Overview

- Debriefing Exercise 3
- Briefing Exercise 4
U3.A1

1. Objects and references (e.g. Strings)
   - String vs. StringBuffer
   - Caesar cypher
   - Encrypt and decrypt, understand how the program works
U3.A1 Decrypt

- Inverse of encrypt
- Take each character and subtract 3 from its ASCII code

```java
/**
 * Decrypts input text based on the CaesarChiffre (i.e., removing 3 from the
 * ASCII code of each character). The decryption employs StringBuffers
 * (instead of Strings).
 *
 * @param s
ciphertext to be decrypted
 */
public static String decrypt(String s) {
    StringBuffer ret = new StringBuffer();
    for (int i = 0; i != s.length(); ++i) {
        ret.append((char) (s.charAt(i) - 3));
    }
    return ret.toString();
}
```
U3.A1 Main

- What is different?
  - Encrypt is much slower than decrypt. Why?
  - StringBuffer is more efficient for appending
  - Strings are immutable
    - Any modification leads to a new copy of the object.
U3.A1 Strings

- Why use Strings in the first place?
  - Strings are constants and allow for optimizations
  - Strings are immutable, which could be a requirement in some cases
  - The biggest benefit for StringBuffer is when we append/modify the string at runtime

More info: http://www.precisejava.com/javaperf/j2se/StringAndStringBuffer.htm
U3.A2 Syntax diagrams

Possible

Impossible

2a) Clause

\[ X_2 \quad \square \quad \times \]

\[ (\sim X_1) \quad \times \quad \square \]

\[ \sim (X_1 \text{ OR } \sim X_2) \quad \square \quad \times \]

\[ (X_2) \text{ OR } (\sim X_1 \text{ OR } X_2) \quad \square \quad \times \]
U3.A2 Syntax diagrams

Var:  \[ X_1 \]
\[ X_2 \]
\[ \vdots \]
\[ X_n \]

Clause:  \[
( \text{Var} \rightleftharpoons \text{OR} )
\]

Expr:  \[
\text{Clause} \rightarrow \text{AND} \rightarrow \text{Clause}
\]

2b) Expr

### Possible

\[
(X_1 \text{ OR } X_2) \text{ AND } (\sim X_1)
\]

\[
(X_1) \text{ AND } (\sim X_1 \text{ OR } \sim X_2) \text{ AND } (X_2)
\]

### Impossible

\[
(\sim X_1 \text{ OR } X_2)
\]

\[
(X_1) \text{ AND } (\sim X_1 \text{ OR } \sim X_2)
\]
U3.A3

- How do we change it to allow empty trees and successors?

**Baum:**

```
Knoten
```

**Unterbäume:**

```
Baum
```

**Knoten:**

```
A
```

```
B
```

```
Z
```
U3.A3

**Baum:**

**Knoten:**

**Nachfolger:**
U3.A3 Syntax checker

```java
private static int parseTree(String kd, int offset) throws ParseException {
    if (offset >= kd.length()) {
        throw new ParseException("Unexpected end of string", offset);
    }
    if (kd.charAt(offset) == '-') {
        return offset + 1;
    }
    else {
        offset = parseNode(kd, offset);
        if ((offset < kd.length()) && (kd.charAt(offset) == '(')) {
            offset += 1;
            offset = parseSubtree(kd, offset);
            if ((offset < kd.length()) && (kd.charAt(offset) == ')) {
                offset += 1;
            }
            else {
                throw new ParseException("expected ')");
            }
        }
    }
    return offset;
}
```
private static int parseSubtree(String kd, int offset) throws ParseException {
    if (offset >= kd.length()) {
        throw new ParseException("unexpected end of string after ", offset);
    }
    offset = parseTree(kd, offset);
    while ((offset < kd.length()) && (kd.charAt(offset) == ',')) {
        offset += 1;
        offset = parseTree(kd, offset);
    }
    return offset;
}
private static int parseNode(String kd, int offset) throws ParseException {
    if (offset >= kd.length()) {
        throw new ParseException("Expected a node", offset);
    }

    if (Character.isUpperCase(kd.charAt(offset))) {
        return offset + 1;
    } else {
        throw new ParseException(String.format("'%c' is not a valid node name", kd.charAt(offset)), offset);
    }
}
U3.A3 Syntax checker - parse

```java
public static void parse(String kd) throws ParseException {
    int offset = parseTree(kd, 0);
    if (offset != kd.length()) {
        throw new ParseException("Garbage at the end of the tree", offset);
    }
}
```
ParseException

double class ParseException extends Exception {

/**
 * Create a parse exception
 * @param what description of the reason
 * @param offset offset of the input string where the problem occurred
 */
    public ParseException(String what, int offset) {
        super(String.format("Parsing failed at offset %d: %s", offset, what));
    }
}
Overview

- Debriefing Exercise 3
- Briefing Exercise 4
Stack

- Abstract data type
- Collection of elements
- LIFO principle
  - Last in, first out
- Two main operations: Push and Pop
U4.A1

• Constructor
  ▪ Initializes internal Array
  ▪ Capacity is an argument to the constructor

• `toString()` with `StringBuffer`
  ▪ Expected Output: "[e0, e1, e2, …]"
  ▪ Concatenation
    ▪ String: `str += "bar"`;
    ▪ StringBuffer: `buf.append("bar")`;

• `grow()`
  ▪ Capacity doubled, copy old values
U4.A1

- `push()`, `pop()`, `peek()`, `empty()`
  - Standard stack functions
  - Arguments are of type `int`
  - If necessary, call `grow()`

- `size()`
  - Number of elements currently on the stack

- `capacity()`
  - Total number of elements which fit on the current stack until the next grow
Ackermann function

- Recursive Definition

\[ A(0, m) = m + 1 \]
\[ A(n + 1, 0) = A(n, 1) \]
\[ A(n + 1, m + 1) = A(n, A(n + 1, m)) \]

- Grows extremely fast
  - \( A(3,3) = 61 \)
  - \( A(4, 2) \) has already 19729 decimal places!!

