ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

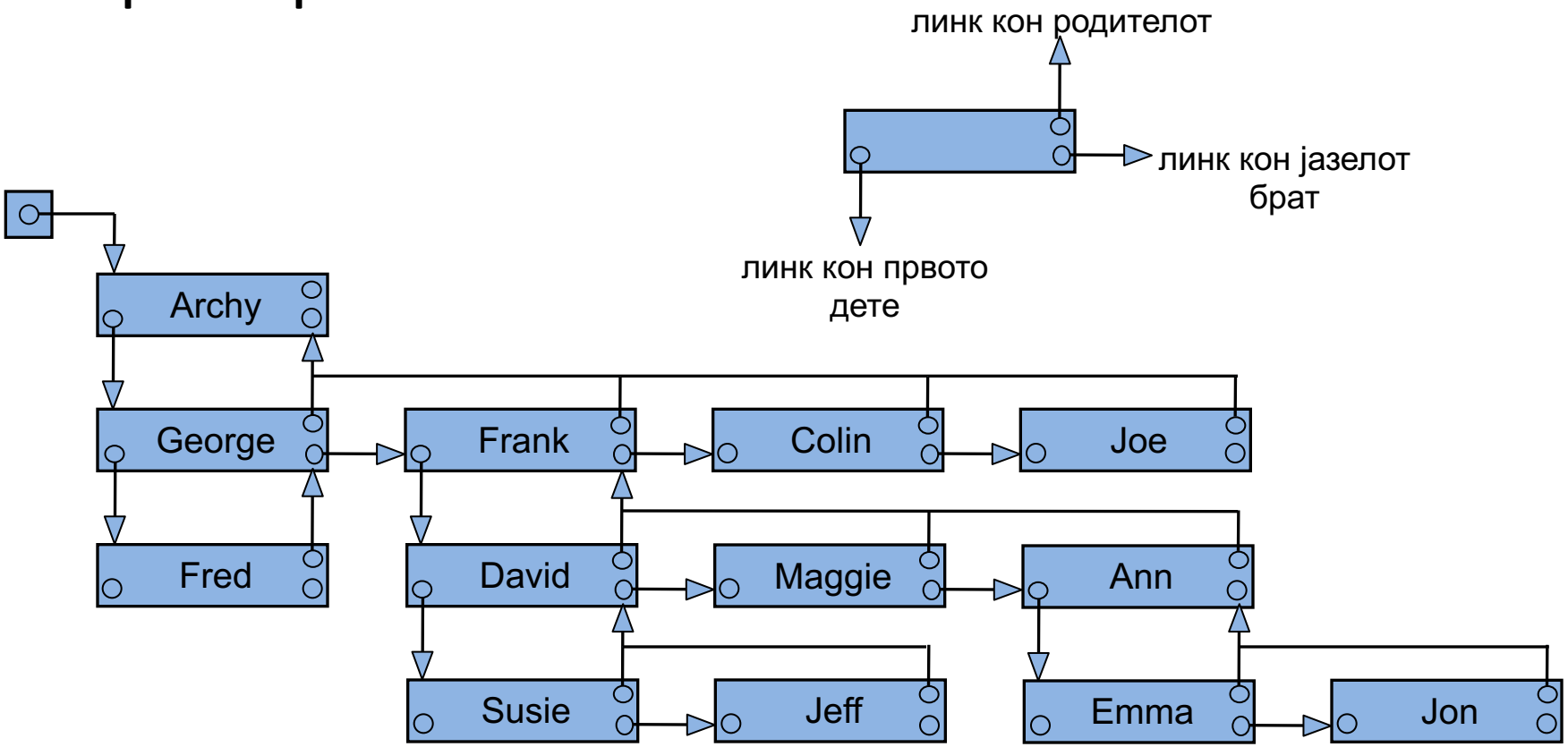
Хиерархиски структури - дрва

Алгоритми и податочни структури

Аудиториска вежба 9

Имплементација на дрво

- Пример:



