

## Exercices proposés / Vorgeschlagene Aufgaben

### Nim

Available program codes:

- Java: A class “Situation” managing a Nim game situation and a small class using the class as a demo. This Netbeans Project can be imported into Eclipse by simply import the src-Folder.
  - Study (and if necessary generate) the associated JavaDoc.
  - Try exploring the game tree.
  - Explore the game tree by storing the intermediate results of each configuration (using the provided methods to get a unique number for each Situation). Compare to the complete tree exploration
  - Implement the XOR-strategy in an additional method of the Situation class, returning an optimal move (if a such exists).
  - Implement a graphical interface for the game.
- JavaScript
  - Programm a computer oponent playing perfectly.
  - If the computer is in a loosing position, try to avoid situations where the human player can make a “symmetric” position.
- Ruby
  - If you have Ruby installed, play with it, study it, understand it.

You may also implement the game in your favorite programming language, of course.

### Connect Four

Available program codes:

- Java: VierGewinnt: A complete implementation of the Game including a simple GUI as a Netbeans Project (in Eclipse simply import the src-Folder)
  - Study (and if necessary generate) the associated JavaDoc.
  - Study and understand the BitField Implementation.
  - Study and understand the Alpha,Beta-pruning in Strategy.java.
  - Implement a different (most likely better) evaluation heuristics and test it.
- Java: JohnTromp: An efficient bit field implementation by John Tromp.
- C: Fhourstones: Completely solved the Connect Four game, see readme.txt.

## Simple Chess

- Ruby: matches: Not meant for distribution, not documented, just some personal hacks, used to generate the game tree graphics.

## Hermit Boxes

Inspired by <http://www.i-programmer.info/programmer-puzzles/203-sharpen-your-coding-skills/5924-hermit-boxes.html>

- Java: HermitBoxes: Not well commented. Simple implementation of the game, with bit fields and a simple tree exploration.
  - You might implement an evaluation heuristics
  - Implement alpha,beta-pruning (once you have an evaluation heuristics)
  - Store intermediate results (challenge: how to efficiently encode a board).