Lasse Vestergaard
Mathias Godske

Sebastian Thorup
Christian Thomsen

Budgetberegneren

Rapport

Indholdsfortegnelse

Forord

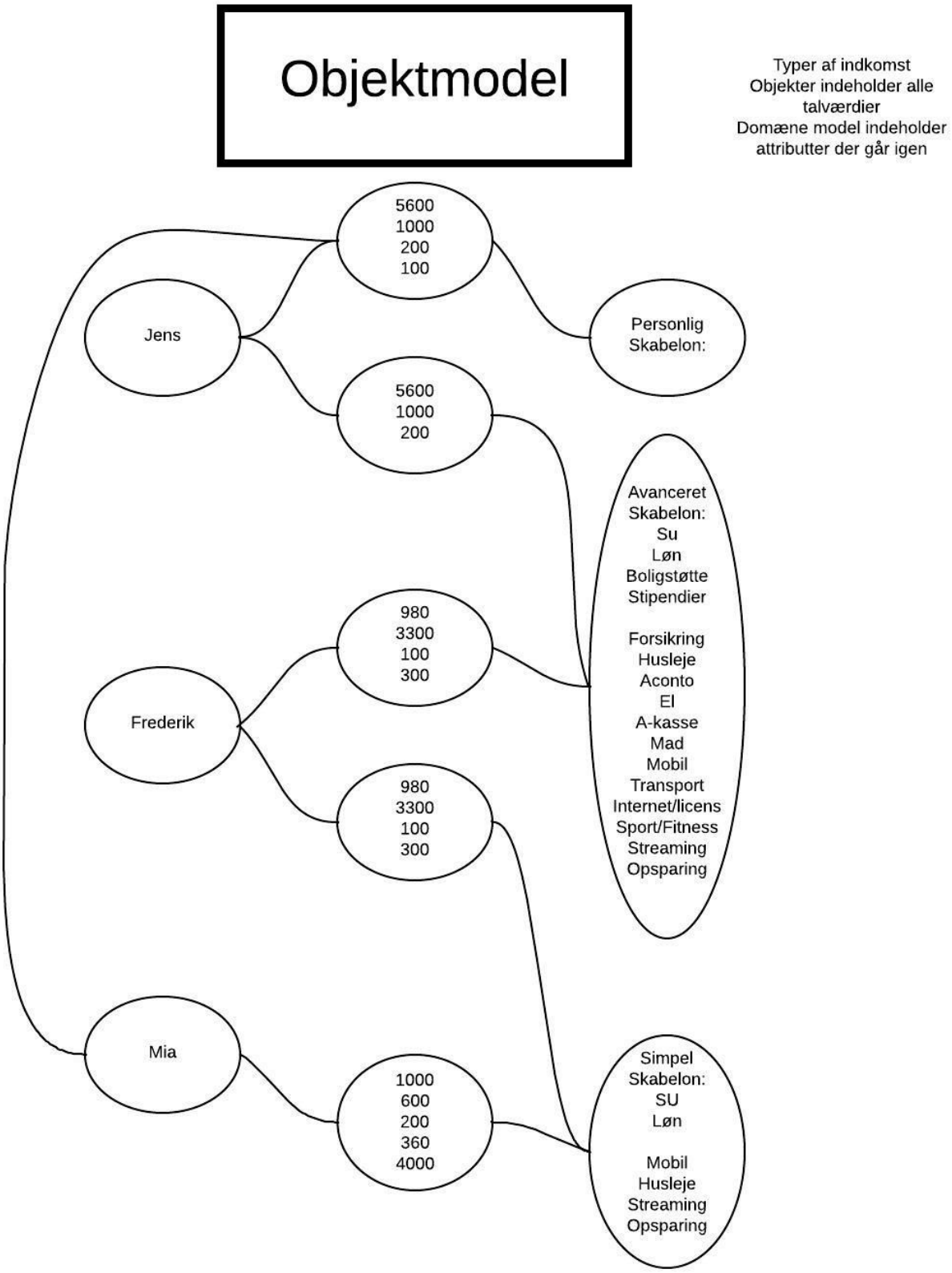
Som ung studerende er man fulgt af en travl og hektisk tid. Forberedelse til den kommende dag runger i hovedet, samtidig med at afleveringen til slutningen af ugen presser på. Når et glas ikke kan rumme mere, flyder det over. Dråben? Privatøkonomi. Regninger, husleje, A-kasse, mm. Fylder i den studerendes hoved og skaber et unødvendigt stresspunkt der med ide og handling kan undgås.

