Krav- och analysdokument för Den försvunna kossan

Ingrid Stake, Olivia Månström, Tove Nilsson, Elvina Fahlgren

24 oktober 2021
Version 1

1 Introduktion

Den försvunna kossan är ett sällskapsspel i form av en desktopapplikation. Spelet går ut på att hitta en ko som gömmer sig under någon av spelets alla brickor placerade runt om på spelplanen, och sedan ta sig i mål. Spelarna turas om att slå en tärning och förflytta sin spelpjäs så många steg som tärningen visar. När en spelare hamnar på en bricka kan spelaren välja att betala eller slå tärningen för att öppna denna. Under en bricka gömmer sig en kobjällra, komocka, ko, häst, gris eller ingenting, beroende på vilken kommer spelaren att påverkas på olika sätt.

1.1 Definitioner, akronymer, och förkortningar

- JSON - Är textbaserat format som beskriver ett objekt och används för att utbyta data. Det är enkelt för människor att skriva och läsa, så väl som en dator.
- GSON - Är ett Java-bibliotek som bland annat kan användas till att konvertera en JSON sträng till motsvarande Javaobjekt.
- JUnit - Är ett ramverk att använda vid enhetstestning i Java.
- Modell - utgör logiken för den information som applikationen bearbetar.
- Vyn - Exponerar delar av modellen för användaren.
- Kontroller - Tar emot input från användaren för att utlösa förändring i vyn eller modellen.

2 Krav

2.1 Användarberättelser

1. Som användare vill jag att det finns spelare för att kunna spela spelet.

   #### Acceptanskriterier

   - Spelaren ska ha en egen färg.
   - Det ska vara 1-4 spelare.

   #### Uppgifter

   - Skapa en spelare-klass.
   - Skapa en boolean som avgör om spelaren har hittat kossan.
   - Skapa en boolean som avgör om spelaren har hittat ett visum.
   - Skapa en metod som flyttar spelare till vald position.
   - Skapa en int för att hålla koll på spelarens saldo.

2. Som användare vill jag att det finns en spelplan för att få en överblick över spelet.

   #### Acceptanskriterier

   - Spelplanen består av positioner av olika typer.
   - Positionerna ska uppbyggd som en graf.

   #### Uppgifter

   - Skapa en JSON fil att representera spelplanen och alla positioner.
   - Kunna läsa in en JSON fil.
   - Hantering för omvandling av JSON fil till Java objekt.
   - Felhantering för syntaxfel i JSON fil.
   - Lägg till noder och dess grannar i JSON filen.

3. Som användare vill jag att spelarna visuellt skiljer sig åt för att veta vem som är vem.

Acceptanskriterier

- Det ska finnas en spelfigur för vardera spelare.

Uppgifter

- Skapa olika FXML-filer med olika färger för de olika spelarna.
- Skapa en Enum att representera olika färger på spelare.
- Skapa en metod i klassen PlayerColor som beräknar vilken FXML-fil som hör till vilken spelarfärg.

4. Som användare vill jag ha olika positioner på spelplanen utan/med markörer för att kunna ta mig fram i spelet

Acceptanskriterier

- Det ska vara rätt fördelat med markörer på spelplanen.
- Markörerna går att vända upp.
- Samtliga markörer ska ha en egen FXML-fil.

Uppgifter

- Skapa Enum för olika typer av spellogik för positioner.
- Skapa FXML-filer för framsidan av varje markör.
- Skapa en klass TilePosition för positionerna av markörer.
- Skapa en metod som uppdaterar markörens läge i TilePosition-klassen.

5. Som användare vill jag att spelpjäsen förflyttar sig till den position jag valt för att spelet ska kunna fortgå.

Acceptanskriterier

- Spelpjäsen ska förflyttas till den valda positionen.
- Endast den aktiva spelaren ska kunna förflytta sig.

