Informatik II
Tutorial 04

TA: Marian George
marian.george@inf.ethz.ch
Distributed Systems Group, ETH Zürich
Solution U3.A1

A1 a) Implement `decrypt(String s)` method

```java
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();
}
```

- Use `StringBuffer` instead of `String` for append/modify operations
- Convert to `String` and return to user
Solution U3.A1

A1 b) Output of the Main method:

Starting encryption (using Strings)
Done - Duration: 3732 ms.

Starting decryption (using StringBuffers)
Done - Duration: 73 ms.

Decryption successful :-)

- Strings: immutable, expensive to modify (must copy to a new String first)
- StringBuffers: mutable, cheaper to modify (edit same StringBuffer object)
Solution U3.A1

Why use Strings?

① Strings are constant (in contrast to StringBuffers, which are mutable) and thus allow for optimizations

② Operations on StringBuffer are threadsafe, which obviously comes with a cost

but Strings are thread-safe too!
Solution U3.A1

Why use Strings?

① Strings are constant (in contrast to StringBuffer, which are mutable) and thus allow for optimizations

② Operations on StringBuffer are threadsafe, which obviously comes with a cost

but Strings are thread-safe too!

Strings are immutable, so not much overhead to ensure thread-safety 😊
Solution U3.A2 – Syntax Analysis

Questions?

<table>
<thead>
<tr>
<th>erzeugbar</th>
<th>nicht erzeugbar</th>
<th>erzeugbar</th>
<th>nicht erzeugbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_2$</td>
<td>$\bigcirc$</td>
<td>$x$</td>
<td>$\sim (X_1 OR \sim X_2)$</td>
</tr>
<tr>
<td>$(\sim X_1)$</td>
<td>$x$</td>
<td>$\bigcirc$</td>
<td>$(X_2) OR (\sim X_1 OR X_2)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>erzeugbar</th>
<th>nicht erzeugbar</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(X_1 OR X_2) AND (\sim X_1)$</td>
<td>$x$</td>
</tr>
<tr>
<td>$(X_1) AND (\sim X_1 OR \sim X_2) AND (X_2)$</td>
<td>$x$</td>
</tr>
</tbody>
</table>
Solution U3.A3 – Syntax Diagram
Solution U3.A3 – Syntax Checker

Idea: int parseXY(String lkd, int pos)
- Parse XY at Position pos in String lkd
- Return value: Position in String after processing XY
- If the string lkd is not correct, a ParseException is thrown during execution

parseEmptyOrSubTree()

parseSubTree()

parseChildren()

parseNode()
public class LKD {
    // string parsing
    public static void parse(String lkd) throws ParseException;

    // parse helpers (entity parsing)
    private static int parseEmptyOrSubTree(String lkd, int position)
        throws ParseException;
    private static int parseSubTree(String lkd, int position)
        throws ParseException;
    private static int parseChildren(String lkd, int position)
        throws ParseException;
    private static int parseNode(String lkd, int position)
        throws ParseException;

    // atomic helpers (single character parsing)
    private static boolean isCharAt(String lkd, int position, char expected);
    private static int parseChar(String lkd, int position, char expected)
        throws ParseException;
}

Lösung von Lukas Beyeler
LKD.parse(String)

Parse a "Linksklammerdarstellung" (LKD) tree

- An empty tree is coded as ‘-’
- A node is coded with a capital character: A,B,C, …
- Successors following a father are separated by ‘,’ in a bracket: V(C1,C2)
- Empty subtree leaves at the end of the list of successors
- Omit empty list

```java
public static void parse(String lkd) throws ParseException {
    int position = parseEmptyOrSubTree(lkd, 0);
    if (position != lkd.length()) {
        throw new ParseException("expected end of string", position);
    }
}
```
LKD.parseChar() & LKD.isCharAt()

```java
boolean isCharAt( String lkd, int position, char expected )
{
    return (position < lkd.length()) &&
            (lkd.charAt(position) == expected);
}
```

```java
int parseChar( String lkd, int position, char expected )
    throws ParseException
{
    if(!isCharAt(lkd, position, expected))
        throw new ParseException("expected character " + expected, position);

    return position + 1;
}
```