Vi har derfor i samarbejde med Arbejdernes Landsbank, videreudviklet på ideen om en budgetberegner applikation der ligger fokus på unge hjemmeboende -og udeboende studerende. Programmet vil bringe nytte til den vante budget planlægger såvel som den uvante, og være med til at skabe et bedre overblik over ens budgetter. I de nedenstående afsnit vil du som læser få et bedre og mere detaljeret indblik i vores program.

Vi håber du vil finde glæde i den kommende læsning.

- *Lasse Vestergaard, Sebastian Thorup, Mathias Godske, Christian Thomsen*

Budgetplanlægning for alle; Den hjemmeboende, udeboende og parret.



Introduktion til afsnit

I den ovenstående model tager vi udgangspunkt i tre forskellige studerende, med tre forskellige situationer. Her bliver vi roligt introduceret til budgetberegneren som program og de forskellige budgetskabeloner som applikationen tilbyder. Eksempelvis vil skabelonerne blive præsenteret efter de studerendes situationer, med formålet om, at du som læser vil få en bedre forståelse for det praktiske og smarte i programmet.

Eksempel 1 - Frederik. Den hjemmeboende studerende med tanker om udflytning

I første eksempel møder vi Frederik. Frederik er en ung studerende, som har boet hjemme hos sine forældre hele sit liv. Han er nu nået til et punkt hvor han føler, at det at bo hjemme hos sine forældre ikke længere er særlig attraktivt, og har derfor valgt at søge efter et lejemål i form af en lejlighed eller et værelse. Det er derfor vigtigt, at han danner sig et overblik over sin privatøkonomi.

Ved hjælp af budgetberegneren har han udregnet sit nuværende budget og sit kommende budget. I sit nuværende budget har han valgt at bruge budgetberegnerens 'simpel skabelon', som hovedsageligt retter sig efter den hjemmeboende studerende. Her ligger de mest basale udgifter og indkomster som ses i modellen foroven.

I sit kommende budget har han valgt at bruge budgetberegnerens 'Avanceret skabelon', som retter sig efter den udeboende studerende. Her ligger de udgifter og indkomster som ikke ville ligge i 'simpel skabelon', da man som udeboende har flere udgifter og potentielt flere indkomster.

Frederik har altså ved hjælp af budgetberegneren og dens redskaber skaffet sig et overblik over sin privatøkonomi. Han er nu i en bedre position og vil gøre overgangen nemmere når der endelig skal flyttes ind i det nye bo.

Eksempel 2 - Jens. Udeboende studerende på vej til at flytte sammen med kæreste

I andet eksempel møder vi Jens. Ligesom Frederik fra eksempel 1 er han en ung studerende, men har det til forskel, at han er udeboende og har derfor et andet budget. Jens har dog været så heldig at løbe ind i Arbejdernes Landsbanks budgetberegner der tilbyder en nem og overskuelig måde at lægge sit budget. I programmet vælger Jens 'Avanceret Skabelon' og udfylder de poster der appellere til ham. Indkomster og udgifter. Programmet udregner herefter Jens rådighedsbeløb og har nu lagt budget for den kommende måned til glæde for Jens.

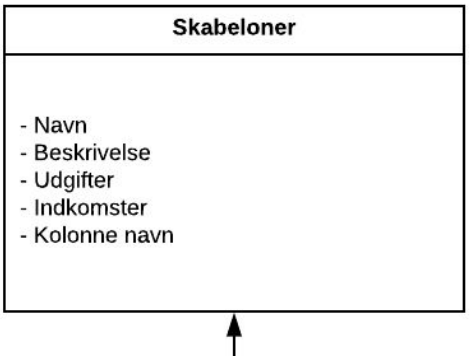
Jens har dog en glædelig nyhed. Han og kæresten Mia har fået deres helt eget lejemål, og skal nu flytte sammen. Jens tidligere budget kommer derfor til at være anderledes nu, og der er en chance for at der kommer nogle udgifter som ikke er at finde i budgetberegnerens 'Avanceret skabelon'. Det kan f.eks. være en udgift til møbler alene. Jens vælger derfor at bruge 'Personlig Skabelon', der giver en mulighed for at tilføje egne poster og skabe sit helt eget personlige skabelon. Ved brug af 'Personlig Skabelon' har Jens og Mia lavet et fælles budget og hjulpet dem godt på vej i deres flytning.