Uppgifter

- Ett tryck på en nod ska starta ett event.
- Kontrollen ska bli medveten om att en användare har tryck på en nod.

- Kontrollen ska anropa rätt metod från modellen.

6. Som användare vill jag kunna slå en tärning för att ta mig fram i spelet.

Acceptanskriterier

- Spelaren ska kunna rulla tärningen.
- Efter tärningen är rullad ska spelaren kunna förflytta sig.
- Spelaren ska kunna rulla tärningen igen om hen har förflyttat sig till en markör, om inte ska spelaren bara kunna rulla tärningen en gång.

Uppgifter

- Skapa en tärningsklass.
- Skapa en metod som slumpar ett tal mellan 1-6 och returnera talet.
- Gör tärningskastet aktivt för att förflytta sig på spelplanen eller för att öppna en markör.
- Tillåt den aktiva spelaren att rulla tärningen.
- Visa resultatet av tärningskastet.

7. Som användare vill jag att pjäsen förflyttas till den position jag väljer så länge det är inom det upplysta området för att spelet ska följa reglerna.

Acceptanskriterier

- Spelpjäsen försvinner från den ursprungliga positionen och dyker istället upp på den valda positionen på spelplanen.
- Spelpjäsens point-property uppdateras.
- Upplysningen försvinner från valbara positioner.

Uppgifter

- Skapa en FXML-fil för att visa att noden är upplyst.
- Skapa en knapp för simulering av tärningskast.
- Byt till den upplysta FXML-filen för de positioner som är giltiga efter ett tärningskast.
- Byt tillbaka till den vanliga FXML-filen när förflyttningen har skett.

8. Som användare vill jag ha en startposition för att inleda och avsluta spelet.

   #### Acceptanskriterier

   - Det ska finnas en startposition på spelplanen.

   #### Uppgifter

   - Skapa logik för för att spelet ska avslutas när en spelare som har antingen kossan eller ett visum tar sig till startpositionen.

9. Som användare vill jag välja antalet spelare innan påbörjat spel för att kunna anpassa spelet.

   #### Acceptanskriterier

   - Applikationen ska kunna läsa input från spelarna.
   - Användarna kan välja bland antalet spelare.
   - Spelet kan inte starta förens antalet spelare är bestämt.
   - Antalet spelare kan inte ändras när spelet har börjat.
   - Spelet anpassas efter antalet valda spelare.

   #### Uppgifter

   - Begränsa antalet spelare till 4.
   - Skapa en *choise box* där användaren kan välja antalet spelare.
   - Koppla antalet spelare som ska spela till den valda siffran.

10. Som användare vill jag kunna starta spelet för att spela spelet

    #### Acceptanskriterier

    - Spelet startar när spelaren har tryckt på startknappen.
    - Spelarna tilldelas rätt mängd pengar.

Uppgifter

- Dela ut startbelopp till varje spelare.
- Placera spelarna på startnoden.
- Skapa en klass som hanterar vybyte.
- Skapa en klass som hanterar vyn för startsidan.
- Skapa en FXML-fil för startvyn.
- Koppla kontroller till knapp på startvyn.

11. Som användare vill jag att markörerna får en slumpad position på spelplanen för att det ska vara spännande att spela.

Acceptanskriterier

- Markörerna ska få en slumpad position på spelplanen.

Uppgifter

- Skapa en metod i spellogiksenumen som returnerar en slumpad lista med rätt fördelning av spellogikselement på spelplanen.

12. Som användare vill jag kunna vända upp en markör genom att slå tärningen för att spelet ska fortgå.

Acceptanskriterier

- En markör kan endast vändas upp vid tärningsslagen 4, 5 och 6.
- Markören ska vändas och det ska vara synligt för spelaren vad som låg under.

Uppgifter

- Skapa en gamelogic klass.
- Om spelar får ett giltigt tärningsslag (4,5,6) vänds markören upp.
- Om ja, kör metoden *executeTileTurn*, om nej händer inget.

