INFO134 Klientprogrammering

Gruppeoppgave

Nim

Innlevering via mittuib: 19. mars, 10:00

Alt dere leverer skal være produsert av gruppens medlemmer, dette inkluderer HTML-kode, CSS-kode, og JavaScript-kode. Dere skal ikke bruke eksterne ressurser i løsningen deres.

1 Nim

I denne oppgaven skal dere skrive et spill. Spillet er en versjon av "Nim". Spillet er mellom to spillere og omhandler å ta klinkekuler fra en felles ressurs. Spillerene veksler mellom å ta én eller flere klinkekuler. Den som tar den siste klinkekulen vinner spillet.

Hver runde skal spilleren ta minst én klinkekule, og maks `maxGrab`. Variabelen `maxGrab` skal ha en verdi større enn 1, vanligvis 3, men kan generelt også være høyere. Hvis "Player 1" begynner spillet mot "Player 2" med standard `maxGrab` verdi (3) og 14 klinkekuler felles, kan et spill for eksempel foregå som følger:

1. "Player 1" tar 1 klinkekule.
2. "Player 2" tar 2 klinkekuler.
3. "Player 1" tar 1 klinkekule til.
4. "Player 2" tar 3 klinkekuler.
5. "Player 1" tar også 3 klinkekuler.
6. "Player 2" tar 2 klinkekuler. Nå er det 2 klinkekuler igjen. Derfor kan "Player 1" bare velge mellom å ta 1 eller 2 klinkekuler.
7. "Player 1" velger selvsagt å ta to klinkekuler og vinner spillet.

I oppgaven skal dere skrive (minst) to HTML-dokumenter. Et som forklarer spillereglene og et som har et JavaScript-program som lar brukeren spille spillet, enten to mennesker som spiller på samme maskin, eller et menneske som spiller mot en AI.

2 Struktur

Dere skal ha minst to HTML-dokumenter. Ett av dem `index.html` skal introdusere siden din til brukeren. Dere kan velge å forklare spillereglene her, eller velge å ha dette i et tredje HTML-dokument som dere lenker til herfra.

Disse HTML-dokumentene skal ligge i en mappe som også har to undermapper: `css` og `js`. Alle stilark dere skriver i forbindelse med dette prosjektet skal ligge i `css`-mappen. Tilsvarende skal alle JavaScript-filer ligge i `js`-mappen.

Dere skal skrive (minst) to JavaScript filer:

`nim.js` Her skal dere definere en konstruktør som oppretter et objekt som representerer et nim-spill.

`game.js` Her skal dere definere logikk som oppretter et `nim`-objekt, setter opp siden til å vise spillet til brukeren, og definere eventuelle andre funksjoner for å la spilleren interagere med spillet.

2.1 Nim-objekter

Dere skal skrive en konstruktør som heter `Nim`, definert med følgende parametre (i denne rekkefølgen):

`player1` Er et enkelt objekt med to egenskaper: `name` (av type tekst) og en sannhetsverdi `human`.

`player2` Som `player1`; angir navn og hvorvidt dette er en menneskelig spiller.

`victory` Er en *funksjon*. Når spillet er avsluttet skal spillobjektet *kalle* denne funksjonen og angi vinneren.

`total` Er et heltall som representerer antall felles klinkekuler i starten av spillet.

`maxGrab` Er et heltall. Dersom denne ikke angis som argument, skal det ha 3 som standardverdi.

Hver spiller må ha mulighet for å *endre tilstanden av nim-objektet* (enkelt å greit "ta noen klinkekuler fra fellesressursen"). Det kan være hensiktsmessig å definere en *metode* for dette. Når den nåværende spilleren tar et gyldig antall klinkekuler, må tilstanden selvsagt reflektere at det nå er færre klinkekuler i fellesressursen, men også at det nå er den andre spilleren sin tur.

