Deltagare Philip R, Zia, Mohammad

Uppgift:

Skapa ett Black Jack-spel i VS och testa enligt TDD.

Godkänt:
• Att spelet kan blanda korten **OK**
• Att spelet kan räkna kortens värde **OK**
• Att spelet kan hålla koll på upp till 5 spelare i en runda **OK**
• Att spelet reagerar om en spelare vinner **OK**
• Att spelet reagerar om huset vinner **OK**
• Att alla publika metoder testas **OK**
• Att alla tester dokumenteras i ett excelblad **OK**
• Att metoderna dokumenteras med vem som skrivit dem **OK**
• Klasser och metoder ska vara delade enligt funktion **OK**
• Clean Code **OK**
• Välkommenterade metoder (XML kommentarer) **OK**

Välgodkänt:
• Att alla godkända delar ska vara uppfyllda **OK**
• Att det ska gå att spela spelet **OK**
• Att man ska kunna satsa fiktiva pengar **OK**
• Hålla koll på spelarnas kassa **OK**
• Datorbaserade spelare eller hjälpare som kommer med förslag **OK** (ProTip hjälper spelaren)
    o Hjälparen räknar inte kort
        ▪ ”Du borde nog stanna”
        ▪ ”Jag skulle nog chansa på ett kort till”
        ▪ osv
• Även datorspelarna ska testas med TDD och dokumenteras **OK**

Reflektion:
GetCard();
När vi först testade metoden som ska hämta det sista kortet i kortleken och radera det ur listan så märkte vi att det inte funkade som det var tänkt. Vi såg till att kalla på metoden och förväntade oss att få returnerat det sista kortet i leken, men testet gick inte genom. Vid närmare koll på metoden såg vi att vi hade skrivit fel index att returnera, och tack vare Testmetoden var det lätt att hitta felet och se till att det blev rätt.

SetProTip();
För att få VG stod det även att man ska testa metoden för datorspelare eller hjälpare, men så som vår metod vår utformad så sätter den ett nytt string-värde till propertyn ProTip. Vi ändrade så att den istället returnerar en string som sparas i en variabel, och då är det enkelt att testa om den väntade strängen är samma som den faktiska.

De flesta publika metoder som vi ha använder sig av ReadLine() och går därför inte att testa. Dock så har vi testat dessa genom vanlig debugging och sett till så att de fungerar som det

är tänkt. Exempelvis InvalidInputCheck() och CheckMinMaxInput(), samt alla metoder som är void och bara skriver ut text i konsolen.

Arbetsfördelning:

När det gäller programmering så har Philip haft fokus på spellogiken och klassen Game, Mohammad har jobbat med Deck-klassen och skapat kort och kortlekar, och Zia har haft ansvar för Output-klassen och allt som skrivs ut i konsollen. Planeringen av själva programmet har gjorts av Philip och Mohammad, medans Zia har förberett dokument för testmetoder och -scenario.

Alla tester och diskussioner kring dessa har gjort gemensamt via skärmdelning i Google Meets.

Reflektion:

Black Jack är ett ganska avancerat spel att göra på bara två veckor, speciellt med tanke på att vi ska fokusera på TDD och inte programmering i första hand, men med tanke på dessa förutsättningar tycker vi ändå att det har gått bra. Vi har kunnat testa de metoder som är möjliga att testa och ändå gjort ett fullt fungerande spel både spellogik och GUI. Den enda som vi saknar och gärna skulle lägga till om vi haft mer tid är datorstyrda spelare med någon form av AI, men det är något man kan bygga vidare på i framtiden.

Men fördelarna med att tillämpa TDD i framtida projekt känns ändå ganska tydliga. Om man jobbar i ett agilt grupparbete så kan man exempelvis komma överens om att alla utvecklare MÅSTE köra alla testmetoder och få godkänt innan man pushar upp sina ändringar. På så sätt minskar man risken att någon ändrar i metoder som redan är testade och funkar som det ska.