**Prosessdokumentasjon**

**Store valg**

**DeskTop**

Et krav for oppgaven var å bare levere en JFrame, derfor valgte vi å bruke flere internalframes til de forskjellige vinduene.   
Den ene JFramen blir nå brukt som et desktop-vindu med en meny øverst på siden. I denne menyen er det mulig å sette fokus/åpne, adde kontrollvinduer, avslutte programmet og lese noe informasjon om programmet.

**Vindusinndeling**

Vi valgte å dele inn programmet i 3 hoved deler(internalframes). En for kontrollvindu, som skal kontrollere heiskort og stå utplassert i bakkene. En for informasjonsvindu, som skal være ”nettsiden” og selgervindu, som både skal inneholde muligheten til å selge heiskort, og den statistikken som er nødvendig. Vi valgte å legge disse to siste sammen, for vi ville gjøre det så realistisk som mulig.

**GUI av Selgervindu**

Når vi lagde selgervindu prøvde vi å passe på at vinduet ikke innholdet for mye så det ble uoversiktlig, og var enkelt nok satt sammen til at en ”ansatt” eller en person med litt forkunnskaper skulle kunne bruke programmet raskt og lett. Siden skisenteret kun tilbyr 3 forskjellige produkter(korttyper) valgte vi å vise alle rett i vindu. Hadde det vært behov for flere vinduer ville det vært mulig og bytte ut panelet med registrering med et panel som inneholder knapper for flere produkter.

**Innlogging**

Siden selgevindu er ment for ansatte og eiere hos skisenteret mente vi at sikkerheten ved programmet kan være viktig. Vi lagde derfor 2 brukere, men ferdig passord og brukernavn. Passordet er lagd i en array som blir sammenligner inputen. Hvis admin logger inn settes en boolean til true, og et bilde vises på skjermen.

Når du logger ut vil også programmet skrive til fil.

**Start/steng skift:**

Vi valgte å legge inn en ”åpna/lukke skift”-knapp som åpner eller stenger skisenteret og styrer om alle transaksjoner kan gå gjennom. Sunn fornuft tilsier at et dersom skisenteret skulle være åpent, så burde det også være bemannet.

**Keycard og elektroniske billetter**

De største skisentrene i dag bruker et system, der brukeren har et fysisk kort (Keycard) og registrerer elektroniske billetter (Dags-, Klipp- og Sesongkort) på det. Vi valgte å bruke et lignende system for vårt program.

Selve Keycard’et inneholder bare en kort-id. Om noen skulle klare å ”hacke” seg inn på det fysiske keycardet vil de ikke kunne lage seg en egen ”gratis”-billett.

Informasjonen lagres i en database og vil være mye sikrere enn dersom billettene skulle lagres på keycard’et. Databasen vår er bare en lokal fil.

I tillegg, dersom en kunde som har betalt et sesongkort skulle miste keycard’et, kan det implimenteres slik at det mistede kortet slettes fra hele databasen, og kunde kan betale for et nytt keycard uten å miste hele sesongkortet.

**Registrere korttyper**

Å registrere en korttype krever at et keycard finnes i lista på skjermen. Hvis den er tom vil du få feilmelding. Registrering av korttypen har en default-innstilling, med 1 enkel voksenbillett. Dette kan endres på i vindu. Når du oppretter en korttype blir et keycard og korttypen send med til en handleliste som holder styr på hva kunde kjøper men lagrer ikke opplysningene i registeret siden kjøpet kan avbrytes. Handleliste vil bli mer forklart lengre ned. Totalsum, handleliste og keycardlista blir også oppdatert. Ved registrering av dagskort er det også en test som finner ut når kortet blir registrert. Hvis klokken er etter stengetid vil kortet bli aktivert neste dag, hvis kortet blir kjøpt midt på dagen får du en rabatt. Rabatten kan stilles på.

**Rabatter**

Vi har to hoved-rabatt grupper, skole og høytid. Hvis begge rabattene er valgt, vil den høyeste rabatten velges. I systemets pris-endring-metode finne det også muligheter for skisenteret til å gi tilbud på forskjellige kort.

**Handleliste**

For å forsikre oss om at kort og keycard som blir registrert, og ikke skal bli brukt likevel ikke skal lagre seg dobbelt og lignende feil, lagde vi en liste for handleobjekter(objekt som består av det vi sender til den når det opprettes: keycard eller korttype). Denne lista ble ikke lagret i noen registrere før betal-metoden gjorde det. Tøm-metoden har mulighet til å avbryte alt i denne listen.

**Visning av handleliste/kvitteringsvindu**

Siden det var behov for flere typer informasjon i den synlige handlelista bestemte vi oss for å bruke en JTable. Problemet vi fort møtte var at en vanlig JTable ikke støtter å legge til nye rader etter den er opprettet. Vi brukte derfor DefaultTableModel() som støtter denne funksjonen.

**Innlegging av korttyper til databasen**

I programmet har vi brukt hovedsakelig to typer lister: LinkedList og ArrayList.

Fordi ArrayList er generelt raskere å bruke og fungerer effektivt om vi setter inn elementer bakerst, brukte vi denne i metoder som krever hurtighet, men også ikke har en array som endrer størrelse. Vi brukte også ArrayList for Kontrollvinduene, men det ville ikke hatt noe særlig å si om vi brukte LinkedList istedenfor.

Siden LinkedList er litt tregere men likevel er veldig pålitelig når vi setter inn eller sletter et element midt i lista, brukte vi derfor hovedsakelig LinkedList i dataregisterene våres. Registere som: Keycard-, Kunde-, AktiveKort-, og HandleListe-register.