Eksempel 3 - Mia. Hjemmeboende studerende på vej til at flytte sammen med kæreste

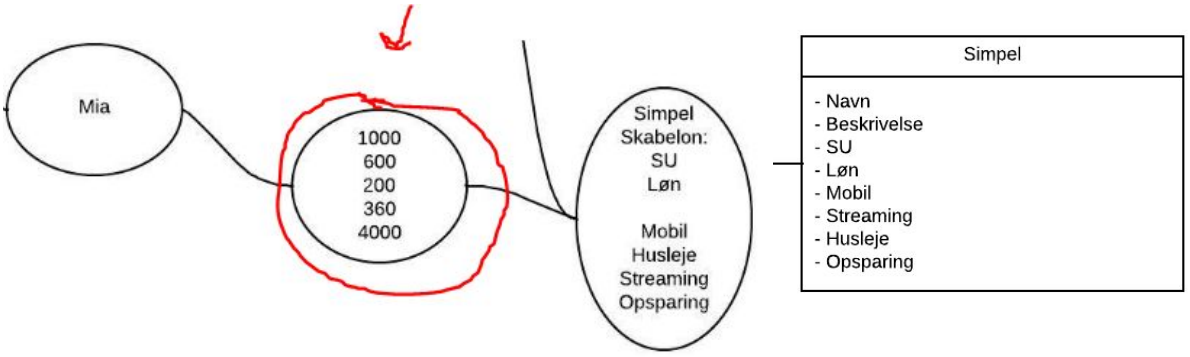
I tredje eksempel møder vi Mia, som er partneren til Jens. Mias situation er meget lig Jens, men med en enkelt forskel: Hun er hjemmeboende studerende. Mia har igennem Jens fået kendskab til Arbejdernes Landsbanks budgetberegner. Her bruger hun 'Simpel Skabelon', da hun som hjemmeboende ikke har særlig mange udgifter. Herefter bliver hendes situation identisk med Jens. De laver sammen et fælles personligt budget, hvor de tilføjer deres egne udgifter og indkomster. Budgetberegneren hjælper i sidste ende Mia og Jens med deres flytning.

