Informatik II (D-ITET)

Tutorial 3

TA: Anwar Hithnawi, E-mail: hithnawi@inf.ethz.ch
Distributed Systems Group, ETH Zürich
Outlook

- Exercise 2 solution discussion
- Exercise 3 overview (Strings, Syntax Diagram, Syntax checker)
Solution Ex2.Q1

- Bracket and indented form OK

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

- Can you reconstruct the tree given by the comma representation shown above without any ambiguity?

Yes, if the position of the nodes is irrelevant (left or right successor?)

- Height, leaves, longest path/s!
Solution Ex2.Q2: recursiveSort()

```plaintext
recursiveSort(4)
recursiveSort(3)
recursiveSort(2)
  recursiveSort(1)
    recursiveSort(0)
      Sorted!
9 <- findLargest(0,3)
  Swap
5 <- findLargest(1,3)
  Swap
2 <- findLargest(2,3)
  Swap
No need for further swap!

→ List is now in decreasing order!
```

Animation from Simon Mayer
/**
 * 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;
}
**Solution Ex.Q2**

- **SWAP without temporary variable, any idea?**
  (hint: using XOR)

\[
\begin{align*}
&x := x \text{ XOR } y \\
&y := x \text{ XOR } y \\
&x := x \text{ XOR } y
\end{align*}
\]

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

\[
1010 \oplus 0011 = 1001 \rightarrow x
\]

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

\[
1001 \oplus 0011 = 1010 \rightarrow y
\]

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

\[
1001 \oplus 1010 = 0011 \rightarrow x
\]

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

\[
0011 \oplus 1010 = 0011 \rightarrow x
\]

\[
0011 \quad 1010
\]
Solution Ex2.Q2e

SWAP inside a loop: good idea?

```cpp
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);
        }
    }
}
```
Solution U2.Q2

Better: search first, then exchange

```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 );
}
```

For 15 values, on average ~57 swaps in the first case and ~11 swaps in the second case.
Solution Ex2.Q2

- Coding Style – Formatting

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

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

In languages like java/C++ indentation is not a requirement but it better conveys the structure of the programs to human readers.
Solution Ex2.Q2

Coding Style: Avoid hard-coding and magic numbers!

\[
x < 10
\]

\[
x < a.length
\]

\[
if(myString.compareTo("hello world") == 0);
\]

\[
private static final String REF = "hello world";
\]

\[
if(myString.compareTo(REF) == 0);
\]

\[
private static final double PI = 3.14159265;
\]

\[
double radius, area;
\]

\[
area = PI * (radius * raduis);
\]

\[
double radius, area;
\]

\[
area = 3.14159265 * (radius * raduis);
\]
Solution Ex2.Q2

- Coding Style – Loops

For is used when iterating

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

While is used for specific cases

```java
int timeout = 0;
while( !userInteraction() )
{
    Thread.yield();
    timeout++;
}
```
Solution Ex2.Q2

- Coding Style – Efficiency

Object instansiation is expensive!

```java
void initialize() {
    Random r = new Random();
    for (int i = 0; i < a.length; i++) {
        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);
    }
}
```
Solution Ex2.Q3a

- Root at index 0
- Successor of \( i \) is at \( 2i + 1 \) and \( 2i + 2 \)
  
  \[ 2^{\text{height}-1} = 2^{\text{depth}} \leq \text{array.length} < 2^{\text{depth+1}} = 2^{\text{height}} \]

```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)
```
Solution Ex2.Q3b&c

- `checkTree(char [] array)`
  Check if the given array represents a valid binary tree
  - Test:
    - Each node has a father
    - The root is its own father
    - Solution is trivial
  - What happens with empty nodes?
    - No need for a father

- `toString()`
Outlook

- Exercise 2 solution discussion

- Exercise 3 overview (Strings, Syntax Diagram, Syntax checker)
Exercise 3

1. Objects and references (e.g. Strings)
   - Program output
   - Program analysis: Objects and references at runtime?

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

3. Syntax checker for trees
   - Complete the syntax diagrams from the class
   - Implement the syntax checker
Exercise 3 – Q 1

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

- **StringBuffer – Mutable**
  - Easily modifiable (without copy)
  - Some operations are more expensive (e.g. search)
Exercise 3 – Q 1

- Difference between String and StringBuffer

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

JAVA String concatenation

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

StringBuffer Method

Memory

- "hello"
- "world"

- "hello world"
- "world"
Exercise 3 – Q 1

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

Memory

"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"

For more in depth insight check the bytecode of sting concatenation for both sting and stingbuffer. Good reference: http://www.javaworld.com/article/2076072/build-ci-sdlc/stringbuffer-versus-string.html
Exercise 3 – Q 1

Insider’s tip: Use Eclipse’s debugger

Warning: The debugger doesn’t prompt temporary Strings, because they are created at one stage and directly discarded

```java
String myString = "hello";
myString = myString+" world";
...
```

```java
myString = "hello world"
...
```

When the line turns yellow, a modification has taken place.
Exercise 3 – Q 1

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

- **StringBuffer – Mutable**
  - Easily modifiable (without copy)
  - Some operations are more expensive (e.g. search)

Eclipse DEMO
Exercise 3 – Q 1

- Analyze the program and state which objects and references exist at the marked spot at runtime?

Example:

```java
String str = "foo";
StringBuffer buf = new StringBuffer("foobuffer");
// Markierung 1
```

- Objects @ **Markierung 1**
  (1, String, "foo")
  (2, String, "foobuffer")
  (3, StringBuffer, "foobuffer")

- References @ **Markierung 1**
  (str, 1)
  (buf, 3)
Syntax diagram

HAUS

Erdgeschoss

Stockwerk

Dach

STOCKWERK

Fenster

FENSTER

TÜR

ERDGESCHOSS

Tür

Fenster

DACH
Exercise 3 – Syntax checker for trees Q 3

Methodology:

- Methods for «Tree», «Successor» and «Nodes»
- Recursion!
- Offset Parameter

    ```java
    public static int f(String str, int offset){...}
    ```

Give the new offset to the above mentioned method. Think about how large the offset should be in the end and what happens if it doesn’t get that length.

Hint: Solution of 3a) should be integrated

- Bounds-checking:
  ```java
  StringIndexOutOfBoundsException – you are trying to access character n in the string, but the array is shorter than n.
  ```
Exercise 3 – Syntax checker for trees Q 3

- `parseEmptyOrSubTree()`: Baum → Unterbaum
- `parseSubTree()`: Unterbaum → Knoten → Nachfolger
- `parseChildren()`: Nachfolger → Baum
- `parseNode()`: Knoten → A → B → Z
Exercise 2 – Simple Syntax diagram (DEMO!)

Let's have a look at the code of this diagram!

**Single**

```
Single  
```

**SIMPLESEQUENCE**

```
Baum
```

**Node**

```
Node
```

Eclipse DEMO
Have Fun!