Општо дрво - Java

```
import java.util.Iterator;

public interface Tree<E> {
    // ///////////Accessors ////////////

    public Tree.Node<E> root();

    public Tree.Node<E> parent(Tree.Node<E> node);

    public int childCount(Tree.Node<E> node);

    // ///////////Transformers ////////////
    public void makeRoot(E elem);

    public Tree.Node<E> addChild(Tree.Node<E> node, E elem);

    public void remove(Tree.Node<E> node);

    // ///////////Iterator ////////////
    public Iterator<E> children(Tree.Node<E> node);

    // //////Inner interface for tree nodes ////////
    public interface Node<E> {

        public E getElement();

        public void setElement(E elem);

    }
}
```

Општо дрво - Java

```
import java.util.Iterator;
import java.util.NoSuchElementException;

public class SLLTree<E> implements Tree<E> {

    // SLLNode is the implementation of the Node interface
    class SLLNode<P> implements Node<P> {

        // Holds the links to the needed nodes
        SLLNode<P> parent, sibling, firstChild;

        // Hold the data
        P element;

        public SLLNode(P o) {
            element = o;
            parent = sibling = firstChild = null;
        }

        public P getElement() {
            return element;
        }

        public void setElement(P o) {
            element = o;
        }

    }
```

Внатрешна класа
SLL Node

Општо дрво - Java

```
protected SLLNode<E> root;

public SLLTree() {
    root = null;
}

public Node<E> root() {
    return root;
}

public Tree.Node<E> parent(Tree.Node<E> node) {
    return ((SLLNode<E>) node).parent;
}

public int childCount(Tree.Node<E> node) {
    SLLNode<E> tmp = ((SLLNode<E>) node).firstChild;
    int num = 0;
    while (tmp != null) {
        tmp = tmp.sibling;
        num++;
    }
    return num;
}
```

Корен на дрвото

Општо дрво - Java

```
public void makeRoot(E elem) {
    root = new SLLNode<E>(elem);
}

public Node<E> addChild(Node<E> node, E elem) {
    SLLNode<E> tmp = new SLLNode<E>(elem);
    SLLNode<E> curr = (SLLNode<E>) node;
    tmp.sibling = curr.firstChild;
    curr.firstChild = tmp;
    tmp.parent = curr;
    return tmp;
}

public Iterator<E> children(Tree.Node<E> node) {
    return new SLLTreeIterator<E>(((SLLNode<E>) node).firstChild);
}
```

Општо дрво - Java

```
public void remove(Tree.Node<E> node) {
    SLLNode<E> curr = (SLLNode<E>) node;
    if (curr.parent != null) {
        if (curr.parent.firstChild == curr) {
            // The node is the first child of its parent
            // Reconnect the parent to the next sibling
            curr.parent.firstChild = curr.sibling;
        } else {
            // The node is not the first child of its parent
            // Start from the first and search the node in the sibling list
            // and remove it
            SLLNode<E> tmp = curr.parent.firstChild;
            while (tmp.sibling != curr) {
                tmp = tmp.sibling;
            }
            tmp.sibling = curr.sibling;
        }
    } else {
        root = null;
    }
}
```

Општо дрво - Java

```
class SLLTreeIterator<T> implements Iterator<T> {

    SLLNode<T> start, current;

    public SLLTreeIterator(SLLNode<T> node) {
        start = node;
        current = node;
    }

    public boolean hasNext() {
        return (current != null);
    }

    public T next() throws NoSuchElementException {
        if (current != null) {
            SLLNode<T> tmp = current;
            current = current.sibling;
            return tmp.getElement();
        } else {
            throw new NoSuchElementException();
        }
    }

    public void remove() {
        if (current != null) {
            current = current.sibling;
        }
    }
}
```

Задача 1

- Да се напише метод за печатење на јазлите на едно дрво. Секој јазел треба да биде испечатен во еден ред, а пред да се испечати содржината на јазелот треба да се вметнат онолку празни места колку што е нивото на тој јазел.