Why do we need it?
LKD.parseEmptyOrSubTree(String)

```java
int parseEmptyOrSubTree(String lkd, int position)
  throws ParseException
{
  if (isCharAt(lkd, position, '-') )
    return parseChar(lkd, position, '-');

  return parseSubTree(lkd, position);
}
```

To deal with branches (OR conditions)
```java
int parseSubTree(String lkd, int position) throws ParseException {
    position = parseNode(lkd, position);

    if (isCharAt(lkd, position, '(')) {
        position = parseChar(lkd, position, '(');
        position = parseChildren(lkd, position);
        position = parseChar(lkd, position, ')');
    }

    return position;
}
```
LKD.parseChildren()

```java
int parseChildren(String lkd, int position) throws ParseException {
    for (position = parseEmptyOrSubTree(lkd, position);
         isCharAt(lkd, position, ',')
             position = parseEmptyOrSubTree(lkd, position) )
    {
        position = parseChar(lkd, position, ',');
    }

    return position;
}
```
int parseNode( String lkd, int position ) throws ParseException
{
    if( position >= lkd.length() )
        throw new ParseException( "expected a node", position );

    char ch = lkd.charAt( position );
    if( !Character.isUpperCase( ch ) )
        throw new ParseException( "invalid character " + ch, position );

    return position + 1;
}
HINTS U4
1. A growing stack
   - A possible implementation using arrays
   - (Interface is known but how it works is hidden in the background, depends on the programmer. We make a better implementation with lists later)

2. Ackermann Function
   - Recursion explodes—compiler is used for Benchmark-tests

3. Java Bytecode
   - Use javap -c to disassemble RecursiveAckermann
   - Find 5 lines of code in the disassembled Bytecode
   - Compare Bytecode from 3b with java code from 2c. What strikes you?
Hints U4.A1 - Stack

- Data structure
- Only the last element is accessed
  - last-in-first-out queue (LIFO queue)
- Always use: function stack for local variables and function parameters
**Hints U4.A1a-c**

- **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
Hints U4.A1d

- `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
Hints U4.A2

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 19,729 decimal places!!

\textbf{Computable but not primitive recursive!} We can program it using \textbf{while-loops} but not \textbf{for-loops}!!

What does this mean?
Hints U4.A2

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)

Computable but not primitive recursive! We can program it using while-loops but not for-loops!!

⇒ We cannot know the number of iterations before hand; it also grows extremely fast!
Hints U4.A2a

- A(2,1) calculated by hand
  - A(2,1) = A(1+1,0+1) = A(1,A(2,0))...

- Write down ALL steps!
Hints U4.A2b - Pseudocode

- 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 can not say in advance how deep the recursion is → use while instead of for-loop!
Exercise 4 - Iterative Approach Q2c

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

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

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

stack

4

7
Hints U4.A2c – Implementation

```plaintext
if n == 0 → result = m+1
else if m == 0 → push(n-1), push(1)
else push(n-1), push(n), push(m-1)
```

```
stack = [n, m]
stack.push(n)
stack.push(m)
m = stack.pop()
n = stack.pop()
```
Hints U4.A2c – Implementation

- Stack
  - The stack of part 1
  - The interface should NOT be modified

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

- I can not do part 1
  - Use java.util.Stack<Integer>
    you just need push(), pop(), size und toString()
  - If necessary: send me an e-mail
Hints U4.A3 – Java Bytecode

```java
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

HelloWorldApp.java

**Diagram:**
- **Java Program**
- **Compiler**
  - JVM
    - Win32
    - UNIX
    - MacOS

**Diagram:**
- **Compiler**
  - MyProgram.java
  - MyProgram.class
  - My Program
Hints U4.A3 – Java Bytecode

Method int f(int, int, int)
  0 iload_0
  1 iload_1
  2 iadd
  3 iload_2
  4 idiv
  5 ireturn

Method int g(int, int)
  0 iload_0
  1 iload_1
  2 iconst_3
  3 invokestatic #f
  6 ireturn
Hints U4.A3 – Java Bytecode

Method int f(int, int, int)

0 iload_0 \(a_0\)

1 iload_1 \(a_1\)

2 iadd \(a_0 + a_1\)

3 iload_2 \(a_2\)

4 idiv \((a_0 + a_1) / a_2\)

5 ireturn

http://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html
Hints U4.A3 – Java Bytecode

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

javap does not show private functions by default!

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
Have fun!