Projektbeskrivning

Schack

2024-03-04

Projektmedlemmar:
Markus Svedenheim <marsv260@student.liu.se>

Handledare:
Simon Hansson <simha158@student.liu.se>

Innehåll

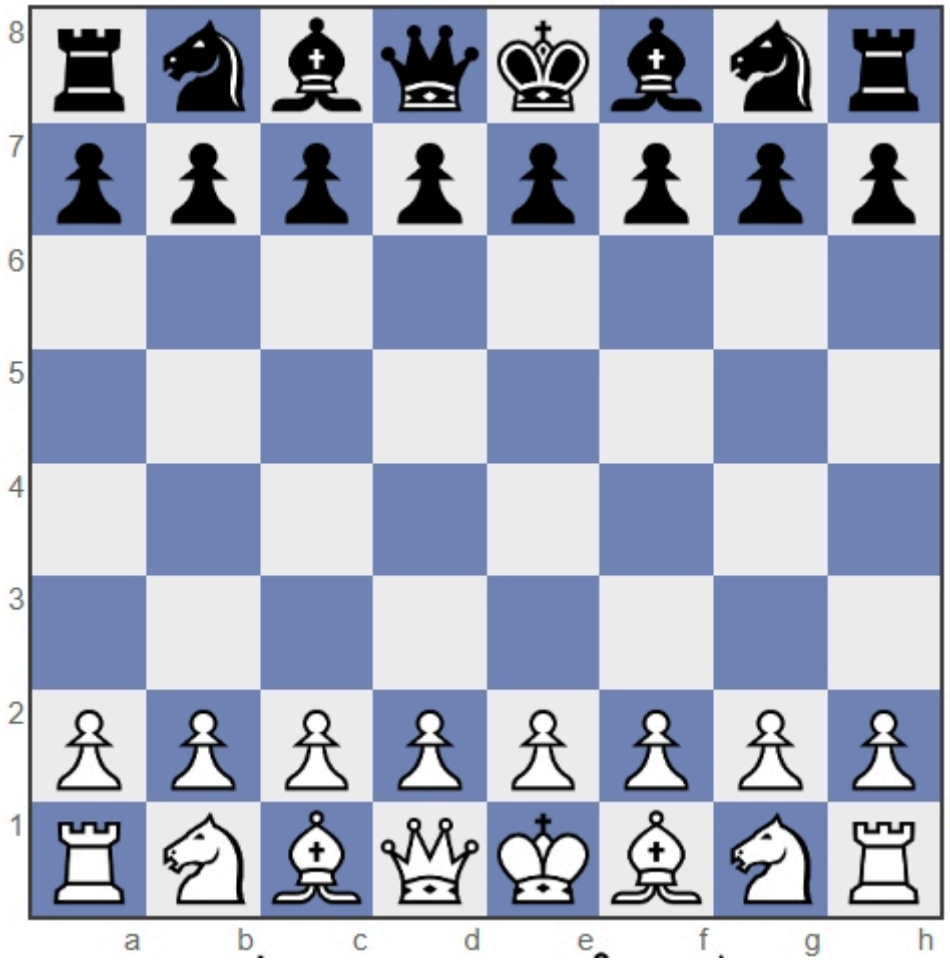
Projektplan

1. Introduktion till projektet

Schack är ett mycket gammalt spel som består av att två olika spelare turas om att flytta en av sina 16 pjäser efter vissa förutsatta regler på ett 8 x 8 stort spelbräde. En spelare vinner först då dess motståndare inte längre kan försvara sin kung och spelet är då över. Det är också möjligt att ett parti schack slutar oavgjort ifall en av spelarna inte har några lagliga drag. Trots vad som kan verka som väldigt enkla regler finns det en enorm komplexitet bakom ett parti schack och det finns en snudd på oändlig mängd olika partier.

Ifall den standardvariant av schack inte längre lockar finns det också flera olika varianter på spelet som var och en bidrar med en unik twist. Exempelvis Fischer Random där startpositionen på alla huvudpjäser slumpas i början av spelet så att all normal teori inte längre är relevant.

Nedan finns en exempelbild på hur ett normalt parti schack ställs upp och kan se ut då det spelas på en dator.



Figur 1. Pjäsernas position i början av ett normalt parti schack. Bilden visar ett exempel på hur ett typiskt parti mot en dator ser ut.