Задача 1 - Java

```
void printTreeRecursive(Node<E> node, int level) {
    if (node == null)
        return;
    int i;
    SLLNode<E> tmp;

    for (i = 0; i < level; i++)
        System.out.print("  ");
    System.out.println(node.getElement().toString());
    tmp = ((SLLNode<E>) node).firstChild;
    while (tmp != null) {
        printTreeRecursive(tmp, level + 1);
        tmp = tmp.sibling;
    }
}

public void printTree() {
    printTreeRecursive(root, 0);
}
```

Задача 1 - Java

```
public class SLLTreeTest {

    public static void main(String[] args) {

        Tree.Node<String> a, b, c, d;

        SLLTree<String> t = new SLLTree<String>();

        t.makeRoot("C:");

        a = t.addChild(t.root, "Program files");
        b = t.addChild(a, "CodeBlocks");
        c = t.addChild(b, "codeblocks.dll");
        c = t.addChild(b, "codeblocks.exe");
        b = t.addChild(a, "Nodepad++");
        c = t.addChild(b, "langs.xml");
        d = c;
        c = t.addChild(b, "readme.txt");
        c = t.addChild(b, "notepad++.exe");
        a = t.addChild(t.root, "Users");
        b = t.addChild(a, "Darko");
        c = t.addChild(b, "Desktop");
        c = t.addChild(b, "Downloads");
        c = t.addChild(b, "My Documents");
        c = t.addChild(b, "My Pictures");
        b = t.addChild(a, "Public");
        a = t.addChild(t.root, "Windows");
        b = t.addChild(a, "Media");

        t.printTree();

    }

}
```

Излез:

```
C:
  Windows
    Media
  Users
    Public
    Darko
      My Pictures
      My Documents
      Downloads
      Desktop
  Program files
    Nodepad++
      notepad++.exe
      readme.txt
      langs.xml
    CodeBlocks
      codeblocks.exe
      codeblocks.dll
```

Задача 2

- Да се напише метод за пресметување на степенот на едно дрво.

Задача 2 - Java

```
public int countMaxChildren() {
    return countMaxChildrenRecursive(root);
}

int countMaxChildrenRecursive(SLLNode<E> node) {
    int t = childCount(node);
    SLLNode<E> tmp = node.firstChild;
    while (tmp != null) {
        t = Math.max(t, countMaxChildrenRecursive(tmp));
        tmp = tmp.sibling;
    }
    return t;
}
```

Бинарно дрво - Java

```
public class BNode<E> {

    public E info;
    public BNode<E> left;
    public BNode<E> right;

    static int LEFT = 1;
    static int RIGHT = 2;

    public BNode(E info) {
        this.info = info;
        left = null;
        right = null;
    }

    public BNode(E info, BNode<E> left, BNode<E> right) {
        this.info = info;
        this.left = left;
        this.right = right;
    }

}
```

Бинарно дрво - Java

```
public class BTree<E> {

    public BNode<E> root;

    public BTree() {
        root = null;
    }

    public BTree(E info) {
        root = new BNode<E>(info);
    }

    public void makeRoot(E elem) {
        root = new BNode<E>(elem);
    }
```

