση υλοποιείται μέσω ξεχωριστής μεθόδου.

```
      // Getters τύπων σχέσεων (με βάση το φύλο)

   /**
    * Επιστρέφει τον τύπο γονιού (πατέρας/μητέρα) με βάση τ
    */
   private String getParentType(String person) {
       return persons.get(person).getGender().equals("man")
   }

   /**
    * Επιστρέφει τον τύπο παιδιού (γιος/κόρη) με βάση το φύλ
    */
   private String getChildType(String person) {
       return persons.get(person).getGender().equals("man")
   }

   /**
    * Επιστρέφει τον τύπο αδερφού (αδερφός/αδερφή) με βάση τ
    */
   private String getSiblingType(String person) {
```

```
        return persons.get(person).getGender().equals("man")
    }

    /**
     * Επιστρέφει τον τύπο συζύγου (άντρας/γυναίκα) με βάση τ
     */
    private String getSpouseType(String person) {
        return persons.get(person).getGender().equals("man")
    }

    /**
     * Επιστρέφει τον τύπο παππού/γιαγιάς με βάση το φύλο
     */
    private String getGrandparentType(String person) {
        return persons.get(person).getGender().equals("man")
    }

    /**
     * Επιστρέφει τον τύπο εγγονού/εγγονής με βάση το φύλο
     */
    private String getGrandchildType(String person) {
        return persons.get(person).getGender().equals("man")
    }

    /**
     * Επιστρέφει τον τύπο θείου/θείας με βάση το φύλο
     */
    private String getUncleAuntType(String person) {
        return persons.get(person).getGender().equals("man")
    }

    /**
     * Επιστρέφει τον τύπο ανιψιού/ανιψιάς με βάση το φύλο
     */
    private String getNephewNieceType(String person) {
```

```
        return persons.get(person).getGender().equals("man")
    }
```

4. **Screencasting**

   Το βίντεο καλύπτει πλήρως τη λειτουργία της εφαρμογής, δείχνει όλες τις λειτουργίες και περιλαμβάνει σαφείς εξηγήσεις.

   https://youtu.be/s4FuF-0I0us

   > https://youtu.be/s4FuF-0I0us

Κριτήρια για τις Κατηγορίες:

Η υλοποίηση είναι πλήρης, χωρίς σημαντικά σφάλματα, και καλύπτει όλες τις απαιτήσεις με ακρίβεια. Για να εξασφαλιστεί, υλοποιήθηκε μια σειρά από τέστ.

```
package tests;

import lib.BinarySearchTree;
import org.junit.Test;
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;
import static org.junit.Assert.*;

public class BinarySearchTreeTest {
    @Test
    public void testInorderTraversal() {
        System.out.println("\n📋 Εκτέλεση δοκιμής: Ενδοδιατετ
        BinarySearchTree bst = new BinarySearchTree();

        System.out.println("➡️ Εισαγωγή τιμών: 5, 3, 7, 1, 9"
        bst.insert(5);
        bst.insert(3);
        bst.insert(7);
```

```
        bst.insert(1);
        bst.insert(9);

        ByteArrayOutputStream outContent = new ByteArrayOutpu
        PrintStream originalOut = System.out;
        System.setOut(new PrintStream(outContent));

        bst.inorder();
        System.setOut(originalOut);

        System.out.println("➡️ Αναμενόμενη σειρά: 1 3 5 7 9")
        System.out.println("➡️ Πραγματική σειρά: " + outConte
        assertEquals("1 3 5 7 9 ", outContent.toString());
        System.out.println("✅ Η δοκιμή ολοκληρώθηκε επιτυχώ
    }

    @Test
    public void testPreorderTraversal() {
        BinarySearchTree bst = new BinarySearchTree();
        bst.insert(5);
        bst.insert(3);
        bst.insert(7);

        ByteArrayOutputStream outContent = new ByteArrayOutpu
        System.setOut(new PrintStream(outContent));

        bst.preorder();
        assertEquals("5 3 7 ", outContent.toString());
    }

    @Test
    public void testPostorderTraversal() {
        BinarySearchTree bst = new BinarySearchTree();
        bst.insert(5);
        bst.insert(3);
        bst.insert(7);
```

```
        ByteArrayOutputStream outContent = new ByteArrayOutpu
        System.setOut(new PrintStream(outContent));

        bst.postorder();
        assertEquals("3 7 5 ", outContent.toString());
    }
}
```

```
package tests;

import lib.GenealogicalTree;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class GenealogicalTreeTest {
    private GenealogicalTree tree;
    private String testCsvPath;

    @Before
    public void setUp() throws IOException {
        System.out.println("\n📋 Προετοιμασία δοκιμών γενεαλο
        tree = new GenealogicalTree();
        testCsvPath = "public/test_family.csv";

        System.out.println("➡️ Δημιουργία δοκιμαστικού αρχείο
        FileWriter writer = new FileWriter(testCsvPath);
        writer.write("name,gender,relation,relatedTo\n");
        writer.write("John,man,father,Mary\n");
        writer.write("Anna,woman,mother,Mary\n");
        writer.write("Mary,woman,daughter,John\n");
```

```
        writer.write("Bob,man,spouse,Mary\n");
        writer.write("Tom,man,father,Bob\n");
        writer.write("Lisa,woman,sister,Mary\n");
        writer.close();
        System.out.println("✅ Προετοιμασία ολοκληρώθηκε");
    }

    @Test
    public void testLoadFromCSV() throws IOException {
        System.out.println("\n📋 Εκτέλεση δοκιμής: Φόρτωση απ
        tree.loadFromCSV(testCsvPath);

        System.out.println("➡️ Έλεγχος σχέσεων:");
        checkAndPrintRelationship("John", "Mary", "father");
        checkAndPrintRelationship("Anna", "Mary", "mother");
        checkAndPrintRelationship("Mary", "John", "daughter")
        checkAndPrintRelationship("Mary", "Bob", "wife");
        checkAndPrintRelationship("Lisa", "Mary", "sister");

        System.out.println("✅ Η δοκιμή ολοκληρώθηκε επιτυχώς
    }

    private void checkAndPrintRelationship(String person1, St
        String actualRelation = tree.findRelationship(person1
        System.out.println("   🔍 " + person1 + " είναι " + a
        assertEquals(expectedRelation, actualRelation);
    }

    @Test
    public void testRelationships() throws IOException {
        tree.loadFromCSV(testCsvPath);
        assertEquals("father", tree.findRelationship("John",
        assertEquals("daughter", tree.findRelationship("Mary"
        assertEquals("husband", tree.findRelationship("Bob",
        assertEquals("sister", tree.findRelationship("Lisa",
    }
```

```
    @Test
    public void testExtendedRelationships() throws IOExceptio
        tree.loadFromCSV(testCsvPath);
        assertEquals("father-in-law", tree.findRelationship("
        assertEquals("brother-in-law", tree.findRelationship(
    }

    public void tearDown() {
        File testFile = new File(testCsvPath);
        if (testFile.exists()) {
            testFile.delete();
        }
    }
}
```

```
package tests;

import lib.Node;
import org.junit.Test;
import static org.junit.Assert.*;

public class NodeTest {
    @Test
    public void testNodeCreation() {
        System.out.println("\n📋 Εκτέλεση δοκιμής: Δημιουργίο
        Node node = new Node(5);
        System.out.println("➡️ Δημιουργήθηκε κόμβος με τιμή:
        System.out.println("➡️ Έλεγχος αριστερού παιδιού: " +
        System.out.println("➡️ Έλεγχος δεξιού παιδιού: " + (n
        assertEquals(5, node.value);
        assertNull(node.left);
        assertNull(node.right);
        System.out.println("✅ Η δοκιμή ολοκληρώθηκε επιτυχώς
    }
```

```
    @Test
    public void testNodeConnections() {
        System.out.println("\n📋 Εκτέλεση δοκιμής: Συνδέσεις
        Node root = new Node(5);
        Node left = new Node(3);
        Node right = new Node(7);

        System.out.println("➡️ Δημιουργία κόμβων:");
        System.out.println("   🌳 Ρίζα: " + root.value);
        System.out.println("   🌿 Αριστερό παιδί: " + left.va
        System.out.println("   🌿 Δεξί παιδί: " + right.value

        root.left = left;
        root.right = right;

        System.out.println("\n➡️ Έλεγχος συνδέσεων:");
        System.out.println("   🔍 Αριστερό παιδί της ρίζας: "
        System.out.println("   🔍 Δεξί παιδί της ρίζας: " + r

        assertEquals(left, root.left);
        assertEquals(right, root.right);
        assertEquals(3, root.left.value);
        assertEquals(7, root.right.value);
        System.out.println("✅ Η δοκιμή ολοκληρώθηκε επιτυχώς
    }
}
```

```
package tests;

import lib.Person;
import org.junit.Test;
import static org.junit.Assert.*;

public class PersonTest {
```

```
    @Test
    public void testPersonCreation() {
        System.out.println("\n📋 Εκτέλεση δοκιμής: Δημιουργίο
        Person person = new Person("John", "man");
        System.out.println("➡️ Δημιουργήθηκε άτομο με όνομα:
        System.out.println("➡️ Φύλο ατόμου: " + person.getGen
        assertEquals("John", person.getName());
        assertEquals("man", person.getGender());
        System.out.println("✅ Η δοκιμή ολοκληρώθηκε επιτυχώς
    }

    @Test
    public void testPersonWithFemaleGender() {
        System.out.println("\n📋 Εκτέλεση δοκιμής: Δημιουργίο
        Person person = new Person("Maria", "woman");
        System.out.println("➡️ Δημιουργήθηκε άτομο με όνομα:
        System.out.println("➡️ Φύλο ατόμου: " + person.getGen
        assertEquals("Maria", person.getName());
        assertEquals("woman", person.getGender());
        System.out.println("✅ Η δοκιμή ολοκληρώθηκε επιτυχώς
    }
}
```