2. Ytterligare bakgrundsinformation

Reglarna i schack kan ofta tyckas vara väldigt svåra, trots det finns det enbart 6 olika pjäser som rör sig på olika sätt. Några ytterligare regler som är viktiga att komma ihåg är att det inte är tillåtet att ta sina egna pjäser eller gå igenom sina egna pjäser. Däremot är det helt okej och oftast smart att ta motståndarens pjäser.

Bonden kan antingen ta en pjäs snett framåt eller gå ett steg framåt. Förutom dess första drag då bonden kan gå två steg framåt. Ifall en bonde når till andra sidan av spelplanen kan den bytas mot en annan pjäs i samma färg, utom kungen.

Tornet kan gå vertikalt eller horisontellt. Dessutom kan tornet och kungen göra en rockad ifall varken av dem har rört sig. Då flyttar sig kungen två steg mot tornet och tornet flyttas till andra sidan av kungen.

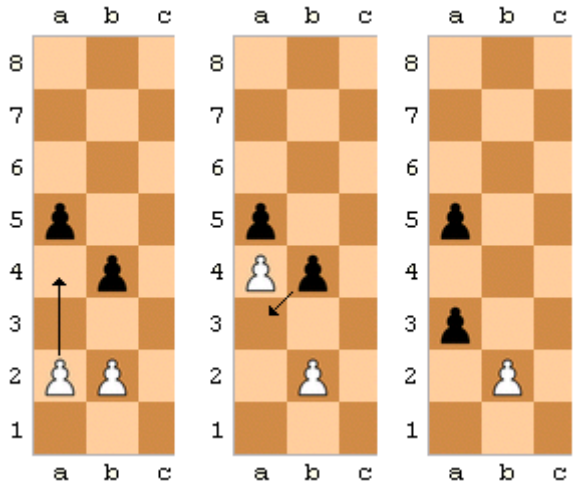
Hästen kan hoppa över pjäser i en L-form. Den hoppar då två steg horisontellt eller vertikalt plus ett till steg åt sidan.

Löparen går på diagonaler, detta innebär att den alltid är fast på samma färg.

Damen är den starkaste pjäsen. Denna kan röra sig både som en löpare eller som ett torn.

Kungen är den viktigaste pjäsen. Denne kan bara röra sig ett steg men ifall fångas av motståndaren är spelet över.

Några ytterligare regler är att ifall kungen är i schack, alltså hotas att tas av en annan pjäs på nästa drag, måste detta åtgärdas omgående. Dessutom får rockad ej genomföras då kungen är i schack. Ifall samma position upprepas tre gånger eller en spelare inte kan göra några drag samtidigt som du inte är i schack, avslutas även spelet i remi (oavgjort). En ytterligare regel som är relativt ny är också en passant: Ifall en bonde går två steg på sitt första drag och därmed går förbi en annan bonde är det möjligt för denna att fångas som om att den endast hade gått ett steg, detta visas nedan.



Figur 2. Denna bild visar hur en passant går till.

Mer information finns att hitta men det garanteras inte att alla regler är implementerade på samma vis. En enkel källa finns här: https://sv.wikipedia.org/wiki/Schack.

3. Milstolpar

#	Beskrivning
1	Ett fönster som går att öppna och stänga som också visar ett schackbräde, det vill säga ett rutnät i skiftande färger.
2	Klass för torn implementeras och går att flytta horisontellt och vertikalt utan att kollidera med kanter eller andra pjäser. Att trycka på pjäser visar också tillåtna drag.
3	Fortsätt med att implementera klass för löpare och häst.
4	Klass för dam som möjligtvis ärver från torn och löpare samt kung.

5	Bönder som kan röra sig ett steg framåt eller ta ett steg diagonalt.
6	Pjäser kan nu tas av andra pjäser och bönder promoveras automatiskt till damer.
7	Implementera rockad, två steg bonde, en passant.
8	Implementera schack och schackmatt.
9	Pjäser ställs nu upp korrekt då spelet startar med svart och vit spelare.
10	Kontrollera att spelet går att köra spelare mot spelare.
11	Meny för att promovera bönder till valfri pjäs.
12	Implementera en startmeny med spela/sluta.
13	Visa en prompt då en spelare hamnar i schackmatt.
14	Visa vilka pjäser som blivit tagna och materiella skillnader.
15	Implementera regler för Fischer Random, undersök speciella regler för rockad. Lägg till detta som alternativ i menyn.
16	Lägg till en informations prompt för olika öppningar. https://github.com/lichess-org/lila-openingexplorer?tab=readme-ov-file#http-api

4. Övriga implementationsförberedelser

Lista av olika grundtankar i projektstruktur.

