

DAT101 HOVEDOPPGAVE

INNLEDNING

Hovedoppgave for DAT101 vårsemester 2023 består av to valgfrie oppgaver og kandidaten/gruppen må velge en av disse to oppgavene.

Oppgavene er basert på pensum fra vår/høstsemesteret 2022/2023, og gjennomgang av praktisk programmering. Oppgavene har omtrent samme vanskelighetsgrad, men arbeidsmengden er forskjellig. Det er med tanke på antall kandidater som er i gruppene. Hver enkelt oppgave beskriver hvor stor gruppe den er ment for, og dette vil det vektes for i sensuren av innleveringen. En gruppe på fire kandidater som velger en liten oppgave vil vektes strengere enn en gruppe på en til to kandidater som velger en stor oppgave.

Generelt for alle oppgavene, de blir gradert fra F til A i henhold til disse punktene:

- Applikasjonen/oppgaven må ikke være 100% løst for å levere eller bestå.
- Arbeidsmengde til gitt antall kandidater i en innlevering blir vurdert.
- Det legges ingen vekt på design eller stil.
- Hvilke elementer av oppgavespesifikasjonene er med.
- Koden trenger ikke å være optimal.

Grunnlag for underkjenning av innleveringen:

- Kodenotasjon avviker fra engelsk, det vil si det skal kun brukes engelske navn i objekter, klasser, funksjoner og variabler.
- Om det leveres flere identiske oppgaver fra to eller flere grupper, vil kommentarer til koden være avgjørende i sensuren.

Det er ingen begrensning på å legge klasser eller annen kode i eksterne Javascriptfiler (.js).

Kommentarer i koden kan være på norsk om ønskelig. Ikke over-kommenter koden, det vil si ikke kommenter unødig, bruk kommentarer til å fortelle om eventuelle problemer, eller nødvendig informasjon (se vedlegg om kommentarer).

Innleveringen tirsdag 9. mai 2023 innen klokken 14⁰⁰ en ZIP-fil med følgende innhold:

- Samme struktur som den som er vedlagt denne oppgaven i tillegg til egen kode.
- Eventuelt PDF eller Word -dokument om det er noe ytterligere dere ønsker å informere om

INNHALDSFORTEINELSE

Innledning.....	1
Innholdsforteinelse	2
Nedalsting av filer og veiledning til oppgavene	3
Oppgave 1 Snake	4
Beskrivelse	4
Ellemeter som skal være med	6
oppgave 2 Tegneprogram.....	8
Beskrivelse	8
Ellemeter som skal være med	9
Kode-kommentarer og notasjon.....	11
Kommentarer	11
Notasjon	12
Variabler.....	12
funksjoner	12
Objekter / Enum elementer	12
Klasser – attributter og metoder	13
forekomster	14

NEDALSTING AV FILER OG VEILEDING TIL OPPGAVENE

Alle filene til oppgavene finner dere på Inspira som tillegg til oppgaven. Her laster dere ned en zip-fil som inneholder hele mappestrukturen til alle oppgavene, uavhengig av hvilken oppgave dere velger. Det følger med noe kode til alle oppgavene. Koden som følger med er noe annerledes enn den dere har programmert i workshopene. Men prinsippet er helt identisk slik vi har gjort i alle oppgavene til modul fem.

Når dere leverer oppgaven på Inspira må dere lage en gruppe. En av dere velger en gruppe og invitere de andre til den gruppen med en nøkkel som personen får via Inspira. Om du er alene, så må du også opprette en gruppe, men da blir du eneste gruppemedlem. Om det ønskes individuell sensur internt i en gruppe, så må det spesifiseres og begrunnes i eget vedlagt dokument. Men fint om vi kan få beskjed på forhånd om det skulle dukke opp noe uforutsett i gruppen som kan ha konsekvenser for sensuren.