Domænemodellen

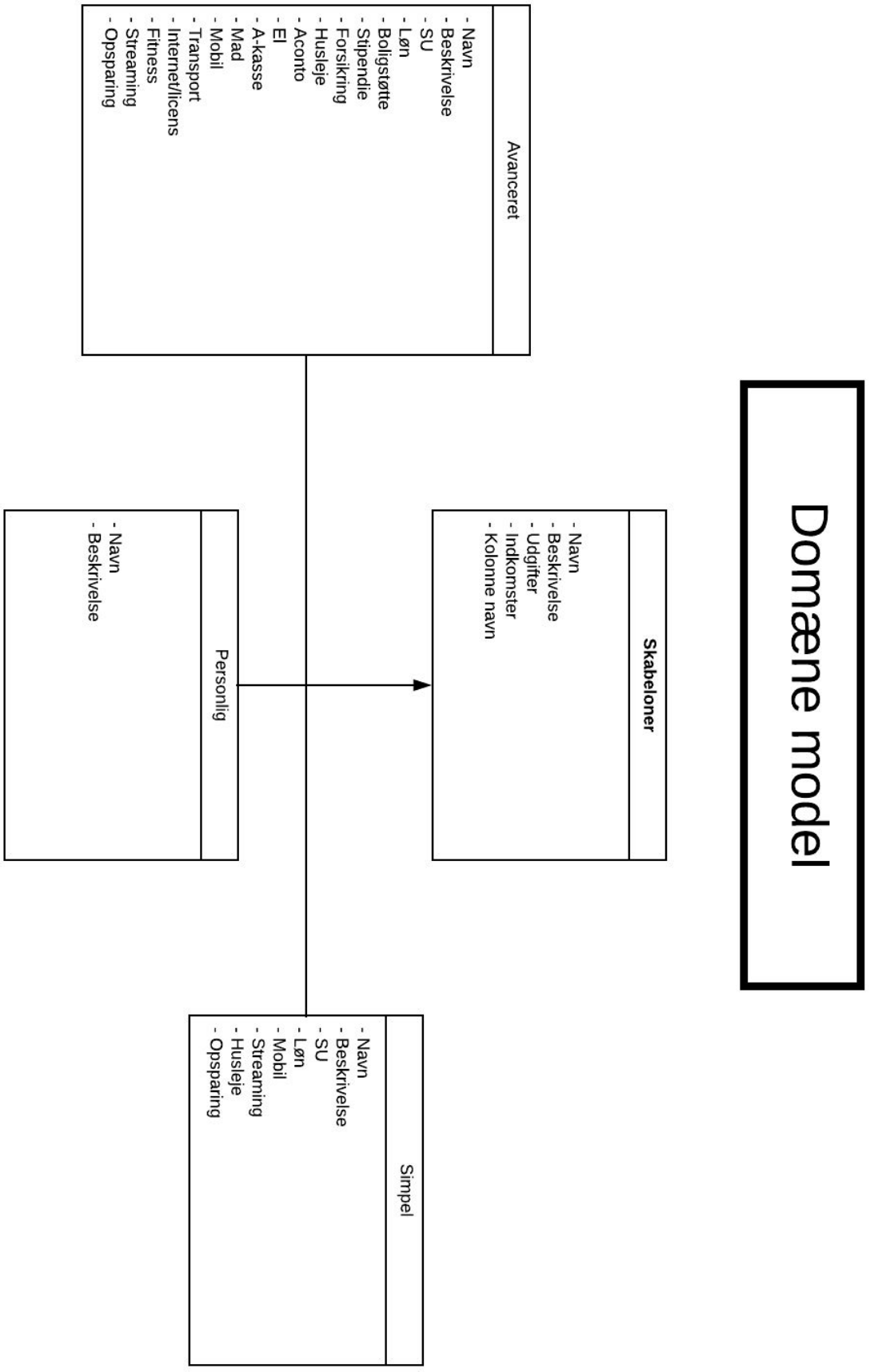
I dette afsnit kan domænemodellen findes. Den bruges i hånd med objektmodellen, for mere beskrivende at vise de enkelte objekter og hvad de indeholder. Dette hjælper med et overblik om man har det korrekte indhold med og om man evt. har for meget med og kan sortere noget fra. Ved at tage et kig på et udsnit fra domænemodellen kan det f.eks. ses hvad vi gerne vil have skabelon til at indeholde. Dette kalder vi for "attributter". Vi vil gerne have vores skabeloner til at have et navn og dernæst en beskrivelse på hvad for en skabelon det er. Udover det så kommer vores skabelon også til at indeholde udgifter og indkomster, enten prefix eller nogen selvlavede, som vil have bestemte kolonnenavne.



Kigger man på de bobler i objektmodellen der hænger på vores skabeloner, kan der ses en masse tal. Disse tal er værdier der evt. kunne findes inde i et budget, som man ville lave ud fra en skabelon, og hvad de enkelte tal betyder kan vi se nærmere på i domænemodellen. Med udgandspunkt i nedenstående udsnit fra domænemodel og objektmodel, kunne ens SU f.eks. være 1000 kr. dernæst en løn på 600 kr. og sådan fortsætter det ned ad.



Herunder kan den fulde domænemodel findes.


Domæne model
Skabeloner
- Navn
- Beskrivelse
- Udgifter
- Indkomster
- Kolonne navn
Avanceret
- Navn
- Beskrivelse
- SU
- Løn
- Boligstøtte
- Stipendie
- Forsikring
- Husleje
- Aconto
- El
- A-kasse
- Mad
- Mobil
- Transport
- Internet/licens
- Fitness
- Streaming
- Opsparing
Personlig
- Navn
- Beskrivelse
Simpel
- Navn
- Beskrivelse
- SU
- Løn
- Mobil
- Streaming
- Husleje
- Opsparing