- Brädet och varje pjäs är en egen klass.
- Brädet kan representeras av en enum/class per ruta i en 2d 8x8 array.
- Varje pjäs håller koll på vilka drag den kan göra och dessa blir tillgängliga då den väljs.
- Möjliga drag kontrolleras genom att stegvis gå i pjäsens riktning till kollision.
- Dam, torn, löpare beter sig lika i att de går rakt till krock.
- Liknande med BoardViewer som i Tetris.

Projektrapport

6. Implementationsbeskrivning

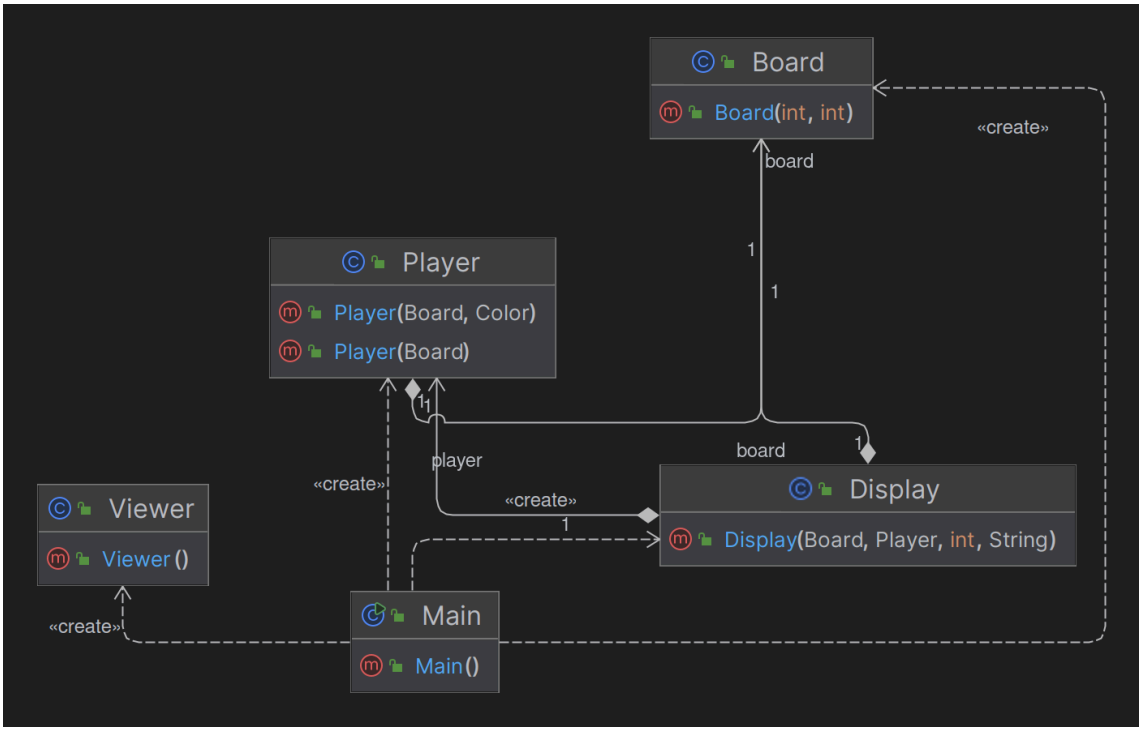
Implementationen följer majoriteten av de givna milstolparna och följer en relativt enkel programstruktur. På grund av att programmet enbart består av självaste schack-partiet är det enkelt att använda och kräver ingen invecklad förklaring.

6.1. Milstolpar

Alla milstolpar från nummer 1 till nummer 10 är avklarade till fullo. Några delar har ändrats däremot, istället för att exempelvis damen ses som en kombination av torn och löpare är nu varje pjäs mer av en egen klass där de ges en riktning de rör sig i och antal steg pjäsen kan ta i den riktningen. Andra milstolpar därefter har ännu inte implementerats med risk för att de ignoreras till mån av tid. Däremot är milstolpe nummer 13 avklarad eftersom att denna bidrar mycket till spelupplevelsen i förhållande till tiden det skulle ta att implementera den.