Faglærer og assistene kan veilede dere om dere står fast. Vi bistår i kode som ikke virker eller har problemer med. Og hjelper til med eventuelle løsninger. Men koden må skrives av kandidaten selv. Dere er tilgjengelige til å spørre om hjelp på Canvas, ved å sende beskjed til Arne-Thomas Aas Sønedeled. Alle spørsmål om pensum, kode, eller annen oppgaverelevant innhold kan bli postet i plenum slik at alle kandidater sitter med samme tilgang til lik informasjon.

Husk; ikke prøv å løse hele dilemmaet på en gang, gjør slik som vi har øvd på i workshopene. Analyser hva som skal gjøres å bryt det ned i mindre deler, da blir det mye enklere å komme i gang. Les oppgaven godt slik at dere ikke glemmer viktige elementer i løsningen deres.

Da er det bare å ønske dere lykke til alle sammen 😊

OPPGAVE 1 SNAKE

Denne oppgaven er mest egnet til grupper med en til to kandidater.

BESKRIVELSE

Det skal programmeres et spill som heter «snake». Spillet handler om en «slange» som beveger seg over skjermen med en gitt fart. Det er ikke mulig å stoppe slangen i å bevege seg. Men den kan styres opp, ned, venstre og høyre. Tilfeldig rundt på brettet kommer det mat til slangen som slangen skal spise. Hver gang slangen spiser så blir den lengre og lengre. Slangen kan ikke kollidere med seg selv eller veggene i spillet, da dør den.

Spillet kan pauses med Space- tasten på tastaturet, og startes igjen ved å trykke på «Resume» knappen.

youtube: <https://www.youtube.com/watch?v=hAz4bGzXPCc>

wikipedia: [https://en.wikipedia.org/wiki/Snake_\(video_game_genre\)](https://en.wikipedia.org/wiki/Snake_(video_game_genre))

Tutorial: <https://rembound.com/articles/creating-a-snake-game-tutorial-with-html5>

Alle elementene i veiledning som er linket til over, er med i starten av programmet. Det som ikke er med, slangen spiser ikke og heller ikke vokser.

Beskrivelse av filene i mappen til snake:

- Snake.html: Er html filen som kjører spillet
- script\bait.js: Inneholder en klasse som er maten slangen skal spise
- script\board.js: Inneholder to klasser som er selve spillebrettet
- script\game.js: Inneholder objekter og funksjoner som kjører spillet. Her ligger også «gameProps» som dere kan utvide med menyer, poeng etc.
- script\GLIB_2D.js: Er den samme filen som dere kjenner til fra workshopen, TPoint, TSprite etc.
- script\numbers.js: Inneholder en klasse som kan vise nummer ved hjelp av sprites.
- script\snake.js: Inneholder klasser som slangen er bygget opp av. Hode, kroppen og halen.
- snakeBodyDirection.js: Inneholder algoritmer som for kroppen til å vise riktig sprite i «svingene», det skal normalt ikke være nødvendig å endre noe i denne.
- media\SpriteSheet_Snake.png: Er bildefilen som inneholder alle sprites til spillet.

BESKRIVELSE AV SPILLEDESIGN

Eksempelkoden som følger med oppgaven kan det selvfølgelig bygges videre på. Det er allerede valgt et design i hvordan spille i kan fungere.

Det er designet et usynlig rutenett med rader og kolonner. Hver selle i dette rutenett har litt informasjon pakket inn i en klasse «TBoardCellInfo». Og forekomster av denne klassen er plassert i en tabell med rader og kolonner.

Hvert element som ligger på spillebrettet har en forekomst av «TBoardCell» som er den raden og den kolonnen som den befinner seg i.

Når slangen beveger seg over spillebrettet, så ender hodet til slangen rettingen som den hadde når den passerte akkurat denne sellen. Da kan de andre elementene i slangen som kropp og hale lese hvilken retning hode hadde da den var på samme selle. Og vi får da den effekten at kroppen «følger» etter hodet.