Dersom skisenteret hadde hatt registrert flere hundretusener skikort i databasen, ville vi måtte ha implimentert en database som ikke ligger på minnet. Så mange filer i databasen ville slukt maskinminnet, og programmet ville gått tregere. Men dette er ikke et reellt tilfelle for vårt skisenter, fordi skisenteret er såpass lite.

**Kundebehandling**

Siden systemet vårt tilbyr personlige kort er det viktig at personene som lagres i et register. Hvis feil skulle skje under registrering, en kunde skulle miste kortet sitt eller noen annet skulle være nødvendig lagde vi derfor et panel for kunde-metoder. Her kan du endre opplysninger som er feil eller utdatert, finne gitte kunder, eller slette de som ikke skal være i lista mer. Kundebehandlingen bruker registeret kundeliste til å lagre og lese inn kunder.

**Prisendring**

Vi syns det ville være lurt å gi admin ved skisenteret muligheten til å endre priser og antall klipp selv, og gjøre det lett. Vi la også med en gjenopprett-metode. Når prisene endres lagres alle prisene i Pris.java, sånn at når programmet starter henter den inn sine verdier for prisene der. Når du endrer prisene blir de nye verdiene sendt med til infovindu så prislistene der holder seg oppdatert.

**Statistikk**

Ved betaling blir først all ny data lagt til i registrene, også adminstat som er lagd for alle salgstall. Dette registeret blir lagret og lest fra ved oppstart.

**Dagsoppgjør**

Når en ansatt lukker skiftet og logger av ved stengetid lagres informasjonen om hva som er solgt osv ved den dagen. Hvis ved en feil en ansatt skulle logge av og på flere ganger i løpet av en dag vil kun den første lagringen skje, mens alt som blir registrert etter bli tatt med til neste dags oppgjør. For at dagene skal lagres, ligger alle dagene i et linkedlist i adminstat. Ved programstart blir det som ligger i denne listen lagt til i vindu.

For å se at programmet klarte å finne fra, add’e og lagre riktige verdier startet vi klassen adminstat med noen kunstige verdier. Verdiene er IKKE ekte eller realistiske, men er til for å vise at registeret fungerer som det skal. Hvis programmet skulle bli brukt som det er nå er det bare å fjerne kallet på én metode i AdminStat sin konstruktør og de kunstige verdiene vil ikke bli generert.

**Diagrammer**

Det er lagd en klasse som oppretter et panel med et diagram. Klassen får inn 3 verdier og setter størrelse på søylene etter disse. Denne klassen blir brukt flere ganger med forskjellige verdier.

**Script**

Scriptet oppretter et gitt antall Klippekort og Sesongkort. Antall klipp, Salgsdato og navn er tilfeldig generert. Om man ønsker kan man legge til en løkke som sjekker at alle kort du lager er gyldige, men vi syntes det var passende og realistisk å ha et keycard-register og kunde-register med ski kort som var ugyldige og gyldige.

**Kontrollvindu**

Kontrollvindu består egentlig kun av et lite panel og en sjekk. Sjekken bruker aktivekortreg for å avgjøre om kortet fortsatt kan brukes. Sjekken sletter også et klipp i klippekort når de blir brukt. Kontrollvindu har også en liten timer funksjon som skal simulere det velkjente grønne lyset en skikontroll lyser ved gyldig kort. Timeren lager en delay på 3 sek hvor lyset lyser(grønn=gyldig, rød=ugyldig). Når et lys lyser kan du ikke scanne inn nytt kort og får beskjed om det.

**Informasjonsvindu**

Infovindu er som sagt flere ganger lagd for en nettside. Den har ingen metoder men inneholder flere paneler men informasjon, bilder og en JTable med priser.

**Button-generator**

Vi lagde også en egen klasse som skulle gjøre det lettere å opprette knapper. Klassen mottar 4 parametere og lager knappen etter de, så sendes den ferdige knappen tilbake. Dette kortet ned det å lage en knapp fra 4 linjer til 1.

**Ansvarsliste**

Begge deltagere i prosjektet jobbet med metoder og oppbygning. Brukergrensesnittet ble skrevet for det meste av Martin. Sondre hadde litt mindre fokus på GUI og litt mer på mekanikk.

**Aktivekort**

Martin lagde første utkast av klassen, med listen, enkle liste metoder, og oppdater og sjekk metodene.

Sondre tok så over og forbedret deler av koden, lagde "avanserte" stringer, utvidet sjekker tok også ansvar for at klassen utvidet seg med programmet

**Kontrollvindu**

Lagd av Martin orginalt med da sjekk metoden og panelet.

Sondre stod for timer og begge to fikset opp i flere "feil" som kom etter at prorammet ble utbedret.

**Kunde**

Martin og Sondre (valgt ut variabler osv)

**Kundebehandling**

KundeBehandlingPanel metodene ble skrevet av Sondre og store deler av panelet, av Martin.

**Pris**

Lagd av Martin men satt i egen klasse og overført av Sondre

**PrisPanel**

Panel og metoder er lagd av Martin og sondre har flyttet og gjort alle endringer som følger med det

**SelgerMetoder**

Metodene er lagd av både Martin og Sondre. Martin har lagd registrering av korttypene, sletting, logg inn,dagsoppgjør og små metoder.

Sondre har lagd metodene for å lage keycard og fjerne dem, betal.

Begge to har også gjort endringer og forbedringer rundt om i alle metodene når det har vært nødvendig

**SelgerVindu**

Martin har lagd panelene. Sondre har knyttet klassene sammen og noen metodre.

Begge to har også gjort endringer og forbedringer rundt om i metodene.

**SkiSenter**

Denne klassen ble skrevet både Martin og Sondre, men hovedsakelig Martin, som oppretta DesktopPane, InternalFrame og MenuBar.