|  |
| --- |
|  |
| Eksamensopgave |
| Softwarekonstruktion 2 |

|  |  |
| --- | --- |
| Formalia: |  |
| Afdeling | AspIT Nordjylland |
| Underviser | Nils-Asbjørn Odinsholm |
| Censor | Jens Clausen |
| Forløb | Modulperiode 1, forår 2023 |
| Opgaven udleveres | Mandag d. 6. marts 2023 kl. 9:00 |
| Opgaven afleveres | Tirsdag d. 7. marts 2023 kl. 15:00 |
| Der arbejdes følgende dage i tidsrummet 9:00-15:00 | Mandag den 6. marts  Tirsdag den 7. marts |

Indholdsfortegnelse

[1 Overordnet beskrivelse 3](#_Toc128406389)

[2 Krav til systemet 4](#_Toc128406390)

[2.1 Registrering af data 4](#_Toc128406391)

[2.1.1 Spiller 4](#_Toc128406392)

[2.1.2 Scores 4](#_Toc128406393)

[2.1.3 Valideringer 5](#_Toc128406394)

[2.2 Oversigter 5](#_Toc128406395)

[2.2.1 Highscore 6](#_Toc128406396)

[2.2.2 Brugerdata 6](#_Toc128406397)

[2.3 Krav til opgaven 6](#_Toc128406398)

[3 Ekstraopgaver 7](#_Toc128406399)

[4 Aflevering 7](#_Toc128406400)

[5 Bedømmelseskriterier og evaluering 7](#_Toc128406401)

[5.1 Eksamen 8](#_Toc128406402)

[5.2 Karakterskala 9](#_Toc128406403)

Game Studio I/S

Game Studio I/S skal have lavet deres første spil.

|  |
| --- |
| Drawing Code |

# Overordnet beskrivelse

Du er hyret til at udvikle et simpelt Flappy bird klon. Du skal altså lave et spil hvor et element ”flyver” afsted, og stiger hver gang du trykker på en knap. Spillet skal stoppe når elementet styret af spilleren rammer en mur og derefter gemme scoren. Scoringsmetoden kan du selv vælge, et forslag ville være at den stiger med et point for hver mur du flyver forbi. Systemet skal kunne modtage et brugernavn ved starten til at oprette et objekt til at holde bruger data. Når spillet slutter skal det gemme scoren fra i en SQL-database. Det skal være muligt at åbne en highscore liste som viser mindst de højeste 10 scores.

# Krav til systemet

Overordnet set skal systemet kunne opfylde følgende krav. Kravene bliver uddybet i underafsnittene nedenfor:

1. Systemet skal kunne registrere brugerinput så det er muligt at styre spillerelementet.
2. Skal være sat op med et ordentligt designmønster, f.eks. MVC.
   1. Husk at gøre det vha. komposition
3. Indeholde spillere og scores
4. Systemet skal kunne vise

* Highscore
  + Brugeren der har sat highscoren
  + Hvad brugerens score var
* Spillets tilstand med konstant opdatering
  + Hvor spilleren befinder sig
  + Hvilke forhindringer der er på vej

Det vil være en god idé at læse dette afsnit helt igennem og løbende tage notater (evt. bare ved at skrive til og bruge overstregningstusch i dette dokument), inden du begynder at løse opgaven. På den måde sikrer du dig bedre at du ikke kommer til at lave noget der skal skrives for meget om senere.

## Registrering af data

I dette afsnit beskrives de objekter systemet skal kunne registrere. Der er fokus på sammenspillet mellem objekterne, ikke så meget attributterne. I et virkeligt system vil der blive registreret meget mere information på hver objekttype.

### Spiller

* Spilleren registreres med et unikt brugernavn.
* En spiller skal kunne holde en score.
* En spiller skal kunne holde antal gange han har spillet.
* Telefonnummer
* Email

### Scores

* Scores skal indeholde den mængde points der er opnået
* En foreignkey der linker til den spiller der har sat scoren.

### Valideringer

Systemet skal hjælpe brugeren med at sikre at der ikke indtastes forkerte oplysninger. Til det formål er der behov for nogle valideringer af systemet. Nedenfor er der givet et eksempel, i det endelige system vil der sikkert være behov for flere. Det er dog begrænset hvor mange valideringer der er tid til på 3 dage, så lav kun nogle få og overvej andre.

* Systemet skal ikke tillade to spillere af samme navn.

## Oversigter

For at få det fulde udbytte af registreringerne skal der laves nogle oversigter hvor man kan finde den information man har behov for.