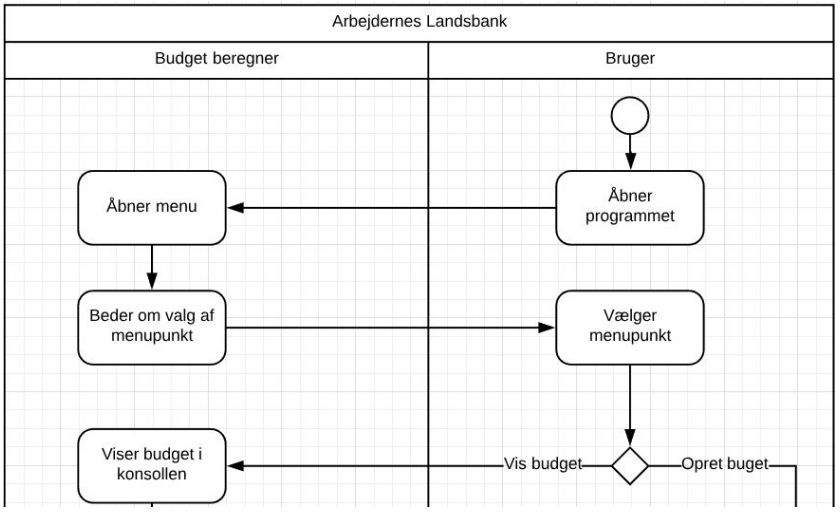
Brugervejledning

I denne del af vores rapport vil vi gerne snakke om hvordan man bruger selve programmet, og hvilke valg som en bruger bliver stillet overfor når de bruger budgetberegneren.

Visning af budget:

Det første man bliver spurgt om efter man har åbnet budgetberegneren er om man vil have vist et eksisterende budget, som er lavet gennem programmet, eller om man vil oprette et nyt budget.

Hvis man vælger at få vist sit budget så bliver brugeren spurgt om navnet på det budget som man vil have vist, hvis beregneren kan finde budgettet vil det blive vist i programmet, hvis ikke beregneren ikke kan finde det bliver man sendt tilbage til startmenuen med en fejlbesked som siger "Ugyldigt budgetnavn".

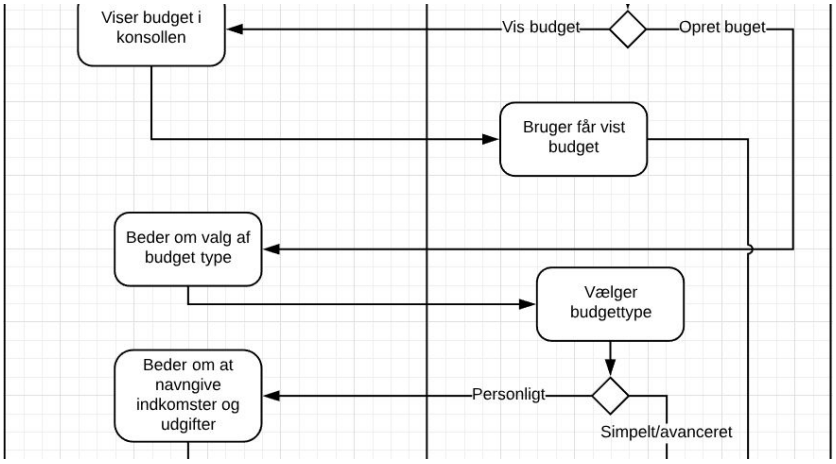


Redigering af budget:

Når man har fået vist sit budget så bliver man spurgt om man vil redigere i sit budget, siger man ja til det bliver man spurgt om man vil tilføje en indtægt, svares der ja til det bliver man bedt om at skrive et navn til den udgift og der efter en værdig, når man er færdig med at skrive sine indtægter bliver man spurgt om man vil tilføje nogle udgifter, igen svares der ja bliver man bedt om at skrive et navn og en værdig til sin nye udgift. Når man er færdig med at redigere i sit budget bliver man spurgt om man vil gemme sit nye budget og der kan man igen sig ja eller nej og hvis man siger ja, bliver man bedt om et navn til sit budget.

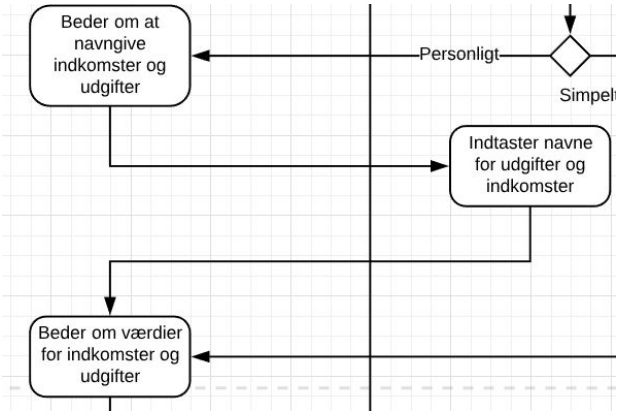
Oprettelse af budget:

Hvis man vælger at oprette et nyt budget bliver man derefter spurgt til hvad for en type af budget man vil lave, om det er et simpelt budget der passer godt til unge mellem 18 og 25 som stadig bor hjemme hos deres forældre samtidig med at de studere, et avanceret budget som passer bedre til unge mellem 18 og 25 som ikke er hjemmeboende men stadig studere og den sidste budget type er et personligt budget hvor man selv skriver indkomster og udgifter, her får du et blankt budget.