6.2. Dokumentation för programstruktur

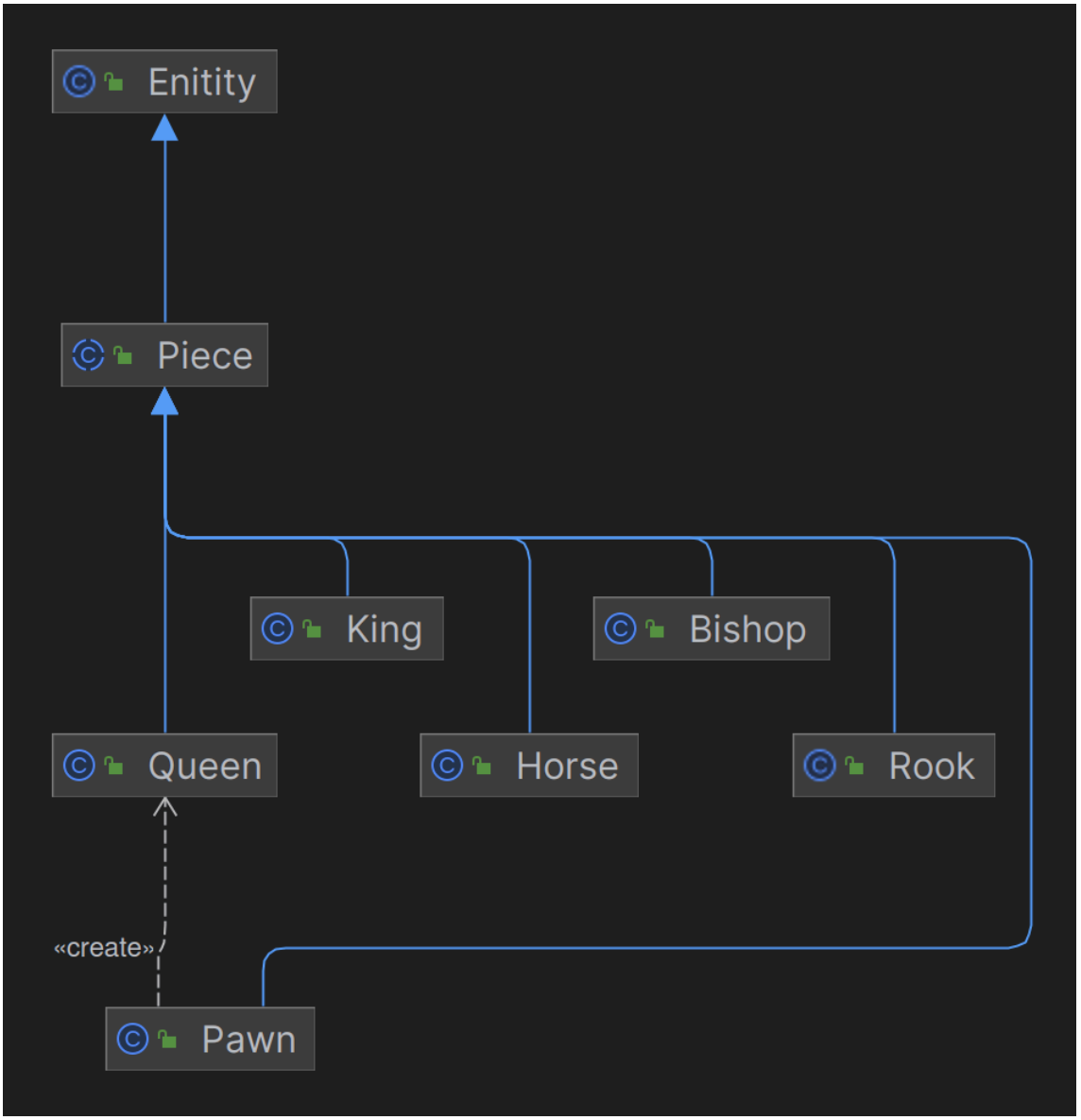
Schackspelet börjar i klassen Main, specifikt dess huvudmetod. Denna metod skapar en instans av Board, en av Player som använder Board, och en instans av Display som använder båda dessa. Board populeras även med flera olika instanser av Piece efter hur ett standard-schackbräde ser ut. Den skapade instansen av Display skickas också till Viewer som skapar ett fönster som användaren kan agera med. Figuren nedan visar hur denna process ser ut.