13. Som användare vill jag kunna vända upp en markör genom att betala pengar för att spelet ska fortgå.

Acceptanskriterier

- Ska endast vara ett alternativ om spelaren har tillräckligt mycket pengar.
- Spelaren ska kunna välja att betala pengar för att öppna en markör.
- Markören ska visas för spelaren.

Uppgifter

- Kolla om spelaren har tillräckligt med pengar.
- Debitera spelaren.
- Ändra status på *tilePositionen* så att den är vänd.

14. Som användare vill jag kunna välja att betala 3000kr om jag står på en flygplats för att flyga. (Inte implementerad - brist på tid)
15. Som användare vill jag kunna välja att betala 1000kr om jag står på en båtplats för att åka båt. (Inte implementerad - brist på tid)
16. Som användare vill jag ha spelarkort bredvid spelplanen som visar den aktuella spelaren tillsammans med dennes egendomar för att kunna lägga upp en spelstrategi.

Acceptanskriterier

- Rutorna ska synas under hela spelets gång.
- Rutorna ska visa vems tur det är i spelet.

Uppgifter

- Skapa FXML-filer för respektive spelarkort.
- Skapa en motsvarighet till *ViewResource* för en *playerCardView* som innehåller den aktuella spelaren.
- Spara spelkorten i gameboardview.
- Visa kossa/kobjällra om spelaren har hittat dessa.
- Ladda in playerCardViews i gameBoardView.
- Gör så att currentPlayer-kortet skjuter upp och sedan återgår till sin originalposition när turen går vidare.
- Uppdatera fälten i varje kort när något har skett.

17. Som användare vill jag att spelet avslutas då den första spelaren med kossan/visum tar sig in i mål.

    #### Acceptanskriterier

    - Spelet verifierar att den som gått i mål har ett visum/kossa.
    - Spelomgången avslutas så att inga fler drag kan göras.

    #### Uppgifter

    - Undersök om det är en giltig vinst. Alltså att användare är på startpositionen och har antingen visum eller kossa.
    - Avsluta spelomgången när någon vunnit.

18. Som användare vill jag få en bekräftelse i form av en ny vy om någon vinner för att veta om spelet är avslutat

    #### Acceptanskriterier

    - Utlöses när första användaren kommer i mål med ett visum/kossa.
    - Deklarerar för alla spelare vem som vunnit spelomgången.

    #### Uppgifter

    - Visa vem som vunnit, dess spelkort hamnar i mitten av spelplanen.

2.2 Definition av slutfört

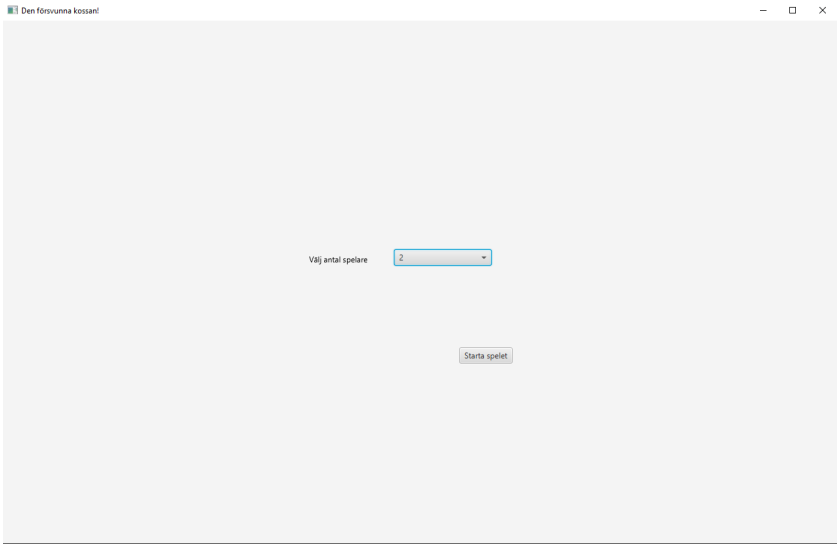
För att en användarberättelse ska anses vara slutförd krävs det att nedan listade acceptanskriterier är uppfyllda.