### Highscore

|  |
| --- |
| Game high scores | Nitrome Wiki | Fandom |

Det skal være muligt at se en highscore liste i programmet.

### Brugerdata

Det skal være muligt at se sin egen brugerprofil. En bonus ville være hvis der også blev vist højeste plads på highscore listen sammen med højeste antal points.

### Kvalitetssikring

Du skal kvalitetssikre dit program gennem automatiseret unittest. Dette kunne f.eks. være en test af oprettelse af allerede eksisterende bruger i databasen.

## Krav til opgaven

Ud over at systemet skal understøtte den funktionalitet der er beskrevet ovenfor, er der nogle krav til hvilke metoder og teknologier der skal anvendes.

* I skal bruge det designmønster I har lært. (MVC)
* Systemet skal implementeres ved brug af objekt orienterede metoder, dvs. alle objekterne skal implementeres som instanser af klasser. Data skal gemmes i en database.
* Unittest skal implementeres til kvalitetssikring.

Det er ikke sikkert du kan nå at lave systemet helt færdigt. Opgaven er med vilje formuleret sådan at der bør være nok at give sig i kast med. Det er vigtigere at det du får lavet er godt end at du når det hele igennem. Samtidig er det dog også vigtigt at du viser at du mestrer de forskellige områder, dvs. lad være med at fokusere kun på én ting. Hvis du f.eks. har svært ved at nå at lave alle oversigter, så er det ok.

# Ekstraopgaver

Er du meget hurtig eller skal det være lidt mere udfordrende, kan du arbejde med følgende ekstraopgaver.

* Highscore søge funktionalitet: Overvej hvordan systemet kan udvides sådan at brugeren kan søge i highscoren efter et specifikt spillernavn.
* Bedre grafiske elementer: Tilføje billeder så du ikke bare er en flyvende kasse.
* Mere du selv kan finde på?

# Aflevering

Projektet samt relevante arbejdsdokumenter afleveres ved at uploade det til Github og sende et link til mig og til censor. HUSK at gøre det synligt for alle med link.

Afleveringstidspunktet er kl. 15:00 den sidste dag før eksamen (se eksamensavisen).

# Bedømmelseskriterier og evaluering

Der vil i bedømmelsen blive lagt vægt på følgende:

* Fuldstændighed – hvor meget er lavet færdigt? Fokus vil være på hvad du har nået, ikke på hvad du mangler.
* Struktur – anvendelse af relevante programmeringsstrukturer og teknikker, såsom anvendelse af 3-lags-model, variabler, properties og metoder.
* Robusthed – giver programmet relevante fejlmeddelelser?

## Eksamen

Evalueringen foretages som løsning af en applikationsopgave med en tilhørende database. Opgaven løses over 2 skoledage, á 6 daglige lektioner. Opgaven giver dig mulighed for at arbejde med:

* Centrale programmeringselementer
* Databaseprogrammering
* Kobling af brugergrænseflade til resten af systemet.

Eksaminationen gennemføres, uden foregående forberedelse, som en præsentation af den fremkommende løsning. Eksaminationen gennemføres over 30 minutter, inkl. tid til bedømmelse. Opgaven bedømmes efter 7 trins karakterskalaen og karakteren meddeles eleven.

## Karakterskala

|  |  |  |
| --- | --- | --- |
| Karakter | Betegnelse | Beskrivelse |
| 12 | Den fremragende præstation | Karakteren 12 gives for den fremragende præstation, der demonstrerer udtømmende opfyldelse af fagets mål, med ingen eller få uvæsentlige mangler. |
| 10 | Den fortrinlige præstation | Karakteren 10 gives for den fortrinlige præstation, der demonstrerer omfattende opfyldelse af fagets mål, med nogle mindre væsentlige mangler. |
| 7 | Den gode præstation | Karakteren 7 gives for den gode præstation, der demonstrerer opfyldelse af fagets mål, med en del mangler. |
| 4 | Den jævne præstation | Karakteren 4 gives for den jævne præstation, der demonstrerer en mindre grad af opfyldelse af fagets mål, med adskillige væsentlige mangler. |
| 02 | Den tilstrækkelige præstation | Karakteren 02 gives for den tilstrækkelige præstation, der demonstrerer den minimalt acceptable grad af opfyldelse af fagets mål. |
| 00 | Den utilstrækkelige præstation | Karakteren 00 gives for den utilstrækkelige præstation, der ikke demonstrerer en acceptabel grad af opfyldelse af fagets mål. |
| -3 | Den ringe præstation | Karakteren -3 gives for den helt uacceptable præstation. |

Tabel 2: Karakterskala for Softwarekonstruktion 3.