Структури от Данни и Обектно-Ориентирано Програмиране
спец. Компютърни Науки
Практикум №11
12.05.2010г.

ТЕМА: Дървета

Клас за представяне на дърво с информационно поле от тип E.

```
/*
 * Abstract structure Binary Tree.
*/
public class BinaryTree<E> {

	protected E value;
	protected BinaryTree<E> parent;
	protected BinaryTree<E> left;
	protected BinaryTree<E> right;

	public static final BinaryTree EMPTY_TREE = new BinaryTree(); // Empty tree

	/**
	 * Constructs an empty tree
	 */
	private BinaryTree() {
		value = null;
		parent = null;
		left = right = this;
	}

	/**
	 * Constructs an one-element tree
	 */
	public BinaryTree(E value) {
		this.value = value;
		parent = null;
		left = right = EMPTY_TREE;
	}
```

```
/**
 * Constructs a tree with given subtrees
 */
public BinaryTree(E value, BinaryTree<E> left, BinaryTree<E> right) {
        this(value);
        setLeft(left);
        setRight(right);
}

/**
 * Returns the left subtree of this tree
 */
public BinaryTree<E> left() {
        return left;
}

/**
 * Returns the right subtree of this tree
 */
public BinaryTree<E> right() {
        return right;
}

/**
 * Returns the parent of this tree
 */
public BinaryTree<E> parent() {
        return parent;
}

/**
 * Returns the value in the root of this tree
 */
public E value() {
        return value;
}

/**
 * Sets the left subtree of this tree
 */
public void setLeft(BinaryTree<E> bt) {
        if(isEmpty())
                return;
```

```
        if(left != null && left.parent() == this)
                left.setParent(null);
        left = bt;
        left.setParent(this);
}

/**
 * Sets the right subtree of this tree
 */
public void setRight(BinaryTree<E> bt) {
        if(isEmpty())
                return;
        if(right != null && right.parent() == this)
                right.setParent(null);
        right = bt;
        right.setParent(this);
}

/**
 * Sets the parent of this tree
 */
protected void setParent(BinaryTree<E> bt) {
        if (!isEmpty()) {
   parent = bt;
}
}

/**
 * Sets the value of the root of this tree
 */
public void setValue(E value) {
        this.value = value;
}

/**
 * Returns number of nodes of the tree
 */
public int size() {
        if(isEmpty())
                return 0;
        return 1 + left.size() + right.size();
}
```

```
/**
 * Returns the root of the tree
 */
public BinaryTree root() {
	if(parent == null)
		return this;
	else
		return parent.root();
}

/**
 * Returns the height of the tree
 */
public int height() {
	if(isEmpty())
		return -1;
	return 1 + Math.max(left.height(), right.height());
}

/**
 * Returns the depth of the tree
 */
public int depth() {
	if(parent == null)
		return 0;
	else
		return 1 + parent.depth();
}

/**
 * Returns true if tree is empty
 */
public boolean isEmpty() {
	return value == null;
}

/**
 * Returns true if tree is full
 */
public boolean isFull() {
	if(isEmpty())
		return true;
	if(left.height()!=right.height())
```

```
                return false;
        return left.isFull() && right.isFull();
}

/**
 * Returns true if tree is empty balanced
 */
public boolean isBalanced() {
        if(isEmpty())
                return true;
        else
                return Math.abs(left.height()-right.height())<=1 &&
                                left.isBalanced() && right.isBalanced();
}

/**
 * Returns true if this is left child of its parent
 */
public boolean isLeftChild() {
        if(parent == null)
                return false;
        else
                return this == parent.left;
}

/**
 * Returns true if this is right child of its parent
 */
public boolean isRightChild() {
        if(parent == null)
                return false;
        return this == parent.right;
}

/**
 * Returns a string, representing the tree
 */
public String toString() {
        if (isEmpty()) return "<BinaryTree: empty>";
        StringBuffer s = new StringBuffer();
        s.append("<BinaryTree "+value());
        if (!left().isEmpty()) s.append(" "+left());
        else s.append(" -");
```

```
        if (!right().isEmpty()) s.append(" "+right());
        else s.append(" -");
        s.append('>');
        return s.toString();
    }

    public void printTree() {
        . . . . . .
    }

    private void preOrder(BinaryTree bt, int level) {
        . . . . . .
    }

    public void printInOrder() {
        . . . . . .
    }

    private void inOrder(BinaryTree bt) {
        . . . . . .
    }
}
```

Задача 1: Напишете реализацията на методите **printTree(), preOrder(...), printInOrder()** и **inOrder(…)**.

Задача 2: Напишете клас ***Main***, който демонстрира всички методи на класа ***BinaryTree***.