Vi kan også bruke dette designet til si noe om hva som befinner seg på hver selle. For eksempel om det er mat på akkurat denne sellen som hodet til slangen beveger seg over. Og slangen kan da «spise» maten.

SLANGEN VOKSER

Når man skal få slangen til å vokse for hver gang den spiser mat. Så kan man gjøre dette ved at hodet og alle kroppsdelene unntatt halen beveger seg ett steg. Da blir det et «mellomrom» som kan sette inn en ny kroppsdel. Det lure her er å «kopiere» den siste delen av kroppen før den flyttes, da havner den nye «kopien» akkurat der den skal med den rette informasjonen den trenger for å følge etter i neste steg.

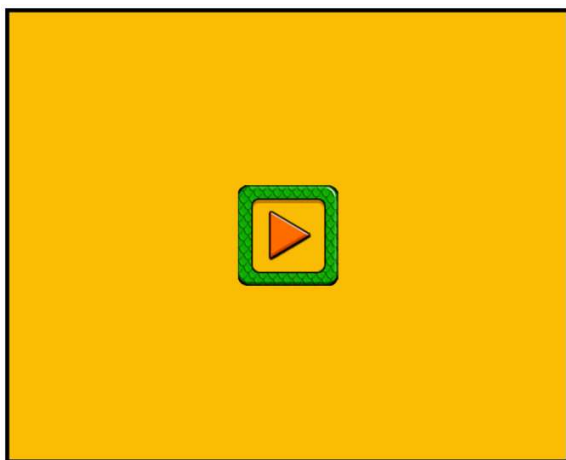
MENYENE

Det er ikke laget noen menyer i den medfølgende koden, så dette må programmeres selv. Se i «png» filen hvilke knapper og menyer som finnes, og om noen av knappene har for eksempel animasjon.

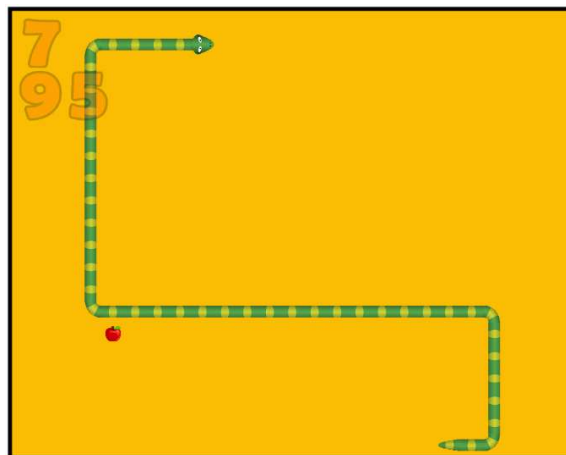
ELLEMETER SOM SKAL VÆRE MED

Oppgaven kan løses på andre måter enn den koden som medfølger oppgaven. Men det er viktig å følge oppgavebeskrivelsen slik at alle programmeringsteknikker og elementer i programmeringen er med.