2.2 Ulovlige verdier

Noen innstillinger for `Nim`-objekter skal ikke være tillatt. For eksempel skal `maxGrab` være minst 2. Hvis den er 1 eller mindre skal konstruktøren din krasje programmet ved å bruke `throw`-nøkkelordet. Dere kan velge å representere feilen med en tekst som beskriver feilen:

```
throw "`maxGrab´ må være større enn 1";
```

som dere kan lese mer om i Kapittel 5.6.5 (s. 105-106), eller opprette et feilmeldingsobjekt og gi det objektet en beskrivelse:

```
throw new Error("`maxGrab´ må være større enn 1");
```

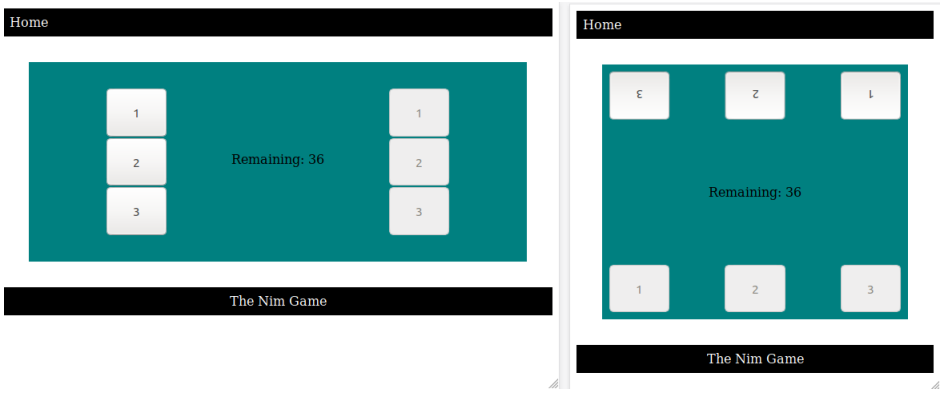
Dere velger selv hvor høy `maxGrab` kan være (utforsk hvor høyt deres måte å visualisere alternativene tillater), men dere må i det minste støtte verdiene 2, 3, og 4. Dersom konstruktøren kalles med for lavt tall (1 eller mindre) eller for høyt (her velger dere) grensen selv, skal dere rapportere feil som beskrevet over.

Dersom ett eller begge av spillerobjektene mangler `name`- eller `human`-egenskapen, skal dere rapportere feil. Dersom begge spillerobjektene sin `human`-egenskap har verdi `false`, skal dere rapportere feil. Dersom initiell total (antall klinkekuler) er mindre enn 12, skal dere rapportere feil.

3 Spill-dokumentet

Dokumentet som presenterer spillet til brukeren skal være *responsivt*. Når brukeren bruker en datamaskin med "stor" skjerm skal de to spillerene presenteres på venstre og høyre side av skjermen. I midten skal dere vise hvor mange klinkekuler det er igjen i fellesressursern. Dersom dere velger at hver spiller skal trykke på en av `maxGrab` knapper, kan det være lurt å organisere disse vertikalt (i en kolonne).

Når brukeren bruker en maskin med "liten" skjerm (f.eks. mobiltelefon) skal spiller 1 vises øverst og spiller 2 vises nederst, med antallet fellesressurser i midten. I så fall kan det være lurt å organisere hver brukers kontrollere (f.eks. knappene) horisontalt.



Dere kan bruke enkle knapper, eller lage grafikk for disse knappene. Eller dere kan velge en annen *brukervennlig* måte å la brukeren velge hvor mange klinkekuler hun skal ta.

4 Victory-funksjonen

Når spillet er avsluttet, skal `nim`-objektet kalle `victory`-funksjonen den fikk som argument til konstruktøren. Denne funksjonen får ett argument: objektet som representerer vinneren. Dere skal nå presentere denne informasjonen.

`victory`-funksjonen fungerer som en såkalt *callback*-funksjon, eller en tilbakeringingsfunksjon. Dere kan tenke på den litt som funksjonsargumentet vi gir til `setTimeout`-funksjonen, eller andre funksjoner vi bruker for å håndtere hendelser, men det er *dere* som må sørge for at funksjonen blir kalt når spillet er over.