COMPUTERSPIL 5

I denne femte og sidste delaflevering skal I bruge nogle af de ting, som I har lært om exceptions og fil-baseret input/output til at udvide programmet, så det kan optage et spil og gemme det på en fil, der så senere kan genindlæses og afspilles. Herudover skal I tilføje et tekstfelt, der lister de træk som brugeren laver under spillet. Hertil bruges *Decorator* designmønsteret, som gennemgås i kursets sidste forelæsning. Endelig skal I rette de fejl og mangler, som instruktoren har påpeget i jeres fjerde delaflevering.

Husk at holde jeres dokumentation opdateret, så den afspejler de ændringer/tilføjelser, som I laver i jeres kode. I behøver ikke at lave regression tests for ***Log*** klassen og jeres ændringer i ***GUI*** klassen. ***Log*** Klassen testes via testserveren, som beskrevet i opgave 1. Ændringerne i ***GUI*** klassen tester I manuelt, efterhånden som I laver dem.

Opgave 1

Implementér klassen ***Log***, der repræsenterer et gemt spil. Klassen har tre feltvariabler ***seed***, ***settings*** og ***choices***, der henholdsvis gemmer den initiale seed-værdi (for det ***Random*** objekt, der bruges i spillet), indstillingerne i ***Options*** dialogboksen og de valg, som brugeren laver (ved at trykke på byer med musen eller bruge piletasterne). Brugerens valg gemmes i en afbildning, der mapper fra skridt (heltal) til navnet på den by, der blev valgt i de pågældende skridt. Hvis brugeren trykker på flere byer i samme skridt, er det kun det seneste valg, der gemmes.

Klassen skal have en konstruktør på formen:

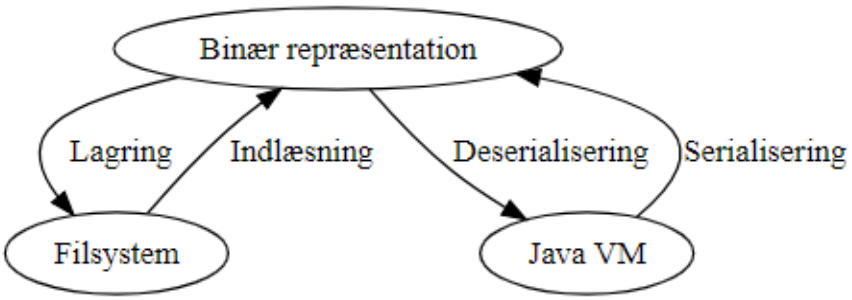
- ***Log(int seed, Settings settings)***

samt nedenstående metoder:

- ***int getSeed()*** // Returnerer den initale seed-værdi
- ***Settings getSettings()*** // Returnerer indstillingerne i Options dialogboksen
- ***String getChoice(int step)*** // Returnerer navnet på byen, der blev valgt i skridtet
- ***void add(int step, City city)*** // Brugeren trykkede på byen city i skridtet step

Den tredje metode returnerer ***null***, hvis der ikke blev trykket på nogen by i det angivne skridt.

I de næste opgaver skal I gemme et ***Log*** objekt i filsystemet og efterfølgende læse det ind igen. For at gøre det, skal der skabes en binær repræsentation af ***Log*** objekt. Heldigvis har Java indbygget funktionalitet til det formål. Nogle objekter afhænger af variabler i operativsystemet (f.eks. et id på en proces) og kan derfor ikke genskabes fra sin binære repræsentation. En klasse angiver, at den er *serialisérbar* (dvs. at dens instanser kan gemmes i binært format og genskabes herfra) ved at implementere ***Serializable*** interfacet, der er et *tagging* interface, hvilket betyder, at det ikke indeholder metoder, men blot angiver, at klassen er serialisérbar (se BlueJ-bogen side 526-527).



Lad ***Log*** klassen implementere ***Serializable*** interfacet.

I skal nu afprøve om de ting, som I har lavet i opgave 1, fungerer korrekt, men først skal nogle af projektets klasser opdateres, hvilket sker ved at I kalder klassemetoden ***download*** i ***TestServer*** klassen med parameteren "CG5".

Kald dernæst klassemetoden ***test*** i ***TestServer*** klassen med parameteren "CG5".

Det tjekkes at ***Log*** klassen implementerer ***Serializable*** interfacet og at konstruktøren/metoderne virker, som de skal.

Hvis testserveren finder fejl, skal l gennemgå jeres kode og forsøge at rette dem.