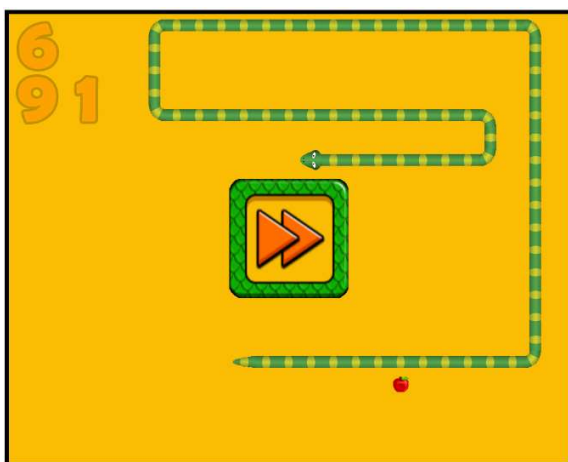
- a) Programmet skal ha et HTML definert canvas. Canvasets størrelse kan fastsettes i HTML delen, men eventuelle eventer skal programmeres i JavaScript.
- b) Det skal vises poeng heletiden under spillet, og poengene er avhengig av hvor lang tid det har tatt fra en matbit dukket opp til slangen har spist den. Jo kortere tid jo høyre poengsum.
- c) Slangens fart skal øke gradvis etter som den konsumerer mat.
- d) Slangen skal bli lengre etter som den konsumerer mat.
- e) Spilleren skal ha mulighet til å start, stoppe og pause spillet. Samt start et nytt spill uten å oppdatere nettleseren.
- f) Programmet skal ha globale variable, objekter og funksjoner. Funksjonsdefinerte objekter (Klasser), klassene skal ha både «Private» og «Public» medlemmer. Det skal ikke brukes kommandoen «class».
- g) Programmet må inneholde enumererte statuser, løkker og logiske sjekker.
- h) Klasse-struktur og navngivning står kandidaten/gruppen fritt til å velge selv, men det er viktig at notasjon og navngivning er entydig til objektets attributter og metoder. Samt alle navngitte variabler og konstanter skal være på engelsk.
- i) Slangens visuelle elementer skal bestå av sprites som lastes via et sprite-sheet.



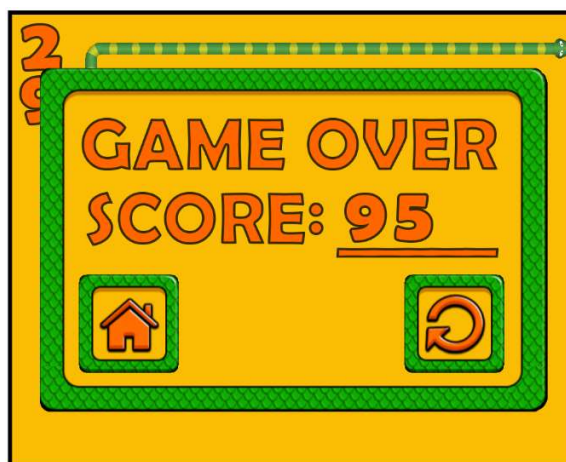
Figur 1: Førre spille starter.



Figur 2: Mens spillet pågår



Figur 3: Spillet er pauset



Figur 4: Spillet er slutt

OPPGAVE 2 TEGNEPROGRAM

Denne oppgaven er mest egnet til grupper med tre til fire kandidater.

Siden dette er en noe større oppgave en snake, og er beregnet på flere kandidater i gruppen, er det ikke gitt like mye tips til løsning i denne oppgaven.

Denne oppgaven baserer seg mest på logikk, tabeller og eventer. Samt det og å kunne lese allerede eksisterende kode. Alt av sprite(s) er allerede laget ferdig og det er ikke nødvendig å lage flere menyelementer.

BESKRIVELSE

Det skal lages et tegneprogram i JavaScript hvor brukeren kan velge et sett ferdig definerte figurer som firkanter, polygoner, sirkler, linjer og frihånd.

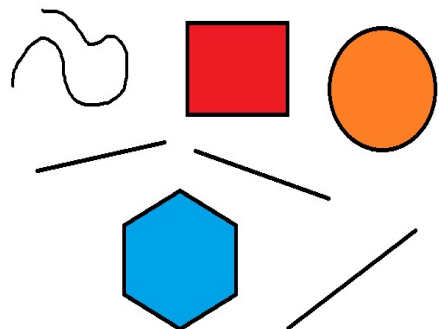
Programmet skal kunne sette fyllfarge, penn tykkelse og pennfarge før du tegner en figur eller linje. Det trenger ikke å være mulig å endre på et allerede tegnet figur.

