DA339A Rapport U4 Grupp 10

Datum för inlämning: 2025-01-14

Gruppmedlemmar:
Namn: Josef Zakaria, Datorid: aq1637
Namn: Mustafa Al-Saffar, Datorid: ap7713

Spelbeskrivning

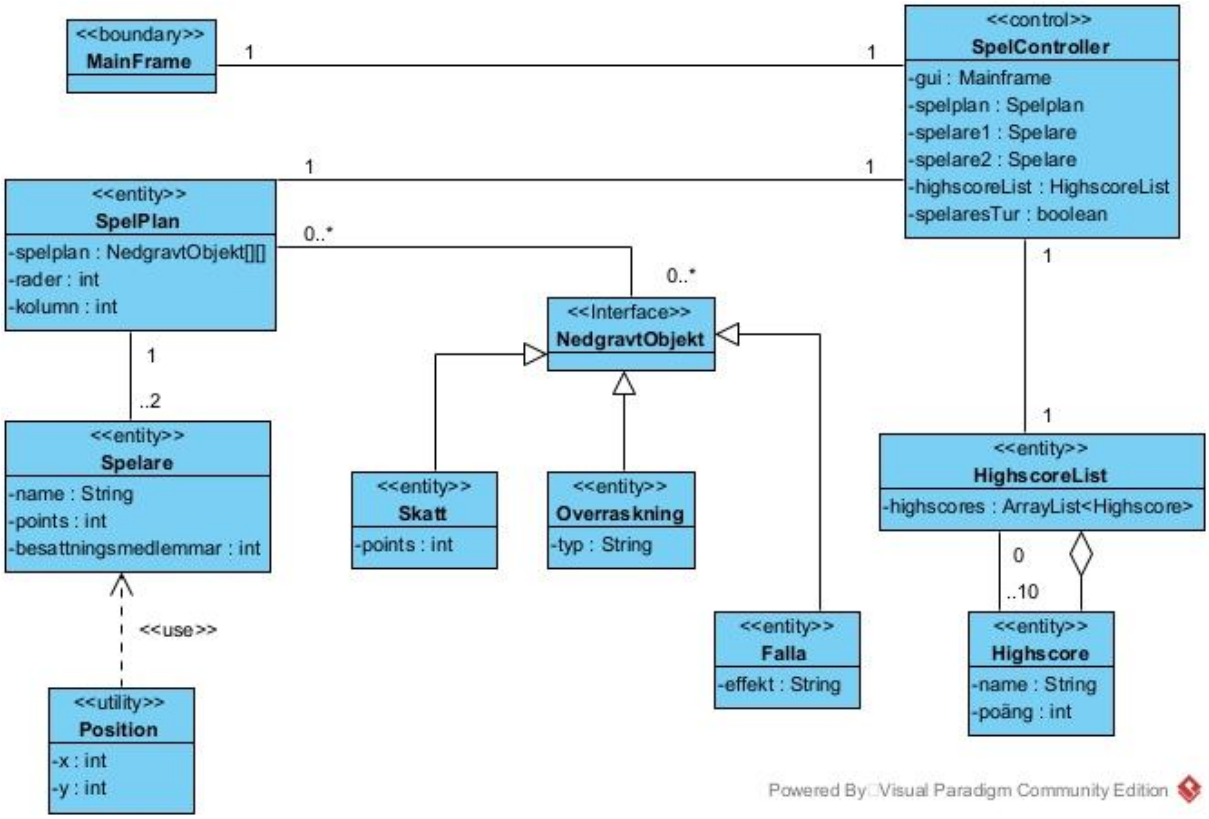
Spelet är baserat på tur där två spelare tävlar om att få ihop så mycket poäng som möjligt, detta gör spelarna genom att ”gräva” på en spelplan som har dolda objekt. Det spelarna gräver upp kan antingen innehålla skatter, fällor eller överraskningar, varje sak som en spelare gräver upp har sina egna unika egenskaper och påverkar spelaren på olika vis.

Skatterna är det som representerar spelarens belöning och kan vara placerade på olika platser och i olika former, som till exempel T-form, en fyrkant, ett kors och L-form. Det finns total 24 positioner på spelplanen som innehåller skatter, dessa positioner är uppdelade i sex förbestämda former med fyra positioner per form. Varje gång en spelare gräver upp en skatt så får den spelaren ett slumpmässigt antal poäng mellan 1 och 50.