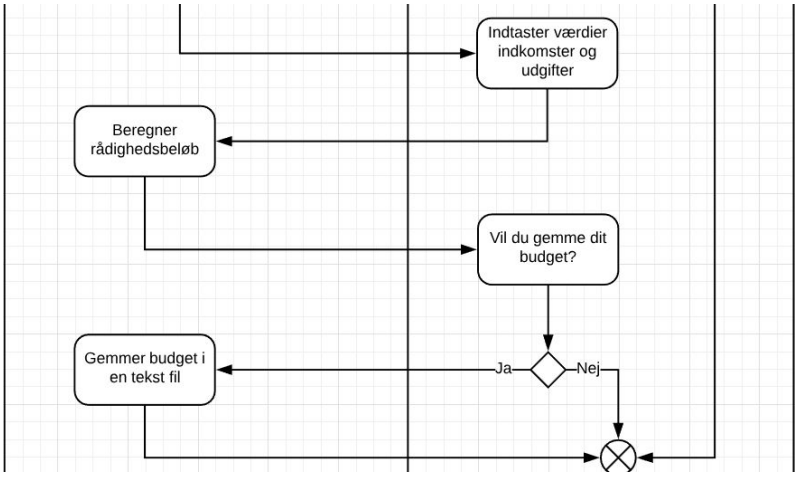
Oprettelse af personligt budget:

Når der oprettes et personligt budget skal man selv navngive sine indkomster og udgifter, da der ikke er nogle forudbestemte punkter så det første man som bruger bliver bedst om er, at man skal skrive navnet på en indkomst og derefter dens værdi, fx. su som navnet og så 2600 som værdien, når man vil skifte mellem navngivningen og værdig sættelsen skal man trykke enter. Når man har skrevet alle sine indkomster og vil til at skrive sine udgifter skal man bare, uden at skrive noget i navnet på en indkomst trykke på enter, så bliver man bedt om at gøre det samme for sine udgifter.



Færdiggørelse af budget:

Når man har fået skrevet alle sine udgifter og indkomster ind og der ikke er mere man vil tilføje til sit budget bliver man spurgt om man vil gemme budgettet til senere visning, hvis man vælger at man vil gemme budgettet, bliver man bedt om at navngive sit budget og når man har gjort det så bliver budgettet gemt på den pågældende pc, man får vist en besked om at budgettet er blevet gemt, og når det er sket bliver man sendt tilbage til budget menuen. Hvis man vælger ikke at gemme sit budget så bliver man bare sendt tilbage til budget menuen, hvor man igen frit kan gøre hvad man vil i programmet.

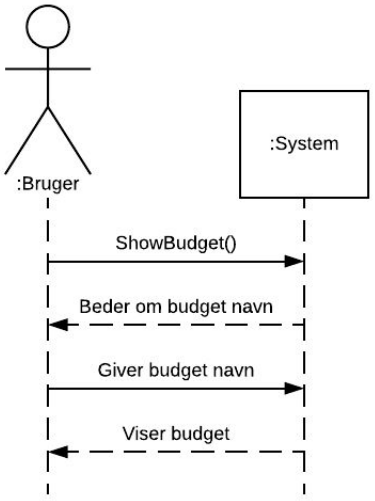


Hvorfor bruger vi SSD'er?

Vi bruger vores SSD'er til at identificere de handlinger der sker mellem system og person. Dette giver et godt overblik over hvordan vores funktion kommer til at fungere og om vi får det resultat i sidste ende vi gerne vil have. Det er også muligt at se om vi enten har for mange steps med til det vi prøver at opnå og om vi eventuelt kunne dele funktionen ud i flere dele for lettere overskuelighed, både for bruger og medarbejder.

