Overview

- Debriefing Exercise 2
- Briefing Exercise 3
U2.A1

- Represent tree with brackets and indented form
- Reconstruct a tree from bracket representation?

\[ S(R(H(K)), P(A(N, O), Q, T), V(J, F(G))) \]

- Yes, if the position of the nodes is irrelevant (left/right)
Some tips

- Pay attention to the number of brackets
- How are “K, N, O, etc.” called?
  - Leaves
- Longest path depends if the tree is defined as directed
  - Computer Science: tree = Connected, acyclic and directed
recursiveSort()
```
[5 1 9 2]
[5 1 9 2]
[5 1 9 2]
[5 1 9 2]
[5 1 9 2]
[5 1 9 2]
[9 1 5 2]
[9 1 5 2]
[9 5 1 2]
[9 5 1 2]
[9 5 2 1]
[9 5 2 1]
```

**recursiveSort(4)**

```
recursiveSort(3)
```

**recursiveSort(2)**

```
recursiveSort(1)
```

```text
Ist sortiert!
```

```
2 <- findLargest(0,3)
swap(0,2)
```

```
2 <- findLargest(1,3)
swap(1,2)
```

```
3 <- findLargest(2,3)
swap(2,3)
```

Swap is not necessary anymore...

→ List sorted in descending order!
/**
 * swaps two fields of {@link RandomArray#numbers}
 * @param i a valid index into {@link RandomArray#numbers}
 * @param j a valid index into {@link RandomArray#numbers}
 */
private void swap(int i, int j) {
    int tmp = numbers[j];
    numbers[j] = numbers[i];
    numbers[i] = tmp;
}
Swap

\[ \begin{align*}
X &:= X \text{ XOR } Y \\
Y &:= X \text{ XOR } Y \\
X &:= X \text{ XOR } Y
\end{align*} \]

\[
\begin{array}{c|c}
X & y \\
\hline
1010 & 0011 \\
1001 & 0011 \\
1001 & 1010
\end{array}
\]

\[
\begin{array}{c|c}
\text{X} & \text{Y} \\
\hline
1001 \oplus 0011 = 1001 \rightarrow x \\
1001 \oplus 0011 = 1010 \rightarrow y \\
1001 \oplus 1010 = 0011 \rightarrow x
\end{array}
\]

\[
\begin{array}{c|c}
0011 & 1010
\end{array}
\]
How to do the swap?

- Swap inside the loop

```c
void recursiveSort( int until ) {
    // 0 elements are considered to be sorted
    if( until == 0 )
        return;

    // sort first until-1 elements in the array
    recursiveSort( until - 1 );

    // bring the greatest element from the rest to position until-1
    for( int i = until; i < a.length; i++ )
        { if( a[i] > a[until-1] ){
            swap(until-1, i);
            }
        }
}
```
How to do the swap?

- Any better idea?
  - First find the item to swap, then do only 1 swap!

```java
void recursiveSort( int until ) {
    // 0 elements are considered to be sorted
    if( until == 0 )
        return;

    // sort first until-1 elements in the array
    recursiveSort( until - 1 );

    // find index of greatest element after until-1
    int maxIndex = until - 1;
    for( int i = until; i < a.length; i++ ) {
        if( a[i] > a[maxIndex] ) {
            maxIndex = i;
        }
    }

    // swap elements at maxIndex and until-1
    swap( until-1, maxIndex );
}
```
Coding Style

- Formatting code
- Eclipse: Ctrl+Shift+F and the code is nicely formatted (indented)

```java
while ((e+i)<=14) {
    if (a[e]> a[e+i]) {
        e++;
        i=1;
    } else
        i++;
}
```
Coding style

- Try to avoid hardcoding!

```java
x < 10

x < a.length
```

```java
if(myString.compareTo("hello world") == 0);
```

```java
private static final String REF = "hello world";
...
if(myString.compareTo(REF) == 0);
```
Coding style

- Loops

for:

  when iterating

```java
for(int i=0; i < MAX_I; ++i){
    nextIterationStep();
}
```

while:

  for specific cases

```java
int timeout = 0;
while(!userInteraction()){
    Thread.yield();
    timeout++;
}
```
Coding style

- Differences

```java
if (index >= boundary)
    return;
else if (array[index] == 'x')
    return;

if (index >= boundary ||
    array[index] == 'x')
    return;
```

Y in expression `(X || Y)` is only evaluated if `X == false` (border effect)

```java
if (index < boundary)
    if (array[index] == 'x')
        array[index] = '\0';

if (index < boundary &&
    array[index] == 'x')
    array[index] = '\0';
```

Y in expression `(X && Y)` is only evaluated if `X == true`

```java
int counter = 0;
while (counter < n) {
    ... 
    counter++;
}

for (int counter = 0;
    counter < n;
    counter++) {
    ...
}
```

Warning: counter is still defined outside the loop!

