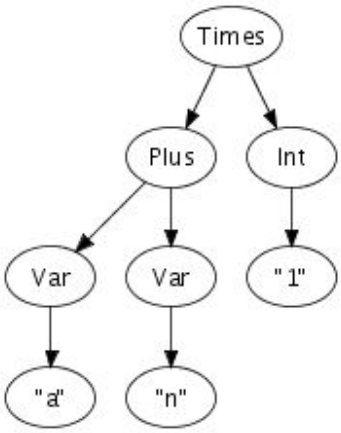
Individual Exercises - Lecture 0

The following exercises you may prefer to do on your own and discuss the outcome with your group:

1. *Familiarize yourself with Java e.g. by following the links above.*

   Some key differences compared to C# that you should look into:
   - Checked exceptions
   - No go to statements
   - Uses packages instead of namespaces
   - Different tools (IntelliJ, Eclipse or Netbeans)

2. *Write a Java program that implements a data structure for the following tree*




```
public interface Node {
   List<Node> getChildren();
}

public static class BinaryNode implements Node {
   private Node leftChild;
   private Node rightChild;
   public final String type;

   public BinaryNode(String type, Node leftChild, Node rightChild) {
       this.type = type;
       this.leftChild = leftChild;
       this.rightChild = rightChild;
   }
```

```
    @Override
    public ArrayList<Node> getChildren() {
        ArrayList<Node> list = new ArrayList<>();
        list.add(leftChild); list.add(rightChild);
        return list;
    }
}

public static class LeafNode implements Node {
    public final String value;
    public final String type;

    public LeafNode(String type, String value) {
        this.type = type;
        this.value = value;
    }

    @Override
    public ArrayList<Node> getChildren() {
        return new ArrayList<>();
    }
}
```

3. *Extend your Java program to traverse the tree depth-first and print out information in nodes and leaves as it goes along.*

```
public interface Node {
        List<Node> getChildren();
        void printTree(int indentation);
}

public class Main {
    public static void main(String[] args) {
       LeafNode int1 = new LeafNode("Int", "1");
       LeafNode varA = new LeafNode("Var", "a");
       LeafNode varN = new LeafNode("Var", "n");
       BinaryNode plus = new BinaryNode("Plus", varA, varN);
       BinaryNode times = new BinaryNode("Times", plus, int1);

       times.printTree(0);
    }
```

```
    public static class BinaryNode implements Node{

        ...

        @Override
        public void printTree(int indentation) {
            String indent = "    ".repeat(indentation);

            System.out.println(indent + "(" + type + " node");
            for (Node child : getChildren()){
                child.printTree(indentation + 1);
            }
            System.out.println(indent + ")");
        }
    }

    public static class LeafNode extends BinaryNode {

        ...

        @Override
        public void printTree(int indentation) {
            String indent = "    ".repeat(indentation);
            System.out.println(indent + "(" + type + ", value " + value + ")");
        }
    }
}
```

4. *Write a Java program that can read the string "a + n * 1" and produce a collection of objects containing the individual symbols when blank spaces are ignored (or used as separator).*

```
public class Main {
    public static void main(String[] args) {
        // Using Scanner for Getting Input from User
        Scanner in = new Scanner(System.in);
        String input = in.nextLine();

        ArrayList<Token> tokens = new ArrayList<>();
        for(char c : input.toCharArray()){
            switch(c){
                case ' ':
                    break;
                default:
                    tokens.add(new Token(c + ""));
            }
        }
        System.out.println(tokens);
    }

    public static class Token{
        String value;
        public Token(String value) {
            this.value = value;
        }

        @Override
        public String toString() {
            return "Token{" +
                    "value='" + value + '\'' +
                    '}';
        }
    }
}
```

5. *Make a list of all the computer languages you know. (Remember to include command, scripting and text processing languages)*
   (Fx JavaScript, Java, Kotlin, Go, C#, Python, Lisp, Ruby, Perl...)

6. *How well do you know the above languages?*
   (Are there some problems you know how to solve in one language, but not in others?)

7. *Categorize them according to the following:*
   1. *Daily use*
   2. *Often*
   3. *Written a few programs*
   4. *Looked at it once or twice*

   Maybe you know some language that you have never used?

8. *Which language was your first programming language?*
   (Fx JavaScript)

9. *Use the statements associated with the Hammer Principle to evaluate each of the programming languages on your list http://hammerprinciple.com/therighttool (http://web.archive.org/web/20170709000745/http://www.hammerprinciple.com:80/therighttool)*
   First scroll down on the page and evaluate your chosen language on some of the statements. Next you can click on your language in the top panel (if it's there) to see what people generally think are its strengths and weaknesses.

10. *Compare some language features from different programming languages like for-loop in C, Java and C#. How do they differ?*
    Newer versions of C, Java and C# are syntactically identical. In older versions of C the counting variable must be declared before the for-loop. An example of a language with another type of for loop would be Python. The regular counting for-loop does not exist in Python, instead you have to use the foreach loop on the Python Range construct.

11. *Compare foreach loops in Java, C# and Python (you don't need to know Python to look at the loop)*
    Syntactically the main differences include the use of curly braces, 'in' instead of :. C# also uses the keyword foreach, whereas Java and Python simply use for.
    The loop variable must be statically typed in Java and C# (the type stays the same throughout loop execution), while the loop variable is of dynamic type in Python (the type can change throughout execution).

Group Exercises - Lecture 0

The following exercises are best done as group discussions:

1. *Compare your individual programs from exercise 2, 3 and 4.*

   (Depends on your previous answers)

2. *Make a drawing or description of the phases (internals) of a compiler (without reading the books or searching the Internet) – save this for comparison with your knowledge after the course.*
   How is the program read? How is it stored and analysed? How do we produce a result that we can execute? Which parts would be needed for these steps?

3. *Compare your individual list of programming languages and make a list of all the computer languages that group members know.*
   Do you know the same languages? Why do you think this is the case?

4. *Create a list of language features group members would like in a new language. Are any of these features in conflict with each other? How would you prioritize the features?*
   Consider features like classes, typing, types, procedures, functions, lambda etc.
   Should it have a for loop? If so, how should it look?

5. *Discuss what is needed to define a new programming language. Write down your conclusions for comparison with your knowledge after the course.*
   What should the syntax look like?
   How should the language constructs be understood (semantics)?
   How about a type system?
   Precedence of operators?