

Arbeitsblatt Debugger (BlueJ)

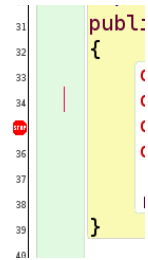
Du willst mit deinen Klassenkameraden verschlüsselt kommunizieren, ohne dass eure Lehrerin mitlesen kann. Wie ihr bereits wisst, ist die sog. Caesar-Chiffre, bei der jeder Buchstabe um eine bestimmte Zahl im Alphabet verschoben wird, aber einfach zu knacken.

Daher habt ihr euch für folgenden Algorithmus entschieden, der das Knacken erschwert:

1. Tausche den ersten mit dem letzten und den zweiten mit dem vorletzten Buchstaben.
z.B. wird aus „geheim“ → „miheeg“
2. Verschiebe jedes Zeichen im Alphabet um einen Schlüssel, der sich in Abhängigkeit von der Länge des Klartextes ergibt ((Länge des Klartextes modulo 4) +1).
z.B. wird aus „miheeg“ → „plkhj“
3. Wie du weißt, können chars durch ASCII-Zeichen und damit eine Zahl kodiert werden. Verschiebe jedes „gerade“ Zeichen um 6 (also b wird zu h, d wird zu j, ...)
z.B. wird aus „plkhj“ → „vrknp“

Aufgabe 1: Öffne die Klasse „*Nachrichtenaustausch*“ des gleichnamigen Projekts, das du unter ddi.cs.fau.de/schule/debugging findest mit BlueJ.

Indem du auf eine Zeilennummer klickst, kannst du sogenannte *Haltepunkte* (engl: *Breakpoints*) an beliebigen Stellen im Code einfügen:



→ **Setze einen Haltepunkt in Zeile 13 und in Zeile 16.**

Erzeuge nun ein Objekt *Nachrichtenaustausch* und führe die Methode *verUndEntschlüsseln()* mit einem beliebigen String als Eingabe aus. Es öffnet sich ein neues Fenster, der *Debugger*. **Finde heraus, welche Bedeutung die folgenden Schaltflächen haben** (Hinweis: Setze dazu auch weitere Haltepunkte).







Aufgabe 2: Beantworte nun mit Hilfe des Debuggers und passend gesetzten Haltepunkten folgende Fragen, jeweils für die Eingabe „geheim“:

1) Welchen Wert hat die Variable *schluessel* in der Klasse *Nachrichtenaustausch* nach Ausführung der Zeile 33 (`int schluessel = text.length() % 4`)? _____

2) Im Rahmen des Algorithmus zur Verschlüsselung werden Zahlen (integers) auf Buchstaben (chars) addiert. Beobachte mit Hilfe des Debuggers, was dabei passiert (z.B. Zeile 17 in der Klasse *Verschlüsselung*): _____

3) Wie oft wird die Alternative in Zeile 22 der Klasse *Verschlüsselung* betreten? _____

4) Beschreibe den zugrunde liegenden Fehler und behebe ihn.

Aufgabe 3: Offensichtlich funktioniert die Ver- und Entschlüsselung trotzdem noch nicht richtig. Finde mit Hilfe des Debuggers alle weiteren Fehler und behebe sie!

Fehler 1: _____

Fehler 2: _____

(Hinweis: Es bietet sich an, Haltepunkte nach jedem Schritt des Algorithmus zu setzen, um zu überprüfen, ob die einzelnen Schritte richtig durchgeführt wurden.)

Der Debugger

Mit Hilfe des Debuggers lassen sich folgende Fragen beantworten:

- _____
- _____

Das hilft mir bei:

- Kompilierzeitfehlern Laufzeitfehlern logischen Fehlern

So gehe ich dabei vor:

The image shows a screenshot of the BlueJ Debugger interface with several annotations in blue boxes:

- Haltepunkt setzen durch Klick am Rand**: Points to a red stop icon in the left margin of the code editor.
- Aktuelle Zeile ist hervorgehoben**: Points to the line `String eingabe = konsole.readLine();` in the code editor.
- Gehe eine Zeile weiter**: Points to the **Step** button in the debugger toolbar.
- Springe in die Methode, die in der aktuellen Zeile aufgerufen wird**: Points to the **Step Into** button in the debugger toolbar.
- Gehe bis zum nächsten Haltepunkt**: Points to the **Continue** button in the debugger toolbar.

The debugger window shows the following details:

- Options**: A tabbed interface for debugging options.
- Threads**: Shows `main (stopped)`.
- Call Sequence**: Shows `TicTacToe.spiele`.
- Static variables**: Empty.
- Instance variables**:
 - `private Spielfeld spielfeld = <object reference>`
 - `private String spielerName = "Wolfgang Händler"`
 - `private boolean spielerAmZug = true`
 - `BufferedReader konsole = <object reference>`
- Local variables**:
 - `String eingabe = "Wolfgang Händler"`

The toolbar at the bottom contains buttons for **Stop**, **Step**, **Step Into**, **Continue**, and **Terminate**.

Hinweise:

- Haltepunkt stoppt ____ Ausführung der Zeile, in der er gesetzt ist.
- In BlueJ werden die Haltepunkte beim erneuten Übersetzen entfernt.

