# Grundlæggende Programmering
# Projekt 2023: Mini-verdener

Dette dokument beskriver projektet for Grundlæggende Programmering (GRPRO) 2023.
(Det er delvist baseret på tidligere versioner af Anders Clausen.)

# 0 Indholdsfortegnelse

# 1 Formål

I projektet skal I *implementere* en simulation af liv. Hver uge vil I blive introduceret til et nyt tema med tilhørende krav og input-filer, som bygger videre på jeres tidligere løsning ved blandt andet at konstruere nye elementer til simulationen og modificere eksisterende dele.

Projektet består således af at *analysere* kravspecifikationen (domæneanalyse) og derefter *designe* og *implementere* et program som kan simulere forskellige former for liv og deres interaktion beskrevet i diverse krav og input-filer. I jeres løsning skal I *anvende* basale Java-konstruktioner som klasser, metoder, interfaces, abstrakte klasser, løkker, Collections mv. Jeres implementering skal dokumenteres ved brug af Javas dokumentationsfaciliteter. I skal hertil argumentere for korrektheden af jeres program ved brug af *systematisk testning*.

I skal udfærdige en rapport, hvori I *præsenterer* og *argumenterer* for jeres design- og implementationsvalg, samt *redegøre* for, om programmet virker som forventet, og *vurdere* i hvilken grad jeres testning af programmet understøtter denne konklusion.

Projektet involverer samlet set; *udvikling*, *test* og *dokumentation* af et Java-program, samt to *akademiske* rapporter, der beskriver jeres overvejelser, valg og beslutninger.

# 2 Projektbeskrivelse

Dette projekt omhandler at simulere en ”mini-verden”, der består af dyr, planter og andre aktører. Hver uge fokuserer på et nyt tema, der kontinuerligt bygger videre på interaktionen mellem de forskellige typer af aktører, der eksisterer i vores ”mini-verden”. Simuleringen anvender vores projektbibliotek (ITUmulator), der er beskrevet nedenunder og i den separate fil **ITU Simulator – beskrivelse af bibliotek**. Ved at bruge dette bibliotek opbygger I, over de kommende uger, en visuel simulering, hvor I vil kunne se de forskellige aktører leve deres bedste liv, konkurrere om mad og udvikle sig gennem simuleringen. Man kan lidt sige det er en hel livscyklus[1].

Temaerne er som følger:

- Uge 1: *Primitivt liv: Herbivore og Planter*
- Uge 2: *Fødekæder: Prædatorer, flokdyr og territorier*
- Uge 3: *Nedbrydning: Svamperiget*
- Uge 4: *Implementering af valgfrit dyr*

For hvert tema følger der en beskrivelse af de ***krav*** som løsningen skal understøtte. For hvert tema vil der også være nogle tilhørende input-filer som I skal indlæse og bruge til at populere forskellige simulationer. I kan dertil selv skrive flere input-filer. I vil selv have mulighed for at bestemme, hvordan de forskellige aktører vises I simulationen. Her refererer vi igen til filen, der beskriver projektbiblioteket.

## Krav

Kravene til jeres system er at det skal:

[1] Circle of life: https://en.wikipedia.org/wiki/Circle_of_Life

- **K1)** Opfylde de enkelte krav der stilles i hvert temas beskrivelse. **Kravene er dog formuleret som værende åbne, og det er derfor op til jer som gruppe at bestemme præcist hvad de betyder, og hvordan I fortolker dem.**
- **K2)** At jeres implementation er dokumenteret ved brug af Java dokumentation.
- **K3)** kunne indlæse en input-fil fra et vilkårligt tema og stadig fungere korrekt.
- **K4)** have et modulært design der gør det let at tilføje nye udvidelser (dette opbygger vi løbende undervejs i de forskellige temaer).
- **K5)** Det afleverede projekt kan kompileres og køres på andre computere.

### Projektbiblioteket (ITU Simulator)

Til projektet hører et Java-bibliotek, som har tilhørende Java-dokumentation og er kommenteret. Dette bibliotek har tre formål:

- **F1)** Det kan visualisere vores program via Java Swing (se eventuelt videoforelæsningen om Swing hvis du er nysgerrig på hvordan man bygger brugergrænseflader).
- **F2)** Det udstiller funktionaliteten til at køre en simulering.
- **F3)** Det udstiller abstraktionen om et 'kort' som vil være fundamentet for vores system.

