Uppdragsbeskrivning

Uppgiften går ut på att i en projektgrupp med 2-3 projektledare (från IPK16-klassen) och 4-5 utvecklare från er klass skapa spelet Yatzy med grafik (Swing) i Java.

Ni väljer själva hur ert spel ska se ut och vilka komponenter ni vill använda er av förutsatt att kraven är uppfyllda.

Ni har cirka fyra veckor på er så planera noga och använd tiden väl. Ni behöver avsätta tid till:

- Analys
    - Fundera kring vilka Epics(Grafik, programlogik etc) och Stories (klasser och metoder etc) som behövs samt ta fram UML-diagram
- Design
    - Kod (namn på variabler och metoder samt struktur på kod, se till att alla i gruppen följer samma stil)
    - Gränssnittet (layout, komponenter och lyssnare)
- Implementation
    - Ta fram fungerande kod för spelet

Krav

För både **godkänt** och **väl godkänt** ska spelet minst uppfylla följande krav:

- Ett enklare UML-diagram över klasserna i spelet
- Spelet visar fem tärningar och ett protokoll där poäng för varje spelare och varje tärningskombination visas
- Användaren ska ha möjlighet att välja vilka tärningar som ska slås om
- Poängen för respektive tärningskombination ska automatiskt räknas ut
- Efter alla tärningskombinationer är ifyllda så ska spelet räkna ut eventuell bonuspoäng och meddela den totala poängen

Dessutom ska koden vara strukturerad, dokumenterad och ni ska visa på att ni förstår och kan använda er av objektorienterad programmering samt grafik med Swing i Java. Onödig kodupprepning ska i största mån undvikas.

För betyget **godkänt** ska spelet minst uppfylla följande krav:

- Tärningarnas värden visas med text
- Spelet kan spelas i ordning uppifrån och ner
- Spelet spelas med en mänsklig spelare

För betyget **väl godkänt** ska spelet uppfylla kraven för godkänt samt minst uppfylla följande krav:

- Tärningarnas värden visas med bilder
- Spelet ska fråga om namn på spelare och visa vems tur det är att slå tärningarna
- Spelet har stöd för godtyckligt antal spelare

samt minst ett av följande krav

- En spelare kan vara mänsklig eller dator
- Spelet kan spelas först övre delen i valfri ordning, sedan nedre delen i valfri ordning
- Spelet kan spelas i valfri ordning

Dessutom ska koden tydligt följa en etablerad stilguide. Ni kan redogöra och argumentera för att er kod är av hög kvalitet.

Metod

Ni kommer jobba med Scrum med Trello som huvudverktyg för att hålla koll på Epics och Stories . GitHub används för att distribuera kod mellan projektmedlemmarna. Gemensamma Slack-kanaler upprättas i syfte att integrera trello och GitHub för PO att följa processen, samt underlätta för kommunikation mellan gruppmedlemmarna.

Möten

Mötestyper

- Status - Hur ligger vi till och är det något som hindrar oss från att komma vidare ?
- Grooming - Är varje Story kortare än fem dagar och tillräckligt specifik?
- Demos - Utvecklarna visar upp vad de gjort för projektledaren
- Kaizen - Vad kan förbättras?
- Dagliga möten Daily StandUp (hålls av utvecklarna själva) - För varje story- Vad behövs göras för att vi ska kunna komma framåt?

Veckomöte 1 - Fokus på nuvarande arbete resultat hittills (Sprint Review/Sprint Retrospective)

- Status
- Frågor och diskussion kring de stories som ligger på Trello-boarden

Veckomöte 2 - Fokus på kommande arbete (Sprint Planning)

- Grooming
- Kaizen

Arbetsflöde

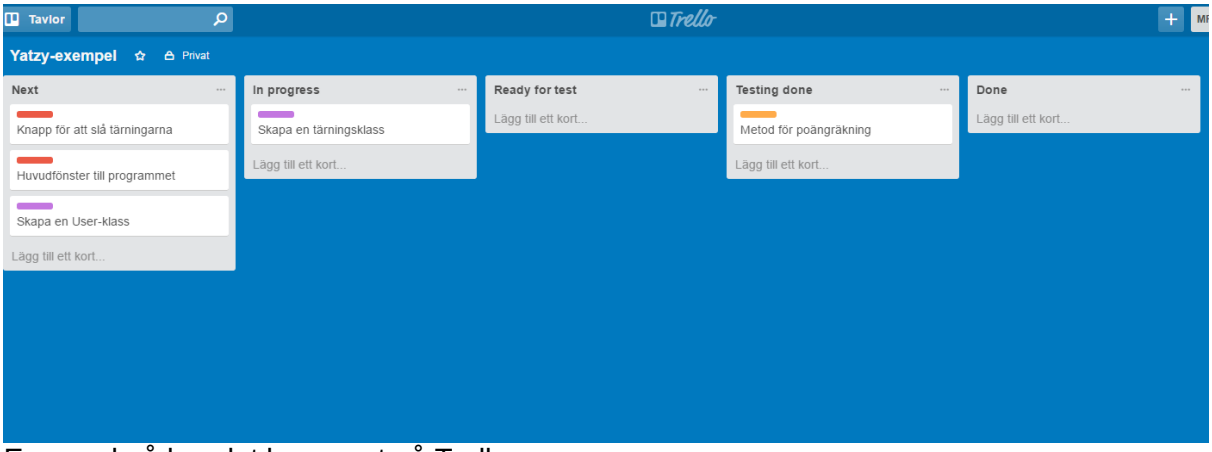
Trello, Epics och Stories

Epics kommer användas för större arbetsdelar (t.ex “grafiskt användargränssnitt”). Varje Epic kommer innehålla Stories där en Story är en mindre del av projektet som kan arbetas fram (t.ex. “En klass för att representera en tärning”). På Trello administreras Epics med etiketter. Exempelvis kan ni välja att allt som har med grafik att göra får en grön etikett.

Stories kommer att läggas i Trello-backlogen och gås igenom på grooming-möten där projektgruppen ser till att ingen Story är längre än fem arbetsdagar (om en story är längre så bör den delas upp i mindre delar)

The Development Board bör bestå av följande kolumner på Trello:
Nästa | Pågående | Redo för test | Test klara | Färdig

1. Epics och Stories bör arbetas fram av gruppen första veckan och läggas in i nästa-kolumnen på Trello.
2. Utvecklarna tar en story från “Nästa” och flyttar till “Pågående” när de börjar jobba med en ny story. En ny branch i Git bör skapas för den “featuren” så ni inte jobbar i master-branchen.
3. När utvecklaren/na känner sig klara med en story bör någon annan i gruppen som inte jobbat med den delen testa och försöka komma på elaka sätt som gör att programmet kan krascha.
4. När utvecklarna anser sig att en story klarar av testerna flyttas den till “Test klara”
5. Nu kan den flyttas till “Färdig” och mergas med Master-branchen på Git.



Exempel på hur det kan se ut på Trello.

Git

Git bör användas som versionshanteringssystem och som huvudsakligt verktyg för att dela kod mellan gruppmedlemmarna.

Varje ny “feature” som skapas bör ha ett en egen branch som namngivs efter ett givet mönster. Exempelvis: feature/<trello-ID>-<namn>