Figur 3. Visar hur Main klassen skapar alla de klasser som bygger upp schackspelet.

6.2.1 Spelpjäser

Alla spelpjäser ärver från klassen Piece som i sin tur ärver från Entity. Entity-klassen representerar enbart en pjäs med dess färg och typ, till exempel en vit löpare. Piece-klassen ärver då ett "utseende" från Entity men gör det också möjligt för pjäsen att röra sig enligt stepBy-metoden. Exempelvis kan damen röra sig rakt och diagonalt och oändligt långt, medan tornet bara kan röra sig rakt. Hästen är lite av ett undantag men genom att ge åtta riktningar som den kan röra sig i, upp två och höger eller vänster och så vidare kan även denna utnyttja samma algoritm genom att den rör sig med max ett steg. Bonden kan enbart röra sig två steg på sitt första drag. StepBy-metoden fungerar genom att den utgår från pjäsens startposition och inkrementellt stegar i de givna riktningarna. Ifall den stöter in i en annan pjäs eller kanten avslutas stegningen för den riktningen och ifall ett drag gör så att spelaren hamnar i *schack* hoppas det över. Med hjälp av denna polymorfiska implementering blir varje spelpjäs en typ av Piece som också har dess metoder som getMoves vilket gör det lätt att hantera alla pjäser på brädet. En bonde kan också bli till en dam genom att den tillkallar damens konstruktor med sin egen position och färg och ersätter sig själv med denna. Detta händer endast om den gått över hela brädet. UML-diagrammet för implementationen av pjäserna syns i Figur 4.



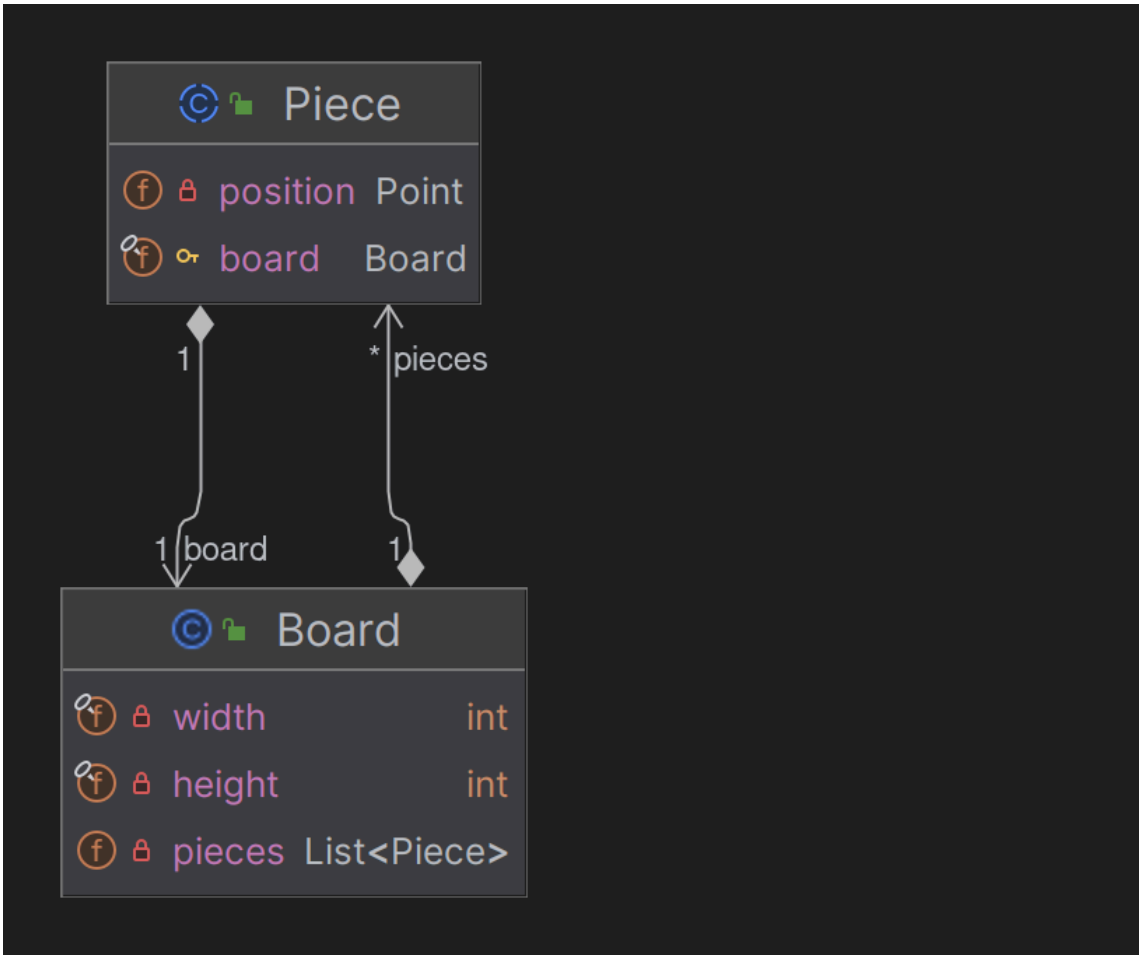
Figur 4. Ett UML-diagram över pjäsernas implementation. Detta visar hur alla pjäser bygger på den abstrakta klassen Piece som i sin tur bygger på den abstrakta klassen Entity. Det syns även hur Pawn skapar en ny Queen.