Бинарно дрво - Java

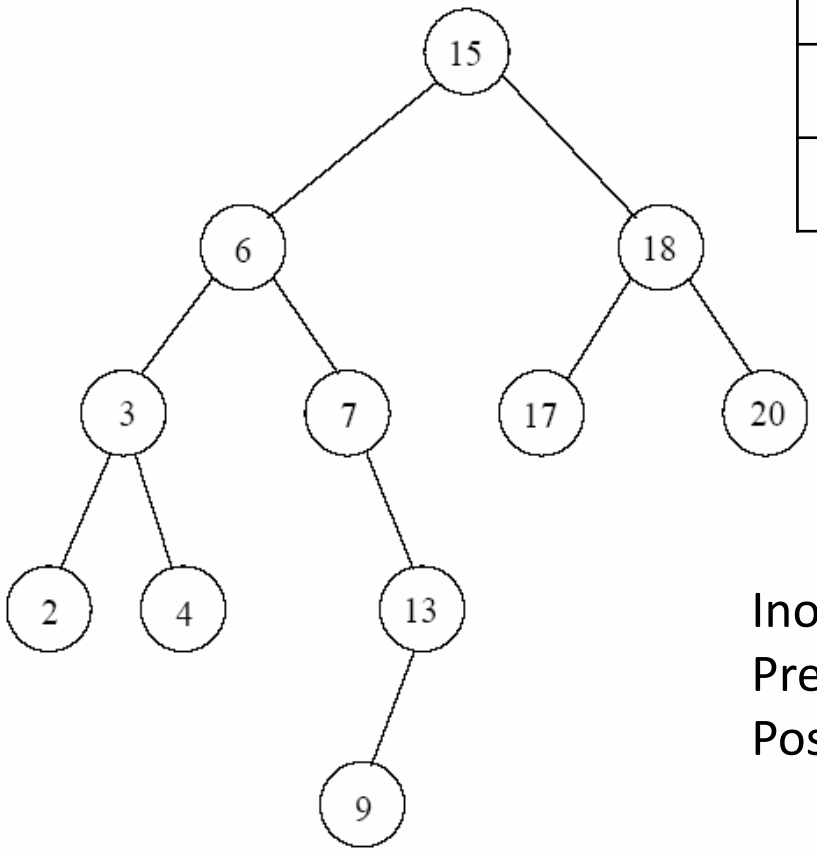
```
public BNode<E> addChild(BNode<E> node, int where, E elem) {

    BNode<E> tmp = new BNode<E>(elem);

    if (where == BNode.LEFT) {
        if (node.left != null)  // veke postoi element
            return null;
        node.left = tmp;
    } else {
        if (node.right != null) // veke postoi element
            return null;
        node.right = tmp;
    }

    return tmp;
}
```

Изминување на бинарно дрво



INORDER	L O D	(лево, корен, десно)
PREORDER	O L D	(корен, лево, десно)
POSTORDER	L D O	(лево, десно, корен)

Inorder: 2, 3, 4, 6, 7, 9, 13, 15, 17, 18, 20
Preorder: 15, 6, 3, 2, 4, 7, 13, 9, 18, 17, 20
Postorder: 2, 4, 3, 9, 13, 7, 6, 17, 20, 18, 15

Изминување на бинарно дрво

- Можно е еднозначно да се дефинира дрвото ако се дадени низите на јазли во:
  - Inorder и preorder изминување
  - Inorder и postorder изминување
- Не е секогаш можно да се дефинира дрвото ако се дадени низите на јазли во:
  - Preorder и postorder изминување
- Докажете ги горните тврдења преку примери

Изминувања на бинарно дрво - Java

```
public void inorder() {
    System.out.print("INORDER: ");
    inorderR(root);
    System.out.println();
}

public void inorderR(BNode<E> n) {
    if (n != null) {
        inorderR(n.left);
        System.out.print(n.info.toString()+" ");
        inorderR(n.right);
    }
}
```

```
public void postorder() {
    System.out.print("POSTORDER: ");
    postorderR(root);
    System.out.println();
}

public void postorderR(BNode<E> n) {
    if (n != null) {
        postorderR(n.left);
        postorderR(n.right);
        System.out.print(n.info.toString()+" ");
    }
}
```

```
public void preorder() {
    System.out.print("PREORDER: ");
    preorderR(root);
    System.out.println();
}

public void preorderR(BNode<E> n) {
    if (n != null) {
        System.out.print(n.info.toString()+" ");
        preorderR(n.left);
        preorderR(n.right);
    }
}
```

Задача 3

- Да се напише нерекурзивна функција за изминување на бинарно дрво во inorder. Аргумент во функцијата е покажувач кон коренот на дрвото.

Задача 3 - Java

```
public void inorderNonRecursive() {
    ArrayStack<BNode<E>> s = new ArrayStack<BNode<E>>(100);
    BNode<E> p = root;
    System.out.print("INORDER (nonrecursive): ");

    while (true) {
        // pridvizuvanje do kraj vo leva nasoka pri sto site koreni
        // na potstebla se dodavaat vo magacin za podocnezna obrabotka
        while (p != null) {
            s.push(p);
            p = p.left;
        }

        // ako magacinot e prazen znaci deka stebloto e celosno izminato
        if (s.isEmpty())
            break;

        p = s.peek();
        // pecatenje (obrabotka) na jazelot na vrvot od magacinot
        System.out.print(p.info.toString()+" ");
        // brisenje na obraboteniot jazel od magacinot
        s.pop();
        // pridvizuvanje vo desno od obraboteniot jazel i povtoruvanje na
        // postapkata za desnoto potsteblo na jazelot
        p = p.right;

    }
    System.out.println();

}
```