Opgave 2

I operativsystemer er filer og mapper angivet via en unik sti. I Windows systemer er harddisken, hvor operativsystemet er installeret, typisk identificeret ved ***"C:\"***. Windows er så installeret i mappen ***"C:\Windows\"*** og hvis brugeren ***phmadsen*** har filen ***picture.jpg*** liggende i mappen ***Images*** på sit skrivebord vil denne fil have stien:

- ***Windows**: "C:\Users\phmadsen\Desktop\Images\picture.jpg".*

I Linux og Mac OS er navngivningen lidt anderledes, og man bruger / som skilletegn i stedet for \:

- ***Linux**: "/home/phmadsen/Desktop/Images/picture.jpg".*
- ***Mac OS**: "/Users/phmadsen/Desktop/Images/picture.jpg".*

I opgave 1 i Computerspil 4 skabte I en ***Save log*** knap og tilføjede en ***ActionListener***, der blot åbner en dialogboks, der viser at knappen fungerer. I skal nu implementere den rigtige funktionalitet for ***Save log*** knappen, dvs. gemme ***Log*** objektet på en fil.

Når brugeren trykker på ***Save log*** knappen, skal der vises en dialogboks, hvori brugeren kan specificere, hvad filen skal hedde, og hvor den skal gemmes. Til dette formål har ***GUI*** klassen en feltvariabel, ***fileChooser*** (af typen ***JFileChooser***). Feltvariablen initialiseres i ***GUI*** klassens konstruktør, så det behøver I ikke at tænke på. Find klassen ***JFileChooser*** i Java API'en. I klassen kan I finde en metode, som tillader brugeren at åbne en dialogboks, der kan gemme en fil. Kald denne metode på ***fileChooser*** (som parameter kan I bruge feltvariablen ***mainFrame***). Metoden returnerer et heltal. Brug dette heltal (og en passende klassevariabel i ***JFileChooser***) til at tjekke, om brugeren accepterede den valgte fil eller lukkede vinduet uden at vælge en fil.

Hvis brugeren valgte en fil og accepterede denne ved at trykke på ***Save/Gem*** knappen, skal ***Log*** objektet, som I får fat i via metodekaldet

- ***game.getLog()***

gemmes på den valgte fil ved hjælp af en ***ObjectOutputStream***. I kan finde et eksempel på brug af ***ObjectOutputStream*** i BlueJ-bogen på side 527. For yderligere information, se Java API'en om objekt output streams og file output streams.

Hvis der opstår exceptions, skal brugeren gøres opmærksom på dette via en dialogboks.

Opgave 3

I opgave 1 i Computerspil 4 skabte I en ***Play log*** knap og tilføjede en ***ActionListener***, der blot åbnede en dialogboks, der viser at knappen fungerer. I skal nu implementere den rigtige funktionalitet for ***Play log*** knappen, dvs. indlæse et ***Log*** objekt fra en fil og dernæst udføre det spil, som ***Log*** objektet repræsenterer.

Når brugeren trykker på ***Play log*** knappen, skal der vises en dialogboks, hvori brugeren kan specificere hvilken fil, der skal indlæses. Dette gøres på tilsvarende vis som i opgave 2, men nu skal i åbne den fil, som brugeren vælger (i stedet for at gemme den) og indholdet af filen bruges til skabe et ***Log*** objekt. Dette sker i et antal skridt, der minder meget om skridtene i opgave 2. Hvis der opstår exceptions, skal brugeren gøres opmærksom på dette via en dialogboks.

Endelig skal loggen afspilles via metodekaldet:

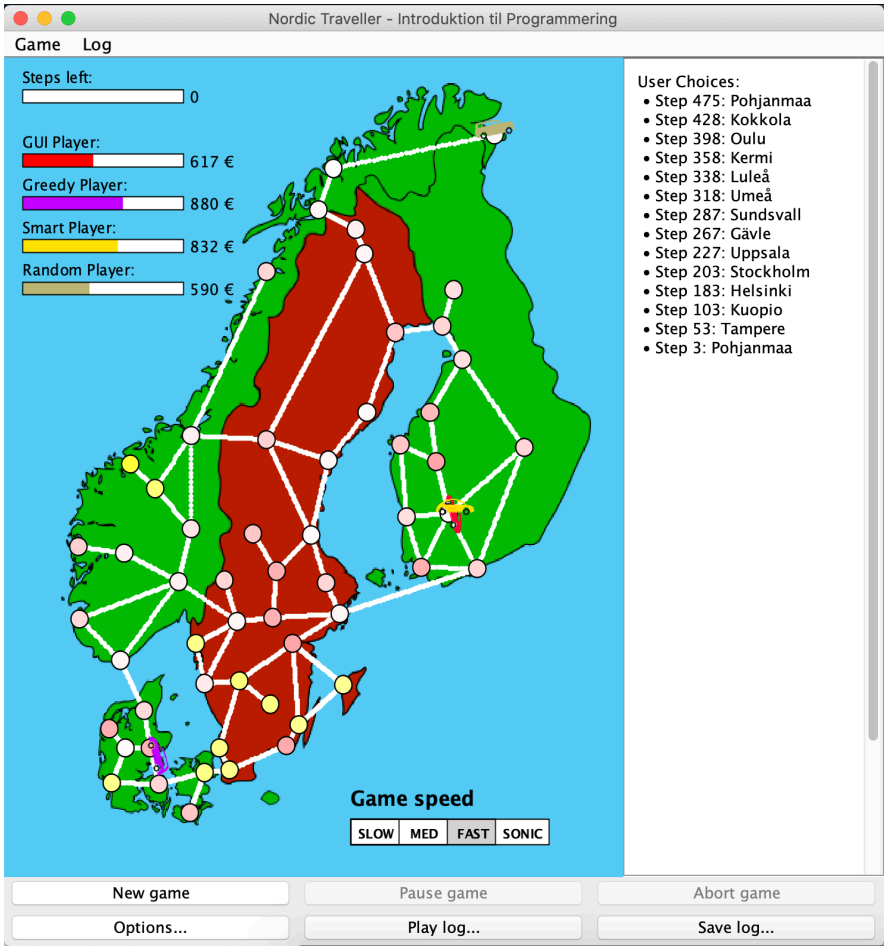
- ***game.playLog(log)***

hvor ***game*** er en eksisterende feltvariabel i ***GUI*** klassen og ***log*** er den log, der skal afspilles.

Opgave 4

Denne opgave er valgfri, forstået på den måde, at afleveringen kan godkendes, uden at I har løst opgave 4, men så kan I højst få 2,5 point.

Indsæt et tekstfelt, hvori I lister de træk, som brugeren laver. Da der kan være mange træk, skal tekstboksen forsynes med scrollbarer, så den får nedenstående udseende.



Nedenfor følger nogle vink til, hvad I skal gøre.

Opret en ny feltvariabel ***choiceText***. I konstruktøren sættes feltvariablen til at pege på et nyt tekstfelt vedhjælp af nedenstående kode:

- ***choiceText = new JTextArea(50,15);***
  choiceText.setText("User Choices:");
  choiceText.setMargin(new Insets(10,10,10,10));

Første linje angiver tekstfeltets størrelse. Anden linje indsætter startlinjen i vores tekst. Tredje linje laver nogle passende marginer. Tilføj en linje, der gør, at tekstfeltet ikke kan editeres.

Dernæst bruges *Decorator* designmønsteret fra kursets sidste forelæsning, og der indsættes nedenstående i konstruktøren:

- ***JScrollPane userChoices = new JScrollPane(choiceText,***
  JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
  JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);

Første parameter angiver, at det er tekstfeltet ***choiceText***, der dekoreres med scrollbarer, mens de to andre parametre specificerer, at vi ønsker en lodret scrollbar, men ikke en vandret.

Find det sted i ***GUI*** klassens konstruktør, hvor der laves en anonym indre klasse, som er en subklasse af ***MouseAdapter*** klassen (beskrevet i forelæsningen om grafiske brugergrænseflader). Efter metodekaldet ***game.clickCity(c)*** insættes nedenstående metodekald, som bevirker at der skrives en linje i vores tekstfelt, hver gang brugeren vælger en by ved hjælp af musen:

- ***choiceText.append("\n ● Step ” + game.getStepsLeft() + ": " + c.getName());***

Find det sted i ***GUI*** klassens konstruktør, hvor der er følgende metodekald ***game.clickCity(best).*** Efter dette metodekald indsættes nedenstående metodekald, som bevirker, at der udskrives en linje i vores tekstfelt, hver gang brugeren vælger en by ved hjælp af piletasterne:

- ***choiceText.append("\n ● Step ” + game.getStepsLeft() + ": " + best.getName());***

Find det sted i ***GUI*** klassens konstruktør, hvor layoutet for ***superpanel*** sættes til ***BoxLayout*** og ret det til ***BorderLayout***. Kortet (som hedder ***panel***) indsættes i ***CENTER***, de seks knapper (som hedder ***buttons***) i ***SOUTH*** og det nye tekstfelt ***userChoices*** i ***EAST***.