SSD for ShowBudget()

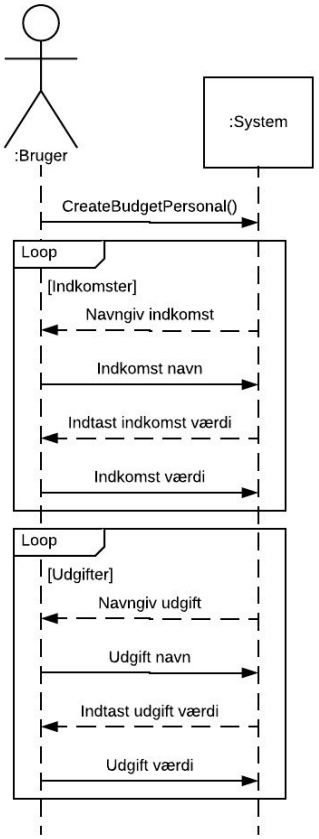
Nedenstående kan ses en SSD for vores ShowBudget() funktion, der indebærer at brugeren gerne vil have et budget vist og vælger derfor menupunktet i programmet. Programmet spørger så om hvilket budget brugeren gerne vil have vist og ud fra et tidligere budget kan brugeren få det vist i programmet.



SSD for CreateBudgetPersonal()

For denne funktion er funktionaliteten lidt større, men generelt er det en masse loops og samme procedure der foregår. Brugeren vælger at lave et personligt budget i vores program, hvorefter at programmet vil starte med at spørge om indkomster. Først angiver brugeren navnet på indkomsten og dernæst selve værdien for denne.

Når brugeren er færdig med at indtaste indkomster, går programmet videre til udgifter. Den vil igen spørge om navnet på en udgift og dernæst en værdi for udgiften.



System Operations Kontrakt (SOC: System Operation Contract)

En SOC er en kontrakt der præcist beskriver, hvad en enkelt funktion (systemoperation) i systemet skal gøre: hvilket input, der modtages, og hvilket output og resultat, der skal leveres.

Et SOC er relevant ift. kodning da SOC kan fortælle hvad det enkelte kode skal have af ansvar til funktionaliteten af programmet, og om den enkelte kode har for mange ansvar eller for lidt.

Dette kan også ses ud fra et SSD hvor hvert pil fra brugeren og indtil systemet kan laves et SOC ud fra, operationen skal have samme navn som inde i et SSD så der kan være sporbarhed.

Precondition i et SOC viser de tilstande og typer af data der skal være før systemoperationen kan gå i gang. Og post conditions viser de ændringer der er forårsaget af systemoperationen.

Operation: ShowBudget()

Cross reference: Usecase: Beregn og visualisering af budget.

Preconditions: Gået gennem menu.

Eksisterende Budget med kendt indtastet navn.

Postconditions: Ingen.

Output: Budget vist I konsol.

Operation: CreateBudgetPersonal()

Cross reference: UseCase: Opret personligt budget.

Preconditions: Gået gennem menu.

Postconditions: En Budget fil der indeholder et brugerbestemt antal indkomster samt udgifter og det udregnede rådighed er midlertidigt gemt indtil det er angivet svar til SaveBudget().

Output: Ingen ud over nuværende vist budget.

Operation: SaveBudget()

Cross reference: UseCase: Hent og gem budget.

Preconditions: En CreateBudget er instantieret.

Postconditions: En budget textfil er gemt i programmets folder.

Output: Budgettet bliver gemt.

Operation: CreateBudgetSimple()

Cross reference: Lav personligt budget.

Preconditions: Gået gennem menu.

Der findes en simpel skabelon.

Postconditions: En Budget fil der indeholder et default antal indkomster samt udgifter og udregnede rådighed er midlertidigt gemt indtil der er angivet svar til SaveBudget().

Output: Der er oprettet et simpelt budget.

Operation: CreateBudgetAdvanced()

Cross reference: Use Case: Opret Budget.

Preconditions: Gået gennem menu.

Der findes en avanceret skabelon.

Postconditions: En Budget fil der indeholder et default antal indkomster samt udgifter og udregnede rådighed er midlertidigt gemt indtil der er angivet svar til SaveBudget().

Operation: CalculateBudget()

Cross reference: Use Case: Beregn og visualisering af budget.

Preconditions: Instantieret budget.

Postconditions: Beregnet rådighedsbeløb tilføjet til budget.

Konklusion

Vores Budgetberegner giver brugerne en bedre og nemmere mulighed for at holde styr på deres penge i dagligdagen, uden at det koster dem ekstra. Vores budgetberegner er lavet til at kunne klare budgetter til alle typer mennesker uden der kommer til at være en masse tomme linjer som bare fylder og fjerne overblikket.