Задача 3. Надополнување

- За дома:
  - Да се напише функција preorderNonRecursive за нерекурзивно изминување на бинарно дрво во preorder
  - Да се напише функција postorderNonRecursive за нерекурзивно изминување на бинарно дрво во preorder

Задача 4

- Да се напише функција што го дава бројот на внатрешни (нетерминални) јазли во дадено бинарно дрво.

Задача 4 - Java

```
int insideNodesR(BNode<E> node) {
    if (node == null)
        return 0;

    if ((node.left == null)&&(node.right == null))
        return 0;

    return insideNodesR(node.left) + insideNodesR(node.right) + 1;
}

public int insideNodes() {
    return insideNodesR(root);
}
```

Задача 5

- Да се напише функција што го дава бројот на листови во дадено бинарно дрво.

Задача 5 - Java

```
int leavesR(BNode<E> node) {
    if (node != null) {
        if ((node.left == null)&&(node.right == null))
            return 1;
        else
            return (leavesR(node.left) + leavesR(node.right));
    } else {
        return 0;
    }
}

public int leaves() {
    return leavesR(root);
}
```

Задача 6

- Да се напише функција која за дадено бинарно дрво ќе ја врати неговата максимална длабочина (најдолгата патека во дрвото од коренот до некој лист).

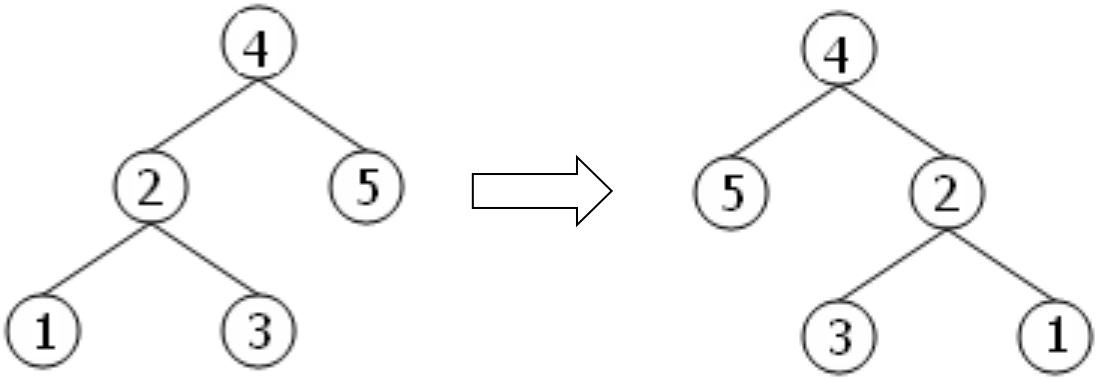
Задача 6 - Java

```
int depthR(BNode<E> node) {
    if (node == null)
        return 0;
    if ((node.left == null)&&(node.right == null))
        return 0;
    return (1 + Math.max(depthR(node.left), depthR(node.right)));
}

public int depth() {
    return depthR(root);
}
```

Задача 7

- Да се напише функција која дадено бинарно дрво ќе го трансформира на тој начин што ќе ги замени улогите на левиот и десниот покажувач на секој јазел. Новодобиеното дрво да се пресликува симетрично во однос на оригиналното.



Задача 7 - Java

```
void mirrorR(BNode<E> node) {
    BNode<E> tmp;

    if (node == null)
        return;

    // simetricno preslikuvanje na levoto i desnoto potsteblo
    mirrorR(node.left);
    mirrorR(node.right);

    // smena na ulogite na pokazuvacite na momentalniot jazel
    tmp = node.left;
    node.left = node.right;
    node.right = tmp;

}

public void mirror() {
    mirrorR(root);
}
```