Det forventes ikke at I forstår al koden i dette bibliotek, kun nok til at kunne interagere programmatisk med kortet og anvende simuleringen. For at lære at komme i gang med dette, se den tilhørende fil **ITU Simulator - beskrivelse af bibliotek**. Der skal i løbet af projektet ***ikke*** redigeres i dette bibliotek.

## 3 Regler

### Aflevering

Opgaven gennemgås ved forelæsningen **d. 23/11** og rapporterne, koden og video demoen skal afleveres senest **d. 21/12 kl. 14:00** gennem LearnIT (under eksamensaflevering & grupper).

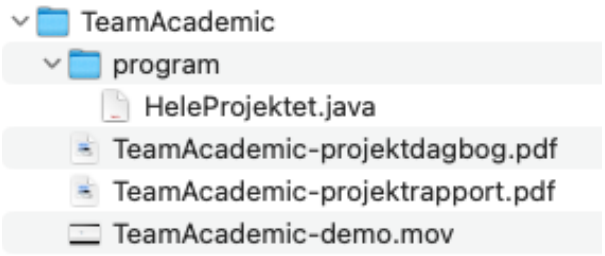
Det er jeres ansvar at aflevere til tiden. Hvis I afleverer for sent, kan I **ikke** gå til eksamen! Der er ingen fleksibilitet fra ITU's side, og det er ikke noget vi - som undervisere - kan gøre noget ved.

I bør derfor sørge for at gennemgå og uploade rapporter samt kodefiler mv. i god tid, for at sikre korrekt og rettidig aflevering. Når man afleverer på LearnIT skal man trykke 'videre' flere gange for at bekræfte, at man afleverer sit eksamensprojekt (indtil man har modtaget en bekræftelsesmail). Undlad at begynde afleveringen kl 13:58.

Ved aflevering af projektet bedes i lave en zip-fil med følgende struktur og navngivning

- **S1)** [gruppenavn -projektrapport.pdf]
- **S2)** [gruppenavn-projektdagbog.pdf]
- **S3)** [gruppenavn-demo] (Et vilkårligt videoformat der kan vises på mac og windows)
- **S4)** program (en mappe der indeholder al jeres kode, billeder, biblioteket. Sørg for at det kompiler)

her eksemplificeret med team navnet *TeamAcademic*:




### Grupper

Grupperne (af en størrelse på tre[2] studerende per gruppe) dannes som udgangspunkt af jer selv og skal være dannet før projektstart (Se undervisningsgang 21 og 22). Er der udfordringer i forhold til grupper bedes I henvende jer til Sebastian <sebni@itu.dk>.

### Programmeringsprog

Systemet skal udvikles i Java og anvende det udleverede bibliotek. I må gerne benytte et andet udviklingsmiljø end BlueJ (fx IntelliJ), men det er jeres ansvar at vide hvordan det bruges. Vi udleverer projektet til både BlueJ, IntelliJ, samt VSCode.

### Rapport

I projektet forventes det at I udarbejder to rapporter: **Projektrapporten** samt **projektdagbogen**. Projektrapporten har et omfang på maksimalt **6 sider** (inklusivt Figurer og forside). **Projektdagbogen** har et maksimalt omfang på **8 sider**. En side er 2400 tegn med mellemrum. I kan få mere hjælp til begge rapporter i dokumentet ***Udformning af rapporten***. Det forventes hertil at jeres program er dokumenteret ved brug af java dokumentation.

**Både projektrapporten og projektdagbogen skrives samlet I gruppen. Nyeste version af projektdagbogen skal sendes til jeres TA senest søndagen inden næste vejledning.**

***Rapporten skal udformes på enten dansk eller engelsk.***

**Forsiden skal opfylde ITUs krav** til en eksamensrapport (https://itustudent.itu.dk/Study-Administration/Exams/Submitting-written-work).