6.2.2 Spelbräde

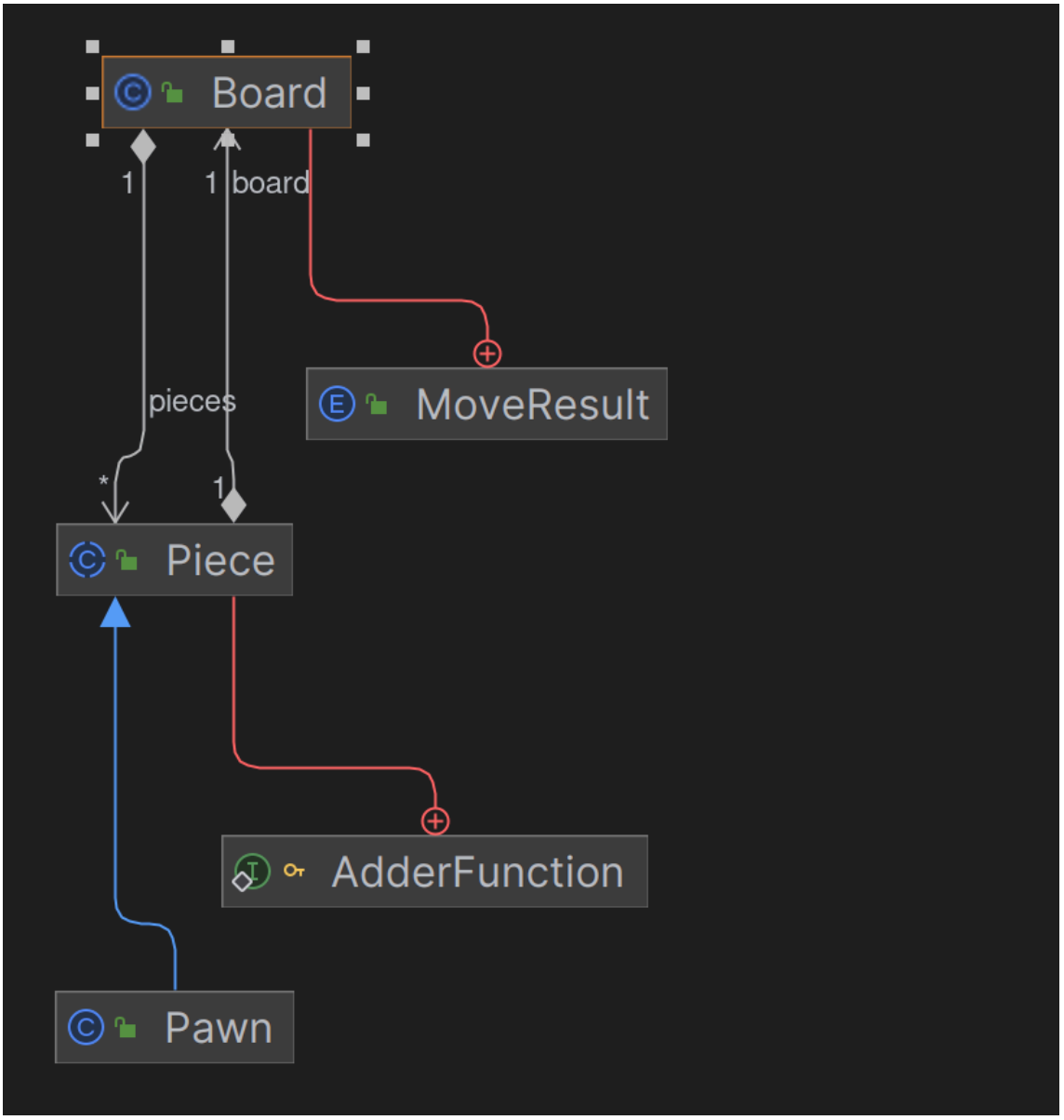
Klassen Board representerar spelbrädet som pjäserna använder. Denna innehåller enbart en bredd och höjd som båda tills vidare är låsta till 8, tillsammans med en lista av pjäser. Detta är istället för att representera brädet som en lista av listor, i detta fall en lista av åtta listor av bredd åtta, där varje punkt innehåller en pjäs. Denna alternativa metod som används istället gör så att varje pjäs måste ha en associerad koordinat på brädet, men då detta ändå skulle behövas för pjäsernas stepBy-metod på något sätt, finns ingen tydlig nackdel men denna implementation. Se Figur 5 för hur Piece och Board relaterar till varandra.