Clean counting: counter can be reused out of the `for` loop.
Coding style

- Efficiency

Object initialization is expensive!

```java
void initialize() {
    for (int i=0; i<a.length; i++) {
        Random r = new Random();
        a[i] = r.nextInt(1000);
    }
}
```

```java
void initialize() {
    Random r = new Random();
    for (int i=0; i<a.length; i++) {
        a[i] = r.nextInt(1000);
    }
}
```
a) leftChild, rightChild and father

- Root at index 0
- Direct successors for $i$ are at position $2i + 1$ and $2i + 2$

```c
int leftChild( node ){
    return 2 * node + 1;
}

int rightChild( node ){
    return 2 * node + 2;
}

int father( node ){
    return (node - 1) / 2;
}

(father(0) = -1 / 2 = 0)
```
checkTree()

Test if an input array represent a binary tree
- Each node must have a father
- The root is its own father

What about empty nodes?
- We ignore them (no need for a father)
/**
 * Check if the given array represents a valid binary tree.
 *
 * @param array a binary tree encoded as char array
 * @throws IllegalArgumentException if check fails
 */

private static void checkTree(char[] array)
{
    if (array.length == 0) throw new IllegalArgumentException("At least one, probably empty node is required.");
    for (int i=0; i<array.length; i++) {
        if (array[i] != ' ') {
            int f = father(i);
            if (array[f] == ' ') {
                throw new IllegalArgumentException("node number " + i + " has no father");
            }
        }
    }
}
/**
 * recursive toString
 * @param node index of the root of the subtree which has to be converted.
 * @param indentation the current indentation
 * @return indented form of the selected subtree
 */

private String toString(int node, String indentation)
{
    assert (tree[node] != ' ');

    String s = indentation;
    s = s + tree[node] + '\n';

    int c1 = leftChild(node);
    if (isNode(c1)) {
        s = s + toString(c1, indentation + " ");
    }

    int c2 = rightChild(node);
    if (isNode(c2)) {
        s = s + toString(c2, indentation + " ");
    }

    return s;
}
Overview

- Debriefing Exercise 2

- Briefing Exercise 3
Homework

1. Objects and references (e.g. Strings)
   - Strings vs. StringBuffer
   - Caesar cipher
   - Encrypt and decrypt, understand how the program works

2. Syntax diagrams
   - Given some diagrams, which expressions can be produced?

3. Syntax checker for trees
   - Complete the syntax diagram from class
   - Implement it
U3.A1 Hints

- **String**
  - Immutable
  - Optimization possible because static
  - Modification only through copy

- **StringBuffer**
  - Mutable
  - Easily modifyable (without copy)
  - Some operations are more expensive (e.g. search)
String vs. StringBuffer

String myString = "hello";
myString = myString + " world";

StringBuffer myStringBuffer = "hello";
myStringBuffer.append(" world");

Animation by Beat Saurenmann
More about Strings

```java
String myString = "hello";
myString = myString + " world";
myString = myString + " how";
myString = myString + " are";
myString = myString + " you";
myString = myString + " today";
```

Speicher

"hello"
"world"
"hello world"
"how"
"hello world how"
"are"
"hello world how are"
"you"
"hello world how are you"
"today"

"hello world how are you today"
### U3.A2 Hints

- Syntax diagrams were covered in class

**Var:**

```plaintext
\sim
\begin{array}{c}
\sim \\
X_1 \\
X_2 \\
\vdots \\
X_n \\
\end{array}
```

**Clause:**

```plaintext
\begin{array}{c}
( \\
\text{Var} \\
\text{OR} \\
) \\
\end{array}
```

**Expr:**

```plaintext
\text{Clause} \rightarrow \text{AND} \rightarrow \text{Clause}
```

E.g. \((\sim X_1 OR X_2) AND (X_n)\)
U3.A3 Hints

- Implementing a syntax checker for trees
  - First you have to modify the syntax to accept empty trees and subtrees
  - Implement
    - Own methods for Tree, Successor and Node

- Offset = current position in the bracket representation of the tree. At the end, the offset should be equal to str.length().

- Possible problems
  - `StringIndexOutOfBoundsException` – you are trying to access character at position n in the string, but the array is shorter than n.
Have Fun!