OOPD - u3

Emil Lagoni
Erik Allin
Peter Spliid

December 13, 2013

Kode

ComputerPlayer

ComputerPlayer

Opretter computerspilleren samt dens sværhedsgrad, som er en tilfældig integer mellem 0 (let) og svær (1).

makeMove

Styrer om computeren laver et smart eller et dumt valg i sit træk.
Hvis sværhedsgraden i computerPlayer er = 1 foretages der kloge smarte valg hele spillet igennem, hvis 0 så dumme valg.

smartMove

Definerer hvordan et smart valg tages.
Hvis tallet er et vilkårligt tal forskelligt fra en toer-potens - 1, så trækkes der en værdi, der får det tilbageværende antal kugler til at være lig en toer-potens - 1 (3, 7, 15, 31, 63).
Hvis antallet af kugler allerede er en toer-potens - 1, så tages der et tilfældigt lovligt tal.

avoidZero er med til at tage værdien, hvis det sidste var tilfældet. Dette gøres da computeren ellers ville trække 0 kugler, hvilket er et ulovligt træk.

WONT_GET_THERE er dels defineret fordi smartMove skal returnere no-

get, dels for at tjekke om der sker en fejl, der så kaster -1, som så vil være en ugyldig værdi.

dumbMove

Definerer hvordan et dumt valg tages.
Her tages der blot et tilfældigt, (random) lovligt antal kugler fra bunken.

Game

Game

Starter spillet med at generere en integer mellem 10 og 100, der angiver størrelsen på heapet.

printHeap

Printer heapets størrelse

getHeap

Gør at heapets størrelse kan blive kaldt/brugt.

isLegit

Tjekker om der bliver taget et lovligt antal kugler fra bunken.

remove

Fjerner det antal kugler fra bunken, som der blev tjekket i isLegit. Hvis tallet ikke er legalt bedes der om et nyt tal i det lovlige interval.

Nim

Main metoden.
Denne afvikler spillet ved at sørge for at spillet bliver startet, computer-spilleren bliver oprettet, der tjekkes for om nogen vinder, og spillet stopper i disse tilfælde. Derudover der scannes for det indtastede nummer, samt at der printes de nødvendige beskeder for at en bruger kan finde ud af spillet.

Overvejelser

Der er undervejs i opgaven blevet gjort flere overvejelser ud af hvordan vi får computeren til at holde styr på heapets størrelse.
Derudover har en implementation af sværhedsgrad el. dumbMove/smartMove også været noget der har givet mulighed for problemer.
Derudover var smartMoves valg af antal kugler der skulle trækkes også en udfordring.

Afprøvning

Der er blevet lavet en række forsøg i forbindelse med at fejlteste programmet.
Først og fremmest bliver det testet om computeren både kan være smart og dum i samme spil.
Dette ses hurtigt ved at computeren i nogle tilfælde altid går efter at der er de bedste tal (2 i x'te - 1) tilbage i bunken.
Derudover har computeren undervejs været 'forsynet' med et system.out kald, der printede om den var dum eller smart.
I forbindelse med dette har vi også testet hvad den smarte computer gør i det tilfælde af, at vi foretager det samme valg, her tages et tilfældigt tal, som også er hensigten.
Herudover er der kørt et stort antal spil (100+) for at være sikre på, at der ikke forekommer fejl hvis mængden i heapet bliver minut eller 0.
Dette har gjort, at vi i Nim-klassen har et tilfælde med (a.getHeap() == 1) og (a.getHeap() == 0), der henholdsvis tjekker om spilleren eller computeren har vundet.