Alle lukkede objekter som firkanter, polygoner og sirkler fylles med farge og omrisset tegnes etter hva brukeren velger. Når programmet starter opp skal det allerede være på forhånd valgt fyllfarge, penn tykkelse og penn farge. Slik at det skal være mulig å begynne å tegne med en gang.

Figurene tegnes med hjelp av musklikk, «mouse down» starter å tegne en figur og «mouse up» avslutter tegning av figuren.

I tillegg til å tegne, skal brukeren få mulighet til å slette en figur som er tegnet, det trenger ikke å være mulig å slette deler av tegningen slik som «viskelæret» fungerer i vanlige tegneprogrammer. Det skal også være mulig å flytte figurer opp eller ned på tegningsrekkefølgen, slik at brukeren kan velge hvilken figur som vises øverst.

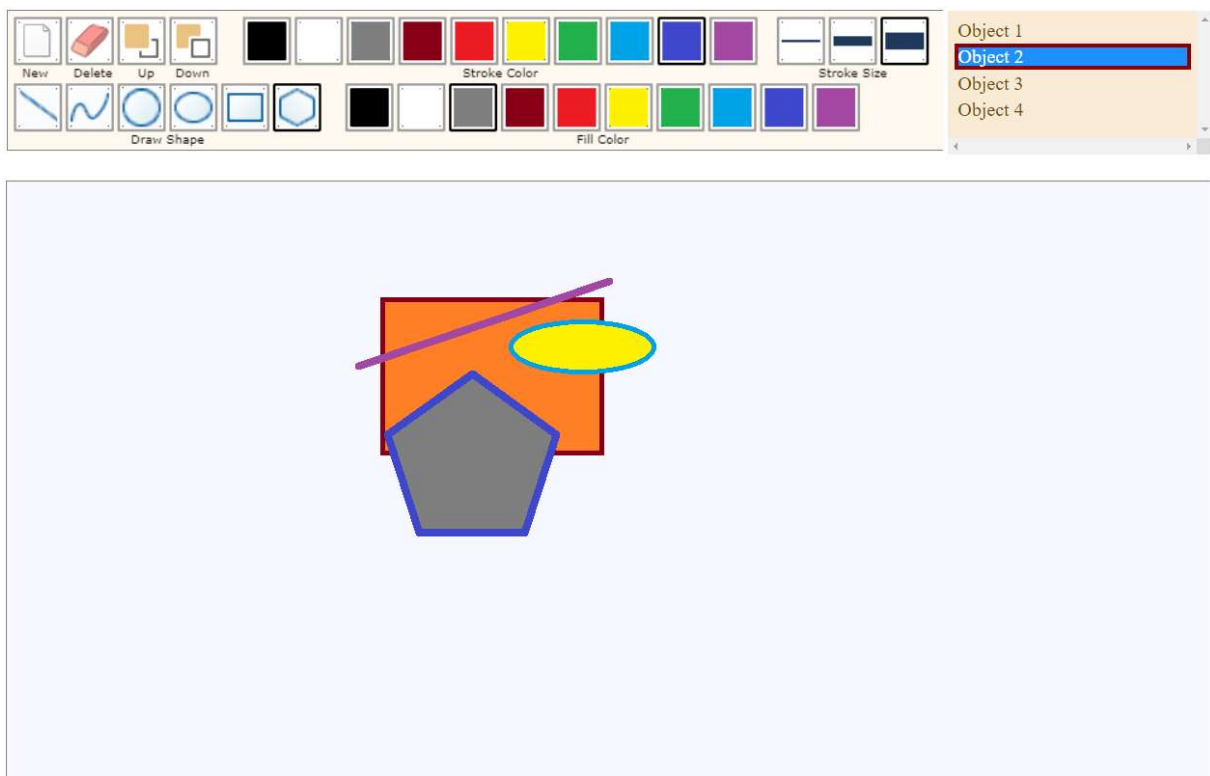
Programmet skal ha en form for meny hvor brukeren kan velge figurtype, sletteverktøy, farge osv. Brukeren skal også kunne «tømme» bildet og starte på nytt.