Board innehåller också metoder för att verifiera drag i relation till resterande delar av brädet, exempelvis verifyMove som kontrollerar vad resultatet av ett drag blir. Det finns fyra olika resultat: blockerad av exempelvis en egen pjäs eller vägg; fångar en motståndarens pjäs; den egna kungen hamnar i schack;

och neutralt drag utan märkvärdig effekt. Beroende på dragets resultat kan varje pjäs hantera om det ska läggas till i möjliga drag eller inte med hjälp av den överladdade addToAndStop-metoden. AddToAndStop tar sedan detta MoveResultat och avgör hur stepBy-metoden ska hantera det givna draget. Figur 6 visar en mall för hur detta förhållande kan se ut.



Figur 5. Ett UML-diagram över hur de olika fälten i Piece och Board relaterar till varandra.

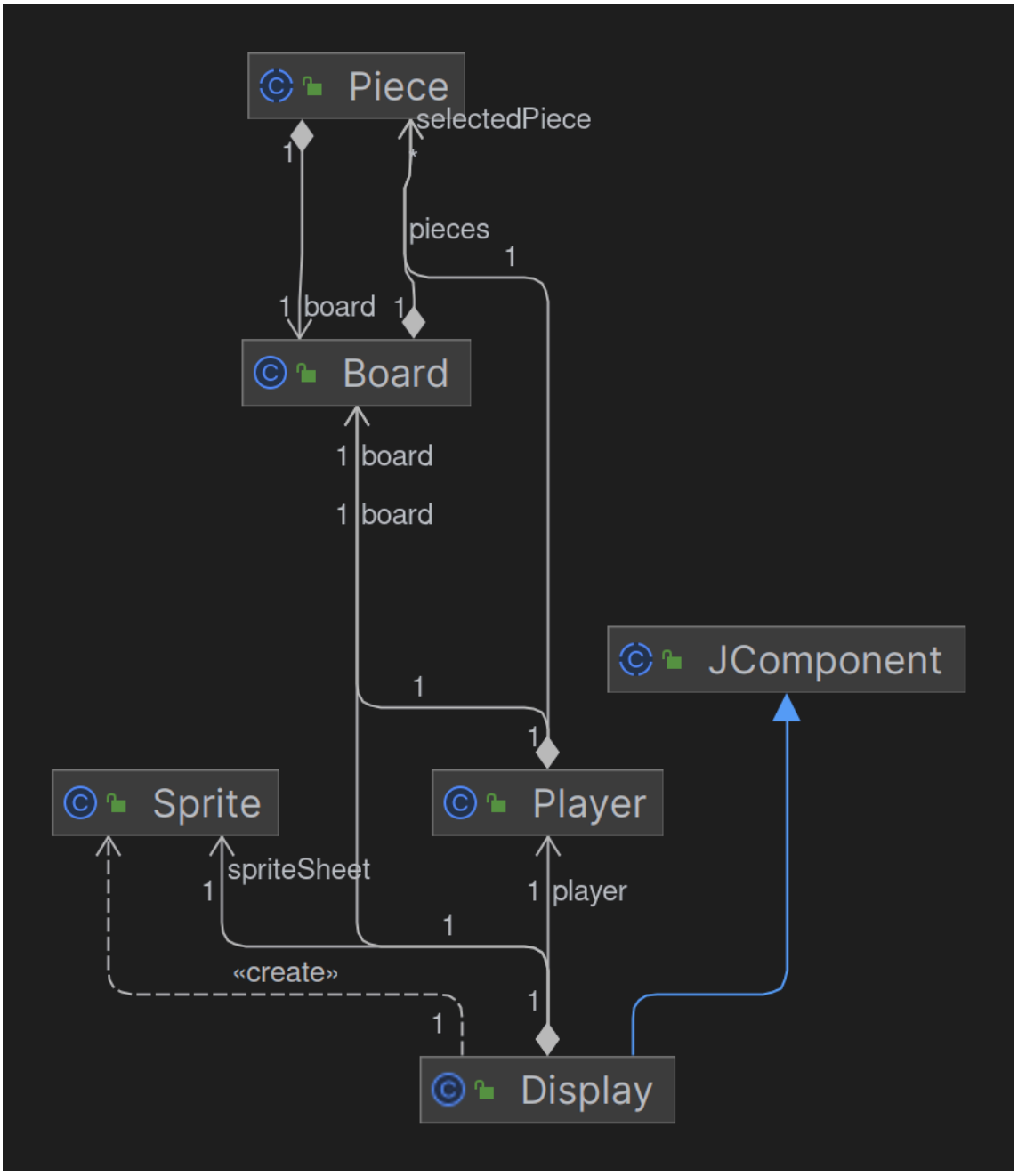


Figur 6. Visar en mall över relationen för hur addToAndStop fungerar. Piece innehåller metoden addToAndStop som agerar beroende på MoveResult från verifyMove. Pawn överladdar denna metod med sin egen eftersom den blockeras framåt av motståndarens pjäser likväl som av sina egna.

6.2.3 Användargränssnitt

Klassen Display styr över det mesta som har att göra med hur användaren använder programmet. Display ärver från JComponent och överladdar paintComponent till att rita ett rutnät färgat som ett schackbräde. Därefter går även Display genom Board för att rita ut alla pjäser på sin korrekta position med sin korrekta sprite som fås av klassen Sprite. Display lyssnar även på musklick för att skicka dessa till Player-klassen som då flyttar pjäserna beroende på hur användaren styr dessa.