### Snyd

I må **ikke** benytte andres tekst, illustrationer eller programkode uden udtrykkelig eksplicit kildeangivelse. Plagiat kan resultere i udelukkelse fra eksamen (og potentielt bortvisning fra universitetet)!

## 5 Projektmaterialer

I forbindelse med projektet får I udleveret følgende materiale:

**M1)** Denne projektbeskrivelse (`this` dokument).

[2] Bemærk følgende: 2 ≠ 3 samt 3 ≠ 4.

**M2)** ”Udformning af rapporten” (dokument).
**M3)** ”ITUmulator” – beskrivelse af bibliotek (dokument) samt java dokumentationen.
**M4)** Basisprojektet (med projektbibliotek).
**M5)** Temaer for de fire uger (de er tilgængelige fra hver søndag), som indeholder både input-filer og en beskrivelse af temaet samt tilhørende krav (dokument).

# 6 Evaluering

Hovedvægten i evalueringen lægges på ***projektrapporten***. Rapporten har til formål at eksplicitere de beslutninger I tager undervejs, samt vise, at I kan reflektere over forskellige muligheder i forhold til både design og implementering af et større program. Det er således vigtigt at I begrunder jeres designvalg og vurderer færdighedsgraden af produktet. (Koden er naturligvis en nødvendig forudsætning for projektrapporten.)

De følgende afsnit skal være i **projektrapporten**:

**A1)** ”Introduktion” indeholdende en præcisering og afgrænsning af hvad jeres system understøtter ud fra de krav som stilles af de enkelte temaer.
**A2)** ”Design” indeholdende begrundede designbeslutninger i løsningsdomænet vedrørende systemet (klasser, interfaces, mv. samt relationer mellem disse). Dette inkluderer hele jeres system fra indlæsning af data til simuleringen er færdig. I skal ikke beskrive det medfølgende bibliotek.
**A3)** ”Testning” indeholdende en kort diskussion af hvorvidt systemet opfylder de opstillede krav, og i hvilken grad. Her er det vigtigt at argumentere for, hvorfor I mener dette ud fra jeres testning (hvis man faktisk *har* gjort dette).
**A4)** ”Konklusion” indeholdende et kort resume af de tre ovenstående afsnit.

Derudover skal I aflevere en **projektdagbog** som indeholder et nyt overafsnit for hver af de fire temaer. Denne skal for hver uge afleveres til den TA som vejleder jer (søndagen inden næste vejledningssession). Afsnittene i projektdagbogen skrives således undervejs.

**D1)** ”Resume” indeholdende en præcisering og afgrænsning af hvad systemet understøtter efter den enkelte uge.
**D2)** ”Design” indeholdende en beskrivelse og argumentation for de vigtigste designmæssige beslutninger I har foretaget og implementeret. Dette kan både være nye tilføjelser til systemet (som eventuelt også påvirker tidligere dele af programmet) eller modificeringer af dele af programmet fra tidligere.
**D3)** ”Testning” indeholdende en beskrivelse af hvordan I har tilgået testning af jeres system denne uge, og I hvilket omfang I har testet.

De afsnit I skriver i **projektdagbogen** for hvert tema bør anvendes som aktivt refleksionsværktøj i designprocessen. I skal se afsnittene for hvert tema som værende en form for ’dagbog’ over jeres beslutninger og vurdering af eget system som også hjælper jeres TA med at få indblik i, hvordan det går. **Det er vigtigt at disse dokumenter ikke beskriver kodenære beslutninger, dette skal I bruge Java dokumentationen til**. Således giver det ikke mening i rapporterne at beskrive, f.eks., de *if*-statements enkelte metoder indeholder mm.

# 7 Gode råd

Her følger nogle gode råd, som er 'uddrag' fra tidligere års projektbeskrivelse, skrevet af Anders Clausen.

## Divide'n'Conquer

Lad være med at lave alt på én gang. Forsøg at se på hvert problem som delproblemer, og notér jeres overvejelser, så I kan huske hvad I har overvejet og besluttet. Disse notater kan også fungere som et indledende skriv til rapporten.

Tag eksempelvis dette (fiktive) krav:

- "indlæs filen og tilføj herefter kaniner og ræve til simulationen afhængigt af filens indhold".