- Minst två medlemmar i gruppen ska gå igenom koden för att se till att den följer den kodstandard som bestämts.
- Javadoc ska skrivas på kod som ingår publika metoder, samt den kod som kan vecka frågetecken.
- Det ska vara minst 90 % testrapportering av raderna.
- Alla uppgifter för användarberättelsen ska vara gjorda.
- Travis ska godkänna koden.

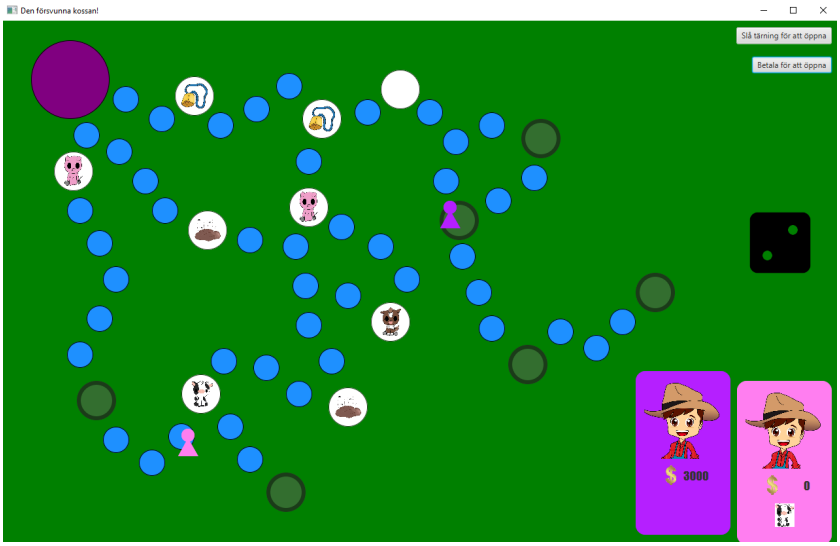
- Användarberättelsen ska finnas representerad i design-modellen, samt beskrivas i SDD och RAD.

2.3 Användargränssnitt

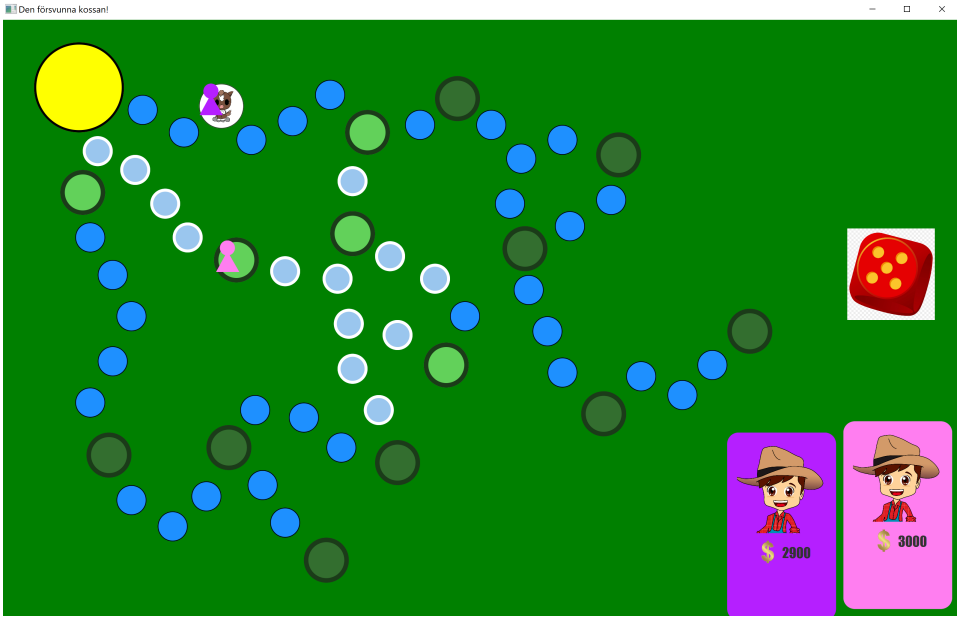
- Startsidan - När användaren startar spelet presenteras en startsida där de har möjlighet att välja antalet spelare.