Player hanterar självaste spelet och styr över vilka drag som går att göra, vilken spelares tur det är och vilken pjäs som är vald. Det är också Player som kontrollerar ifall det är schackmatt efter att någon spelare har rört en pjäs och styr över när spelet ska sluta. Figur 7 visar hur alla dessa klasser förhåller sig till varandra.



Figur 7. Visar förhållandet från Display-klassen till alla andra klasser som styr över hur spelet går till. Display skapar en egen instans av Sprite men tar Player och Board som argument till dess konstruktor. Board ärver även från JComponent vilket gör att den går att rita.

6.3. Loggning

Under spelets gång kommer också information loggas med hjälp av Javas loggningssystem till en fil vid namnet chess.log med hjälp av FileHandler. Detta inkluderar information som felmeddelanden eller mindre viktig information om hur partiet och pjäserna rör sig.

7. Användarmanual

Programmet är enkelt i sin struktur. Nästan alla regler för schack är implementerade enligt hur de beskrivs i sektion 2 och 3, förutom remi efter 3 drags repetition som ännu

inte finns. Längst ner i vänstra hörnet finns information om spelet, det vill säga vems drag det är eller resultatet från partiet. Det är vit som får det första draget och då det inte finns någon meny är det bara att börja spela. Genom att välja en pjäs visas de olika tillgängliga dragen för den pjäsen och genom att trycka på en av dessa flyttas pjäsen. Ifall man vill flytta en annan pjäs kan man trycka på den pjäsen istället eller på en tom ruta för att inte ha någon pjäs vald. Efter varje drag kontrollerar programmet om det är schackmatt eller patt och i det fallet visas en dialog med resultatet från partiet.