Wilhelm Ackermann (1896 – 1962, Germany)
U4.A2

- A(1,1) given as example in the homework

- Calculate A(2,1) by hand
  - A(2,1) = A(1+1, 0+1) = A(1, A(2,0)) …

- Write down all the steps!
\[ A(4, 3) = A(3, A(4, 2)) \\
= A(3, A(3, A(4, 1))) \\
= A(3, A(3, A(3, A(4, 0)))) \\
= A(3, A(3, A(3, A(3, 1)))) \\
= A(3, A(3, A(3, A(2, A(3, 0))))) \\
= A(3, A(3, A(3, A(2, A(2, 1))))) \\
= A(3, A(3, A(3, A(2, A(1, A(2, 0)))))) \\
= A(3, A(3, A(3, A(2, A(1, A(1, 1)))))) \\
= A(3, A(3, A(3, A(2, A(1, A(0, A(1, 0))))))) \\
= A(3, A(3, A(3, A(2, A(1, A(0, A(0, 1))))))) \\
= A(3, A(3, A(3, A(2, A(1, A(0, A(0, 2))))))) \\
= A(3, A(3, A(3, A(2, A(1, 3))))) \\
= A(3, A(3, A(3, A(2, A(0, A(1, 2)))))) \\
= A(3, A(3, A(3, A(2, A(0, A(1, A(1, 1))))))) \\
= A(3, A(3, A(3, A(2, A(0, A(0, A(1, 1))))))) \\
= A(3, A(3, A(3, A(2, A(0, A(0, A(0, A(1, 0)))))))) \\
= A(3, A(3, A(3, A(2, A(0, A(0, A(0, A(0, 1)))))))) \\
= A(3, A(3, A(3, A(2, A(0, A(0, A(0, 2))))))) \\
= A(3, A(3, A(3, A(2, A(0, A(0, 3)))))) \\
= A(3, A(3, A(3, A(2, A(0, 4))))) \\
= A(3, A(3, A(3, A(2, 5)))) \\
= \ldots \\
= A(3, A(3, A(3, 13))) \\
= \ldots \\
= A(3, A(3, 65533)) \\
= \ldots \\
= A(3, 2^{65536} - 3) \\
= \ldots \\
= 2^{65536} - 3. \]
Specify the algorithm using the usual two stack operations:
- push(x)
- x = pop()

Pseudocode:
- No language-specific syntax
- Pseudocode is self-explanatory
- Based on comments

The function has the property that one cannot say in advance how deep the recursion is
- Use while instead of for-loop!
U4.A2 Iterative approach

- Ackermann’s formula always requires (exactly) two values:
  - The currently required values should be at the top of the stack…
  - What does it means when there is one item left in the stack?

```java
Stack stack = new Stack();
stack.push(4);
stack.push(7);

while(stack.size() != 1)
{
    ...
}
```
U4.A2 Implementation

```
stack.push(m)
stack.push(n)

if n == 0
    result = m + 1
else if m == 0
    push(n - 1), push(1)
else
    push(n - 1), push(n), push(m - 1)
```
A(1,1)
A(1,0)
A(0, 1)
<- 2
<- 2
A(0, 2)
<- 3
<- 3

Start
A(1,1)

Iteration

size >= 2?

m = 0
n = 0

n == 0?
m + 1

m == 0?
n - 1

else
n
m - 1

End
A(1,1) = 3

By Leyna Sadamori
U4.A2 Hints

- Stack
  - The stack from U4.A1
  - The interface should NOT be modified

- “Snapshots”
  - With toString() method of the stack

- I cannot do U4.A1
  - Use java.util.Stack<Integer>
    you just need push(), pop(), size und toString()
  - If necessary: send me an Email
U4.A3 Bytecode

- Before you disassemble the code, it must be compiled

- For Linux and Mac users:
  - Use the >> operator in the terminal to send the output to a file
  - E.g.: javap -c RecursiveAckermann >> output.txt
U4.A3 Bytecode

- For Windows:

D:\Projects\DisassemblerDemo>
javac JavapTip.java  //compiler
java JavapTip     //run
javap -c -private JavaTip   //disassembler

Common mistake: „javap is not recognized as an internal or external command, operable program or batch file”

Reason: java binaries are not defined in System variable PATH

Solution: RClick on Computer → Properties → Advanced System Settings → Environment Variables → PATH → add (where you installed the Java JDK) save and restart Windows

;C:\Program Files\Java\jdk1.7.0_31\bin
U4.A3 Bytecode example

```java
public int greaterThen(int intOne, int intTwo) {
    if (intOne > intTwo) {
        return 0;
    } else {
        return 1;
    }
}
```

```
0: iload_1
1: iload_2
2: if_icmpeq
5: icmpgt
6: ireturn
7: icmple
8: ireturn
```
U4.A3 Bytecode

- **Instructions:**
  - `iload_n`: load int from local variable
  - `aload_n`: load reference from local variable
  - `if_icmp<cond>`: Branch if int comparison succeeds
    - E.g. `if_icmple`: le = less or equal
  - `if<cond>`: Branch if comparison to zero succeeds
    - `ifeq`: equal to 0
    - `ifne`: not equal to 0
  - `invokevirtual`: invoke instance method

- **Documentation:**
Have Fun!