Frem for at løse det hele på en gang, kunne vi dele det op i brudstykker; f.eks.:

- "Jeg skal indlæse en fil, lad mig skrive noget kode til det…"
- "Jeg skal tilføje kaniner til simulationen…"
- "Jeg skal også tilføje ræve til simulationen…"
- "Nu skal jeg så sørge for at det er det rigtige dyr jeg indsætter afhængigt af hvad jeg læser fra filen…"

Når I når til det fjerde delproblem, så har I allerede de enkelte brudstykker ved at have løst de foregående tre, og deraf reducerer I opgavens størrelse indledningsvist.

## Start simpelt

Hvis noget virker meget uoverskueligt, så "løs et lettere problem først". Lav en løsning der er åbenlyst utilstrækkelig, men dog et skridt i den rigtige retning. Derefter er I meget klogere, og kan tage et nyt skridt (dette kalder vi en *iterativ tilgang* eller *trinvis forfinelse*).

Eksempelvis kan man starte med at implementere et dyr som ikke gør andet end blot at udskrive noget tekst, hver gang den skal "gøre noget". Det er tydeligvis ikke det opgaven beder om, men så er du I gang, og kan bygge videre.

## Behandl ikke rapporten som en eftertanke

Arbejdspapirer udfærdiget undervejs i projektet kan med fordel benyttes i rapport-skrivningsprocessen, så alle begrundelser for designbeslutninger og tests bliver reflekteret i den endelige rapport. Husk at bruge tegninger og tabeller – ikke kun tekst – når I skriver rapport. Rapporten skal kunne give et fyldestgørende overblik over jeres arbejde og tanker herom.

## Behandl ikke tests som en eftertanke

Husk at designe jeres program på en måde, så I kan teste dets funktion. Programmet kan nemmere testes såfremt I deler det op i logiske, velafgrænsede enheder – for eksempel ved at separere det at indlæse tekstfiler fra kørslen af simuleringen. Dertil er det vigtigt at jeres rapport kan kvalificere hvorvidt programmet opfører sig som I forventer, og som specificeres i kravene. Dette gøres bedst med et godt belæg, såsom en velovervejet testningsstrategi.

### Test inkrementelt

Det er en fordel at teste koden, efterhånden som I skriver den - på den måde er I bedst stillede til at identificere de interessante corner-cases som skal testes, og I får samtidig sikret jer, at færre fejl kommer til at påvirke andre moduler som programmeres sidenhen.

## 7 Inden afleveringen

Inden aflevering bør I sikre at alle krav, som er beskrevet i dette dokument, og de individuelle temaers beskrivelse, er opfyldt (gennemgå gerne dokumenterne flere gange).
I kan også tage udgangspunkt I denne tjekliste:

*Projektrapport*

- [ ] En *velskrevet* rapport i .pdf-format.
- [ ] Der er en logisk sammenhæng mellem overskrifter i alle afsnit.
- [ ] Der bliver ikke blandet mange forskellige emner sammen i de enkelte afsnit (her kan underafsnit bruges).
- [ ] Der beskrives ikke ting, der først introduceres senere i rapporten.
- [ ] Rapporten er *velargumenteret* og kan læses *selvstændigt* fra programkoden (f.eks. kan det ikke forventes at læseren ved ting på forhånd, der *skal* beskrives begrundelser for jeres valg).
- [ ] Jeres programkode afspejler det I skriver om i rapporten.
- [ ] Rapporten indeholder de obligatoriske afsnit.

*Projektdagbogen*

- [ ] Indeholder en sektion for hvert tema med tilhørende underafsnit der beskriver det forventede indhold.
- [ ] Projektdagbogen opfylder også punkt 1-6 for projektrapporten.

*Demo*

- [ ] En video optagelse af hvordan systemet fungerer og kører.

*Kode*

- [ ] Java-kildeteksten til det udarbejdede softwaresystem (og tilhørende filer)
- [ ] Den vedhæftede version af projektet kan kompilere og køre på alle computere
- [ ] Irrelevante filer er fjernet (kompilerede .class-filer, IDE-projektfiler, .DS_Store, mv)