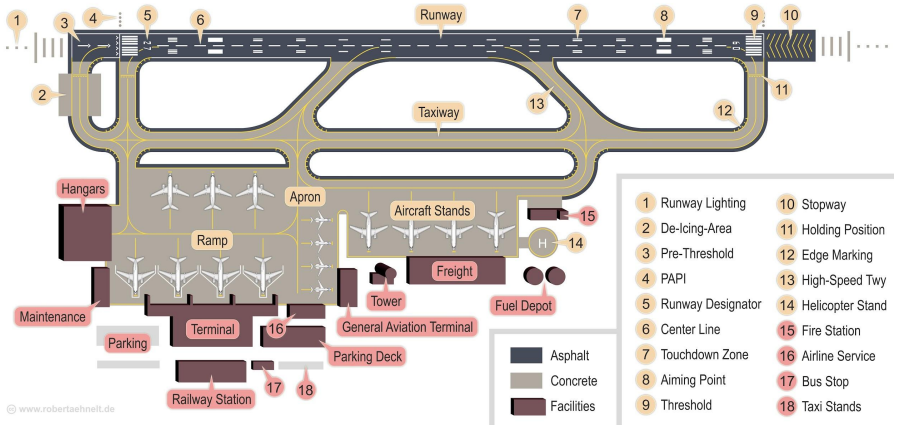
Fly, flyplass og flygeleder

Oppgave til repetisjon-seminar

Bakgrunn

I denne oppgaven skal vi jobbe med et system for Flyplasser, som skal kunne håndtere Fly-ankomster og -avganger. En flyplass er ofte hektisk, og det er mange avganger og ankomster på ulike flyplasser. Vi trenger å holde kontroll på Flyene, for å sørge for at det ikke blir fullstendig kaos og krasj. Dere har fått i oppgave å modellere Flyene og Flyplassene. Til slutt skal dere også lage en Flygeleder-klasse som dirigerer innkommende og utgående trafikk.

Det er viktig at Flyene er ordnet etter avgangstid, slik at Flyplassen har kontroll over når de ulike flyene tar av. Sagt med andre ord, en Flyplass må oppføre seg som en prioritetskø som sorterer Flyene etter avgangstid.



Bilde hentet fra: https://commons.wikimedia.org/wiki/File:Airport_infrastructure.png

Deler av oppgaven blir live-kodet under forelesningen, og dere skal fortsette i grupper etterpå. Det er veldig lurt å teste underveis, dere finner et testprogram og et kode-skall dere kan ta utgangspunkt i her.
Link: https://github.com/veronicanh/Flyplass

Klassen Fly

Et Fly er egentlig en flygning / flytur, og inneholder følgende informasjon:

- **`flyNummer`** (f.eks. `"DY3820"`)
- **`til`** (avreisested): En peker til Flyplassen flyet tar av fra
- **`fra`** (destinasjon): En peker til Flyplassen flyet skal reise til
- **`avgangstid`**: Et heltall mellom `0` og `23` som representerer klokketimen flyet har avgang

Variablene tas inn som parametere i konstruktøren.

Fly har pekere til flyet som står foran og etter i køen, de starter med verdien `null`:

- **`foran`**: En peker til Flyet som står foran
- **`etter`**: En peker til Flyet som står etter

Merk: Fly skal ikke være en indre klasse i Flyplass.

Metoder

Dere skal implementere en `toString`-metode, som skal gi informasjon om avgangstid, flynummer, fra- og til-Flyplass. Dere vil også få bruk for metoder som henter og setter instansvariablene, for eksempel for å hente flyet foran. Implementer de hent- og sett-metodene som dere har bruk for når dere jobber med oppgavene.

Comparable

Vi ønsker å kunne sortere Flyene etter avgangstid, slik at de kan stilles opp i en kø i samme rekkefølge som de skal ta av fra Flyplassen. For å få til dette må Fly-klassen implementere `Comparable`-interfacet, som har en metode: `compareTo`.

Repetisjon av reglene for `compareTo`-metoden:

< 0 (negativt)	hvis vårt objekt er "mindre" enn det andre
> 0 (positivt)	hvis vårt objekt er "større" enn det andre
= 0	hvis objektene er "like store"

☐ Oppgave 1: Skriv klassen Fly

Klassen Flyplass

En Flyplass har denne informasjon:

- **`sted`**: Beskriver lokasjonen til flyplassen, f.eks. `"Oslo"`. Tas inn som parameter i konstruktøren
- **`nesteAvgang`**: Peker til det neste Flyet som er neste avgang ved Flyplassen
- **`sisteAvgang`**: Peker til det siste Flyet som skal ta av

Metoder

Flyplassen må kunne håndtere at nye Fly ankommer (legge til Fly i køen), Flyet legges inn på riktig sted i prioritertskøen (sortert etter avgangstid). Den må også håndtere Fly-avganger (fjerne Fly fra køen), det første flyet i køen er det neste som skal ta av.

