DAT110 øving 9: Polymorfisme og enkelt GUI

Denne øvingen består av to deler med ulike læringsmål. Hver for seg er dette en-ukers øvinger. Til sammen blir de en to-ukers øving.

Del 1: Polymorfisme

Læringsmål: Du skal få øving i å bruke polymorfe metodekall

Beskrivelse av programmet

I denne delen skal du lage et enkelt spill. Spillerne starter på den ene enden av en vei som er 20 ruter lang. Hver runde kaster spilleren en terning og flytter så mange ruter framover som terningen viser. Hva som skjer deretter avhenger av hva slags rute spilleren havner i:

- Vanlig Rute: Spilleren flytter til ruta og turen går til neste spiller
- Tilbake til start: Spilleren rykker tilbake til start
- Hinder: Spilleren blir stående i ruta spilleren var i før terningkastet
- **Frivillig:** Ekstrakast: Spilleren får et ekstra kast og går videre framover. Husk at du må sjekke den nye ruta for hva som skjer med spilleren etter flyttet.
- **Frivillig:** Stå over tur: Som hinder med det tillegget at spilleren må stå over sin neste tur.

En spiller vinner om vedkommende kommer til eller forbi enden av veien.

Klassene «Spiller» og «Terning» er delt ut. Bruk «poengsum» egenskapen til spiller for å markere hvor langt spilleren har kommet på veien.

Deloppgaver

a) Start med å lage en klasse Rute for ei vanlig rute. Alle ruter skal ha en metode flytt_hit. Denne metoden skal ta et spillerobjekt som parameter. Den blir kalt når en spiller prøver å flytte til ruta. For ei vanlig rute skal flytt_hit flytte spilleren til denne ruta ved å oppdatere poengsum egenskapen til spilleren til å være posisjonen til denne ruta. Metoden skal også returnere en streng som inneholder en beskjed om at spilleren har blitt flyttet og til hvilken rute. Denne strengen skal skrives ut av hovedprogrammet (oppgave d) og e))

b) Lag deretter en klasse TilbakeTilStart for ei rute som gjør at spilleren rykker tilbake til start. Metoden flytt_hit skal sette spillerens poengsum til 0. Metoden skal også returnere en strengsom inneholder en beskjed om hvilken rute spilleren prøvde å flytte til (posisjonen til denne ruta) men at spilleren må rykke tilbake til start.

c) Lag en klasse Hinder for ei rute som stopper spilleren for denne runden. Metoden flytt_hit skal returnere en streng som inneholder en melding om at spilleren er hindret. Metoden skal ikke flytte spilleren (spilleren blir stående der han/hun stod sist tur).

d) Lag et Python skript som spiller spillet med én spiller. Lag en vei som ei liste med ruter. Fyll lista med en kombinasjon av de ulike typene rute. Det skal være flest vanlige ruter og færrest Rykk tilbake til start ruter.

e) Lag et Python skript som spiller spillet med flere spillere. Programmet skal avslutte når en spiller har vunnet. En spiller vinner når vedkommende kommer enten til slutten eller forbi slutten av veien.

f) **Frivillig**: Lag en klasse Ekstrakast for ei rute som gir spilleren et ekstra kast og flytter spilleren videre langs veien. Metoden flytt_hit skal gi spilleren et ekstra kast og flytte spilleren videre (kaller flytt_hit på ruta spilleren havner i etter ekstrakastet)

g) **Frivillig**: Lag en klasse StaaOverTur for ei rute som stopper spilleren denne og neste runde. Metoden flytt_hit skal ikke flytte spilleren (spilleren blir stående der han/hun stod sist tur) samt at metoden skal sørge for at spilleren ikke får kaste neste runde. Det er opp til deg å finne ut hvordan dette skal implementeres.

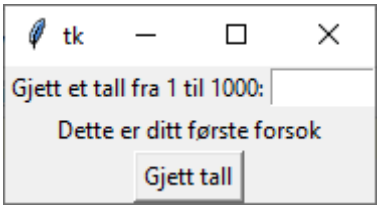
Del 2: Enkelt GUI

Læringsmål

Du skal lære hvordan sette opp et enkelt grafisk brukergrensesnitt. Du skal plassere komponentene og reagere på hendelser. Denne oppgaven baserer seg på videoene 79-86.

Beskrivelse

Gjett tallet spill: Oppgaven til spilleren er å gjette hvilket tall datamaskina har på færrest mulig trekk. Datamaskina har et tilfeldig tall mellom 1 og 1000. For hver runde skal brukeren skrive inn et tall i tekstfeltet og trykke «gjett tall» knappen. Programmet skal deretter skrive ut i en Label om brukerens tall er større enn eller mindre enn tallet programmet har, samt om det er nærmere eller lengre unna tallet enn forrige tall brukeren gjettet. Programmet skal også skrive ut hvor mange runder spilleren har brukt til nå. Et eksempel på utseende er vist under:



Oppgaver

a) Lag en klasse som tegner det grafiske brukergrensesnittet, som kan se ut som på bildet.

b) Lag en metode i GUI-klassen som leser ut teksten fra et tekstfelt og konverterer det til et heltall hvis mulig.

c) Lag en metode i GUI-klassen som skal kjøres når brukeren trykker på knappen. Metoden skal bruke metoden fra oppgave b). Hvis det brukeren skreiv kan konverteres til et heltall, skal koden sjekke tallet brukeren skrev mot tallet i systemet og skrive riktig beskjed til beskjedfeltet. Hvis det brukeren skreiv ikke kan konverteres til et heltall skal brukeren få opp en feilmelding.

d) Når brukeren skriver inn riktig tall (tallet datamaskina har lagd og som brukeren skal gjette) skal programmet vise en dialog «Du har gjettet riktig!» som også sier hvor mange forsøk brukeren brukte.

e) Legg inn kode i konstruktøren til GUI-klassen som lager et tilfeldig tall mellom 1 og 1000, setter opp spillet, legger til lyttere om det trengs, slik at programmet fungerer som beskrevet i «beskrivelse» avsnittet.

f) **Frivillig:** Etter at spillet er over skal spilleren få et valg om vedkommende vil spille på nytt. Hvis spilleren velger «ja» skal spillet startes på nytt. Spillet skal lagre en high-score og for hver omgang skal resultatet fra omgangen sammenliknes med high-score. Hvis nåværende

resultat er bedre (det vil si at spilleren brukte færre forsøk) skal spilleren få skrive inn sitt navn, som vises sammen med high-score.

g) **Frivillig:** Legg til en lytter på Enter-tasten som gjør det samme som når brukeren trykker på knappen.