- Spelplanen - När användaren har valt antalet spelare visas själva spelplanen, i det här fallet är två spelare valda. Här har användaren möjlighet att slå en tärning och ta sig fram på spelplanen. När en spelare står på en markör har den valet att öppna markören genom att betala eller genom att slå tärningen.

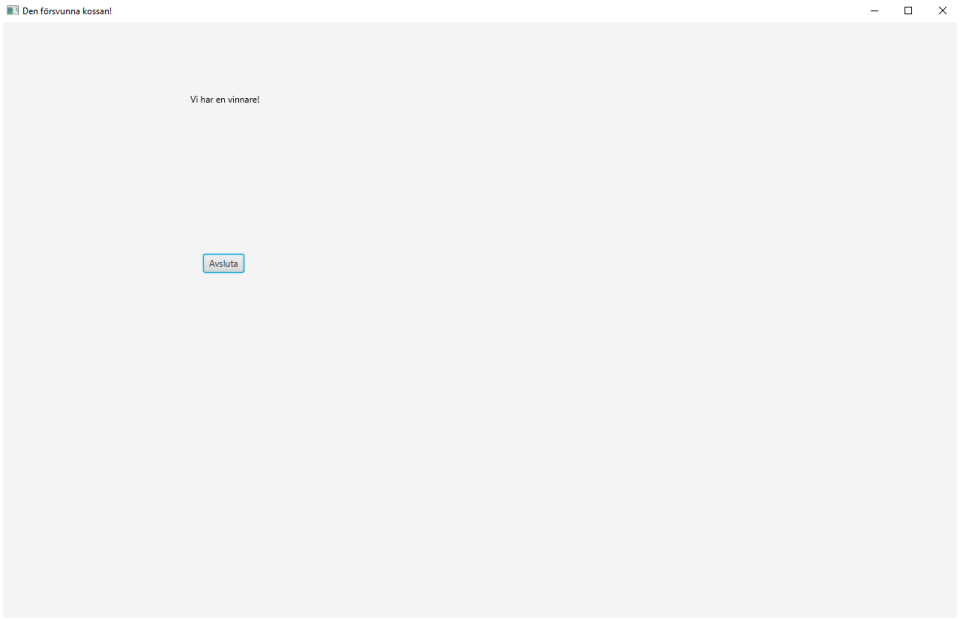


- Upplysta noder - När användaren slår tärningen visas vilka positioner man kan förflytta sig till genom att de lyses upp.