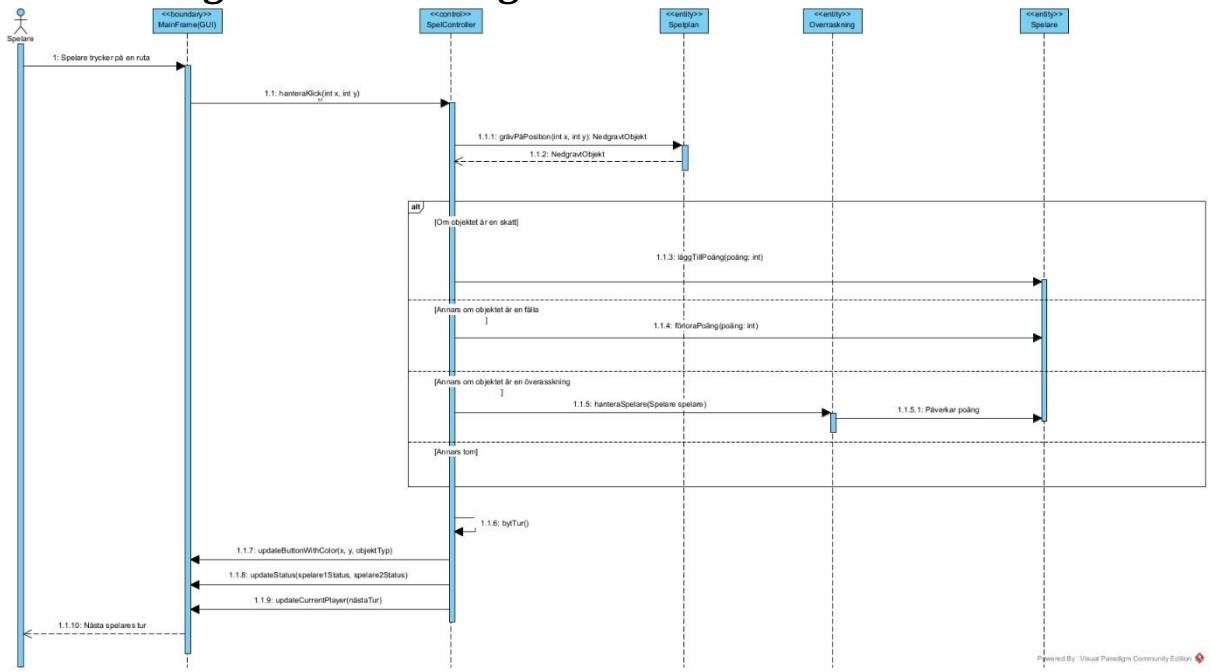
Fällorna är då spelets riskmoment, en spelare som gräver upp en fälla påverkas den spelaren negativt. Fällorna är uppdelade slumpmässigt och placerade på 5 olika positioner. Konsekvenserna till en fälla är förlust av en besättningsmedlem samt fem poäng.

Överraskningarna är det som gör spelet mer spännande då de kan både gynna en spelare eller påverka spelaren negativt. Överraskningarna är fördelade på tre olika positioner på spelplanen, samt att de slumpmässigt antingen är positiva eller negativa. En positiv överraskning belönar spelaren med 10 poäng medan en negativ överraskning tar bort 10 poäng från spelaren.