Figur 5

ELLEMENTER SOM SKAL VÆRE MED

- a) Programmet skal ha to HTML definert canvas. Canvas størrelse kan fastsettes i HTML delen, men eventuelle eventer skal programmeres i JavaScript. La ett Canvas være meny og det andre skal være Canvas som brukeren tenger i.
- b) Programmet skal ha globale variable, objekter og funksjoner. Funksjonsdefinerte objekter (Klasser), klassene skal ha både «Private» og «Public» medlemmer. Det skal ikke brukes kommandoen «class».
- c) Programmet må inneholde enumererte statuser, løkker og logiske sjekker.
- d) Klasse-struktur og navngivning står kandidaten/gruppen fritt til å velge selv, men det er viktig at notasjon og navngivning er entydig til objektets attributter og metoder. Samt alle navngitte variabler og konstanter skal være på engelsk.
- e) Menyelementer skal være sprites som hentes fra en sprite-sheet og tegnes i Canvas. Programmet begrenses til ti farger for fyll og penn, tre tykkelser på linjer (se sprite sheet)



Figur 6

Det følger med en del kode for styring av menyene, dere står fritt til å endre menyenes funksjonalitet eller bruke den som medfølger.

Fil beskrivelse:

- paint.html er html-filen som kjører Paint applikasjonen.
- paint.css er stylingen til applikasjonen.
- menu.js er den delen av koden som styrer menyene.
- SpriteSheet_Paint.png er bilde over alle menyknappene (sprites).

I oppgaveteksten følger ingen beskrivelse på hvordan den medfølgende koden fungerer, det er en del av oppgaven å klare å tyde den koden som medfølger, slik at det kan legges til ny kode for å få applikasjonen til å fungere.

KODE-KOMMENTARER OG NOTASJON

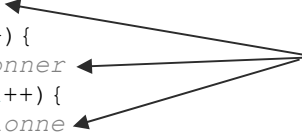
Dette vedlegget beskriver litt om kommentarer og notasjon i koden du/dere skriver.

KOMMENTARER

Det har ingen betydning om du/dere kommenterer på norsk eller engelsk.

Det skal ikke over kommenteres i koden, det vil si: ikke skriv kommentarer som er uvesentlige eksempel:

```
//Dette er en for-løkke for rader
for(let row = 0; row < rows; row++){
  //Dette er en for-løkke for kolonner
  for(let col = 0; col < cols; col++){
    //En selle for hver rad og kolonne
    ...
    ...
    ...
    ...
  }
}
```



Det er ikke nødvendig med kommentarer som dette

Her er det greit med kommentarer slik at vi kan se hvilke knapper som er i bruk. For eksempel kode 38 sier ikke så my om knappen men *//Up key* forteller hvilken knapp som brukes. I slike tilfeller er det best å bruke enum-objekter.

```
function keyDown(e) {
  const code = e.keyCode;
  switch (code) {
    case 38: //Up key
      ...
      break;
    default: //Everything else
      break;
  }
}
```

```
function moveHero() {
  // Need to check speed in order to NOT get hero "glued" to the bottom
  if(hero.hitBottom() && (this.hero.speed.y>0) ) {
    //if speed is positive and hero has passed the bottom of the canvas
    // set speed and gravity to zero in order to stop her from moving down
    hero.speed.y = 0;
    hero.gravity = 0;
  }
  if(this.gameOver === false) {
    hero.move();
  }
  hero.draw();
}
```

Dette er også bra kommentarer, ikke uvesentlige og de er beskrivende for funksjonaliteten til spillet.

NOTASJON

All notasjon skal foregå på engelsk. Det vil si, ikke bruk norsk språk i klasser, objekter, forekomster, variabler eller funksjoner. Navnet som du bruker, må ha en tilhørighet til hva den **er** eller **gjør**.