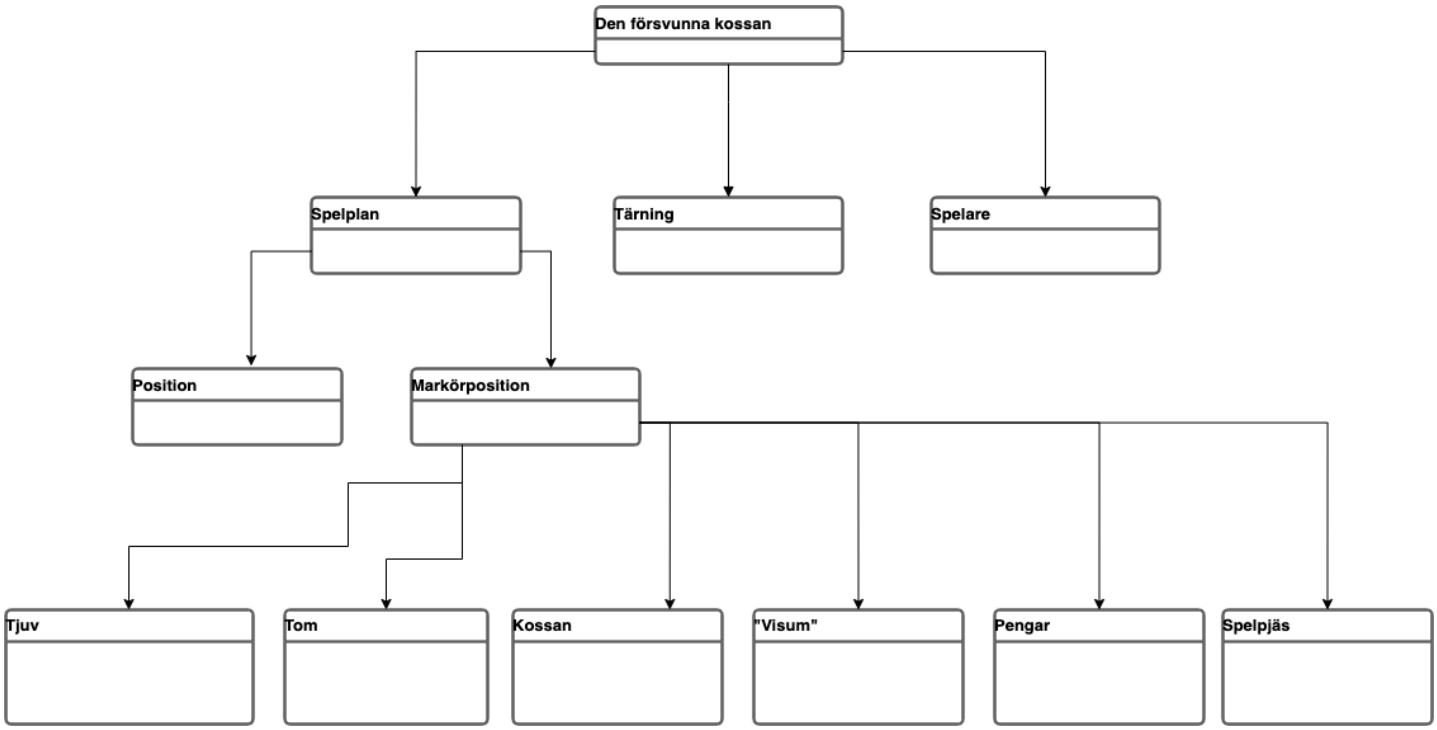


- Spelarkort - På spelplanen visas spelarnas kort där saldo och eventuella egendomar finn synliga för spelarna. Spelkortet för den aktuella spelaren höjs upp för att det enkelt ska synas vems tur det är.

- Slutsidan - När en spelare har tagit sig in i mål visas en sida som talar om att en spelare har vunnit och att spelet har avslutats.



3 Domänmodell



Figur 1: Domänmodell

3.1 Klassansvar

- GameLogic - Ansvarig för logiken under spelets gång.
- Spelplan - Ansvarig för att samordna spelare och positioner.
- Tärning - Ansvarig för att simulera en tärning, det vill säga att slumpa fram värden till resren av programmet.
- Spelare - Den modul med ansvaret att representera en spelare och dess tillstånd.
- Position - Dn modul med ansvaret att representera positioner och dess data och logiktyp.
- MarkörPosition - En modul som representerar markörers positioner och dess data och logiktyp.
- Markörer (Tjuv, Blank, Visum, Pengar, Spelpjäs) - Tillåter spellogiken att bestämma vad som ska ske.

4 Referenser

GSON, 2021. Hämtad från https://github.com/google/gson.

JSON, 2021. Hämtade från https://www.json.org/json-en.html.