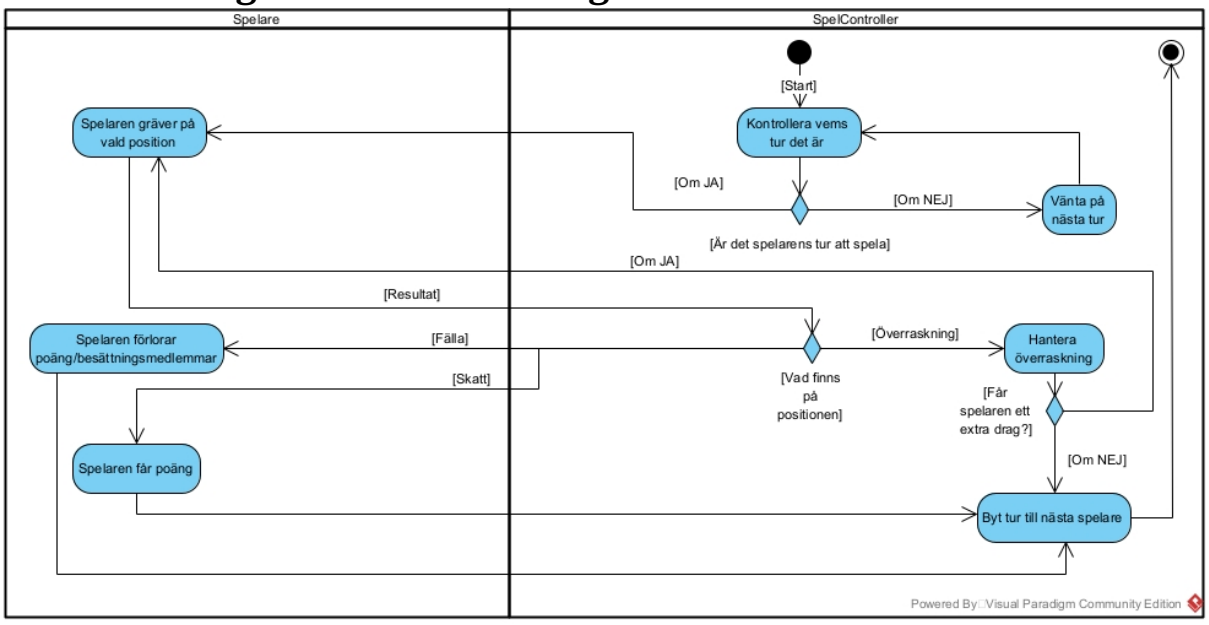
Klassdiagram



Sekvensdiagram för ett drag



Aktivitetsdiagram för turordning



Turordningen i spelet hanteras genom att systemet hela tiden kontrollerar vems tur det är i spelet efter varje drag. Detta sker genom en metod i spelControllern som avgör när en spelare som grävt upp en överraskning ska få ett extra drag eller om nästa spelare ska få turen.

I vanliga fall går turen från en spelare till nästa efter att den nuvarande spelaren har gjort sitt drag. Men för att kunna hantera situationer där en spelare får en extra tur eftre att ha grävt upp en överraskning använder man sig av en kontrollstruktur. När spelaren gräver upp en överraskning sätts en metod igång som då bestämmer ifall spelaren ska få spela en till omgång

eller om nästa spelare får sin tur. Detta kan implementeras med en retur av ett boolean-värde som visar om spelaren får fortsätta.

Om spelaren skulle få spela en omgång till, återgår programmet till att låta samma spelare gräva igen. På så sätt skapas en loop som fortsätter itereras fram tills att spelaren inte har rätt till fler drag, därefter får nästa spelare sin tur. En sådan mekanism gör så att spelet kan hantera överraskningar på ett dynamiskt sätt utan att logiken för övriga turordningar blir påverkade.

Reflektion och motivering till koddesign

Vi valde att använda oss av ett interface för att hantera alla typer av nedgrävda objekt såsom skatter, fällor och överraskningar. Alla klasser som använder sig av dessa objekt implementerar vårt interface, detta skapar en bra struktur för hur vi hanterar våra objekt i spelet. Interfacet gör att spelkontrollern kan interagera med objekten utan att behöva veta deras specifika klass som de tillhör till, detta är på grund av att interfacet innehåller gemensamma metoder.

Vi valde att använda oss av ett interface istället för en abstrakt klass på grund av att de olika typerna av nedgrävda objekt inte delar tillräckligt med gemensam logik som alla typer av objekt kan använda, för att kunna motivera en abstrakt basimplementation. Överraskningar, skatter och fällor har beteenden och unika egenskaper som passar bäst att implementera direkt i respektive klass. Om vi hade använt oss av en abstract klass hade det gjort strukturen onödigt komplicerat.

Ett interface skapade en tydligare och mer flexibel struktur, därför är det ett bättre val. Det räcker med att spelkontrollern hanterar objekten på ett enhetligt sätt, oberoende av deras specifika egenskaper tack vare att de delar en gemensam struktur. Detta gör koden enklare att förstå och underhålla. Att testa systemet blir också enklare eftersom vi enkelt kan skapa testobjekt som följer interfacet.

En nackdel med att använda ett interface är att vi inte kan dela gemensam kod mellan olika typer av objekt. Om flera objekt skulle behöva en metod för att räkna måste varje klass skriva sin egen version. Om vi hade använt en abstract klass så hade vi kunnat lägga den logiken på ett ställe för att slippa upprepa koden på flera platser. Men i vårat projekt har detta inte varit ett stort problem för oss, men om man skulle göra spelet större eller arbetar med ett större spel så är det något som man bör tänka på.

Frågor till andra grupper

Fråga 1

Vi valde att använda ett interface för våra nedgrävda objekt. Hur gjorde ni för att hantera era objekt och varför valde ni den lösningen?

Fråga 2

Vi funderade på hur man skulle implementera extra drag för en spelare vid en överraskning. Hur hanterade ni liknande situationer i ert spel?