For å få til dette trenger vi en rekke metoder:

- `void` **`flyAnkomst`**`(Fly nyttFly)`: Et nytt Fly ankommer flyplassen, må plasseres på rett sted i prioritetskøen
- `Fly` **`flyAvgang`**`()`: Lar flyet som er `nesteAvgang`, altså første Fly i køen, "ta av" fra flyplassen

Vi trenger også disse metodene:

- `String` **`hentSted`**`()`: Returner stedet
- `int` **`antallFly`**`()`: Returnerer antall fly i køen
- `String` **`toString`**`()`: Vi ønsker å enkelt få oversikt over gjenværende avganger for dagen. Ta med sted og list opp alle Flyene i køen, en på hver linje

Det kan være lurt å implementere noen hjelpe-metoder som gjør de andre metodene enklere å skrive. Her er et forslag til hvilke som kan være nytte å ha med:

- `private boolean` **`skalForan`**`(Fly nyttFly, Fly nesteFly)`
  Sjekker om nyttFly skal plasseres foran nesteFly i køen
- `private void` **`plasserFremst`**`(Fly nyttFly)`
  Plasserer nyttFly fremst i køen
- `private void` **`plasserBakerst`**`(Fly nyttFly)`
  Plasserer nyttFly bakerst i køen
- `private boolean` **`plasserFly`**`(Fly nyttFly)`
  Plasserer nyttFly på riktig sted i prioritetskøen

☐ Oppgave 2: Skriv klassen Flyplass

Sjekke sisteAvgang- og foran-pekere

For å sjekke at `sisteAvgang` i Flyplass og `foran`-pekerne til Flyene er riktige skal dere implementere en metode som itererer over flyene i motsatt rekkefølge. Start med `sisteAvgang`, og følg `foran`-pekerne helt til starten av køen. Legg flyene inn i et Array etterhvert som dere itererer gjennom køen. Signaturen til metoden er: `Fly[] motsattRekkefoelge()`

☐ Oppgave 3: Skriv metoden motsattRekkefoelge

Enhetstester

Kjør `Hovedprogram.java` som dere finner her (versjon uten emojier her)
Link: https://github.com/veronicanh/Flyplass/tree/main/TesterMedEmoji%20😁

Merk at dere også trenger klassene `Enhetstester` og `FlyplassTester` for å kjøre programmet. Det vil utføres en rekke enhetstester for å sjekker de ulike metodene dere har skrevet i oppgavene over. Her er det viktig at dere har brukt metode-navnene som er oppgitt over. Alle testene må passere før dere kan gjøre neste oppgave.

☐ Oppgave 4: Passer alle testene i Enhetstest.java

Datastruktur-tegning

Få alle Enhetstestene til å passere slik at integrasjonstesten / brukstesten i `Hovedprogram` kjøres. Sjekk at utskriftene stemmer med det dere forventer.

Lag en datastruktur-tegning som viser strukturen slik den ser ut etter at metoden `tegneOppgave()` har kjørt.

- ☐ Oppgave 5: Lag datastruktur-tegningen

Flygeleder

Flygelederen sitter i toppen av flytårnet og dirigerer ankomstene og avgangene til flyene, og har til enhver tid full kontroll over alle flyene ved alle flyplassene. I denne klassen skal dere implementere en meny-løkke som gir brukeren en rekke valg:

1. Opprette et nytt Fly og la det ankomme på en oppgitt Flyplass
2. La neste Fly i køen ved en oppgitt Flyplass ta av
3. Skriv ut info om en oppgitt Flyplass
4. Skriv ut info om alle Flyplassene

0. Avslutt meny-løkken

Bruk en Scanner for å hente input fra brukeren i terminalen. Løkken skal fortsette å be brukeren om input helt frem til 0 skrives inn. (Om dere vil være ekstra fancy: avslutt løkken også dersom brukeren trykker enter).

Det er lurt å lage en metode for hver av de fire valgene brukeren har, på den måten kan dere holde koden i `menyLoekke`-metoden kort og enkel.

Tips: Her er bør dere bruke metodene som dere har skrevet i de andre oppgavene.

- ☐ Oppgave 6: Skriv klassen Flygeleder

Ferdig?

Gratulerer! 🎉

Løsningsforslag publiseres her i morgen kl 12:00, ta gjerne en titt på det!
Link: https://github.com/veronicanh/Flyplass/tree/main/Løsningsforslag

Ideer til utvidelser

Dersom dere likte denne oppgaven og vil jobbe videre med den, så har vi noen ideer til utvidelser dere kan legge til:

- [ ] La `Flyplass`-klassen implementere `Iterable`.
- [ ] Legg til mulighet for å lese inn `Fly` fra fil i `Flyplass`., dere finner eksempel-filer her. Link: https://github.com/veronicanh/Flyplass/tree/main/InputFiler
- [ ] Endre avgangstid i `Fly` til å være en kombinasjon av time og minutter.

- [ ] Reiserute fra `A` til `B`: Start på en flyplass og finn ut hvilke fly man må ta for å komme seg til en annen flyplass, spesielt dersom det ikke går en direkte-rute, f.eks. fra Oslo til Alicante.
- [ ] Utvidelse av punktet over: Hva er den korteste mulige ruten fra `A` til `B`?
- [ ] La flyene reise frem og tilbake mellom Flyplassene, og ikke bare en vei. Sørg for at avgangstidene gir mening.
- [ ] Sett opp en reiserute for hvilke fly man må ta dersom man ønsker å besøke *alle* flyplassene i nettverket, gitt en start-flyplass.

  Tips: Ta en titt på graf-algoritmer

- [ ] Simuler en dag i flytårnet: La fly ankomme og ta av til riktige tidspunkter automatisk. Dette kan også utvides til å inkludere flere flyplasser, slik at alle fly som har avgang kl 14 tar av samtidig, uansett hvilken flyplass de drar fra.
- [ ] Forslag til implementasjon for punktet over: Dette kan gjøres for eksempel ved å ha en egen klokke-tråd som går i bakgrunnen og holder styr på tiden (dere skal lære om tråder i uke 10).

Ha det gøy, og lykke til!

Spørsmål eller feil i oppgaven? Send mail til: veroninh@uio.no