VARIABLER

Start variablene med små bokstaver og om navnet består av flere ord skill ordene fra hverandre med stor bokstav.

```
let cvs = null;
let ctx = null;
let canvasFrame = null;
let game = null;
let interval = null;
let frameNumber = 1;
```

FUNKSJONER

Start funksjonsnavn med små eller store bokstaver, men vær konsekvent. Noen foretrekker å ha storbokstav i starten på en funksjon for å skille variabelnavn og funksjonsnavn, jeg foretrekker å bruke liten bokstav i starten på både funksjoner og variabler.

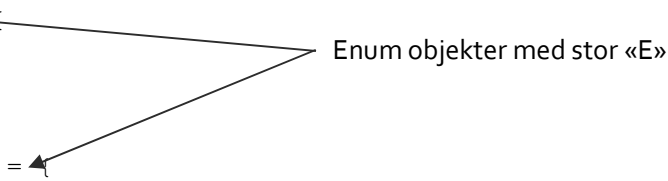
```
function getRandom(min , max) {
    return Math.ceil(Math.random() * (max-min)) + min;
}
```

OBJEKTER / ENUM ELEMENTER

Objekter som aldri endrer seg eller som brukes som «enum» elementer starter med stor bokstav. Alle attributtene til objektet har også stor bokstav. Hvis objektet blir brukt som «enum» element er det lurt å starte med stor E i navnet.

```
const EButtonState = {
    Up: 0,
    Down: 1
};

const EGameObjectType = {
    Hero: 1,
    Obstacle: 2
};
```



```
const Coordinates = {
  UserColorPic: [
    //Right side user color picker
    {x: 262, y: 87}, //Black
    {x: 262, y: 131}, //Blue
    {x: 262, y: 175}, //Brown
    {x: 262, y: 219}, //Green
    {x: 262, y: 263}, //Orange
    {x: 262, y: 307}, //Red
    {x: 262, y: 351}, //White
    {x: 262, y: 396}], //Yellow
  CorrectAnswer: [
    //Computer correct answer
    {x: 72, y: 46}, {x: 102, y: 46}, {x: 132, y: 46}, {x: 162, y: 46}],
};
```

Statistiske objekt som aldri endrer verdi mens programmet kjører.

KLASSER – ATTRIBUTTER OG METODER

En klasse starter med en stor bokstav, dere velger selv deres bokstav men dere er jo nå godt kjent med «T» og står fritt til å bruke den. Alle attributtene og metodene starter med små bokstaver. Om de er «Public» eller «Private» spiller ingen rolle for navngivningen.

```
function TRect(aX,aY,aWidth,aHeight){
  this.left = aX;
  this.top = aY;
  this.width = aWidth;
  this.height = aHeight;
  this.right = this.left + this.width;
  this.bottom = this.top + this.height;

  this.move = function(aDeltaX, aDeltaY){
    this.left += aDeltaX;
    this.top += aDeltaY;
    this.bottom = this.top + this.height;
    this.right = this.left + this.width;
  };

  this.setPos = function(aCenterPos){
    this.left = aCenterPos.x - (this.width/2);
    this.top = aCenterPos.y - (this.height/2);
    this.bottom = this.top + this.height;
    this.right = this.left + this.width;
  };

  this.mouseHit = function (aPos) {
    return !(( aPos.x<this.left) || (aPos.x>this.right) ||
      (aPos.y<this.top) || (aPos.y>this.bottom));
  };
}
```

FOREKOMSTER

Forekomster av en klasse starter med småbokstaver på lik linje med variabler, de skiller seg ut blant statiske objekter på grunn av at de endrer innhold etter som programmet går. Men navngivning er lik.

```
const surviveGame = new TGame();  
  
let canvasFrame = null;  
canvasFrame = new TRect(0,0,cvs.width, cvs.height);  
canvasFrame.thick = 10;
```