Student: Tatjana Mostachjova

23/12/2022

Redovisning av grupparbete.

Deadline för inlämning av redovisning: 23 december 2022 kl. 17:00

1. Inledning

Kunden Christopher´s vision att en grupp programmerare skapar det berömda gammal klassisk brädspelet ”Hängman” 2.0. Han har sina egna speciella önskemål och sin egen vision om konceptet för detta spel. Istället för att vänta på sin tur, varje spelare har en av motståndarens ord för att gissa på och hänga egen ”Hängman”.

Hans förväntningar är :

- Spelet är förväntas som ett familjespel och lämpligt för barn och ”Hangman” bytes mot en ”Tårta” där tåtrbitar försvinner vid misstag.
- Det är ett lokalt spel.
- Minst 2 spelare som spelar samtidigt, gärna mer i framtiden (upp till 4-8).
- Varje spelare får poäng när de gissat rätt på motståndarens ord.
- När spelare får ett antal poäng, då avslutas spel.
- Vinner den spelare som har mer poäng.
- Spelare kan gissa på hela ord eller på bokstäver.
- Programmet måste ha en rättstavningsfunktion
- Spelare skriver själva dem ord som ska gissas av varandra.
- Spelare har möjlighet att avsluta spelet.

Länk till gruppens Github repo: https://github.com/rodercode/HangmanGruppFive

Länk till gruppens Trello: https: //trello.com/b/C95KjqR7/hangman

2. Reflektion

User story 1 formulering:

”As a player I want to be able to see visually how many letters in the word I will guess, and see those that I have already guessed correctly.”

Tatjana Mostacjova arbetade med den här delen av uppdraget. Min uppgift var att göra det så att spelarna visuellt kunde se hur många bokstäver det finns i ordet de gissar på. För att lösa detta problem använde jag ett speciellt SceneBuilder program där det finns en så kallad visuell editor. Med hjälp av detta program skapade jag speciella fält där placerades sådan " _ " tecken som kommer att visas för användare, vilken indikerar de dolda bokstäverna i det dolda orden och placerade dessa fält på rätt plats. Efter det gjorde jag ändringar i koden så att när spelarna tittar på skärmen kommer de att se hur många bokstäver orden de kommer att gissa på innehåller. Senare, under utvecklingen och kompliceringen av programmet, ändrades och förbättrades koden många gånger, men ursprunglig kod var sådan.

```
16 +
17 +
18 +     public void button() {
19 +         String word = userInput.getText();
20 +         String hiddenWord = " ";
21 +         for (int i = 0; i <word.length() ; i++) {
22 +             hiddenWord = hiddenWord + " _ ";
23 +         }
24 +         wordGuess.setText(hiddenWord);
25 +     }
26 + }
```

Fxml koden ser så ut:

```
</Label>
<Label fx:id="wordGuess1" alignment="BASELINE_CENTER" contentDisplay="RIGHT" layoutX="17.0" layoutY="430.0" prefHeight="67.0" prefWidth="419.0" text="_ _ _ _ _ _ " textFill="#e41397">
   <font>
      <Font name="Comic Sans MS" size="48.0" />
   </font>
</Label>
```

User story 2 formulering:

”As a player I want to be able to change scene so I can go to a different scene in the program.”

Spelare kan se visuellt hur ”tårta” försvinner hos motståndare i ett nytt fönster, då behövs det att implementera en metod som byter scene.

Nästa uppgift jag arbetade med var att spelaren efter att han skrivit in sitt ord kunde byta scen dvs hoppa över till nytt fönster. I början av spelet öppnas den första tangentbordsscenen, där spelaren kommer på och skriver in ett ord, som hans motståndare, sedan, ska försöka gissa. Efter att spelare har skrivit in ordet var det nödvändigt att få programmet att automatiskt byta till en annan scen och ett nytt fönster öppnas för spelaren, där dolda ord visas med hjälp av specialtecken och spelet fortsätter i en ny scen.

Min uppgift var att skriva en metod (changeScene) så att programmet byter scen. Och den här scenbytesmetoden anropas sedan i den andra metoden (changePlayer) och börjar fungera först efter att särskild villkoret är uppfyllt, under vilket alla deltagare i spelet har skrivit klart sina ord.

```
44
        1 usage   tanjamost +1
45      @FXML
46  @   public void changeScene(Stage stage) throws IOException {
47          FXMLLoader fxmlLoader = new FXMLLoader(StartMenu.class.getResource( name: "GameView.fxml"));
48          stage.setTitle("Hangman");
49          stage.setScene(new Scene(fxmlLoader.load()));
50          stage.show();
51      }
52
        1 usage   RoderCode +2
53      public void changePlayer() throws IOException {
54          if (currentPlayer < 4) {
55              currentPlayer++;
56              playerPlate.setText("Player " + currentPlayer + ": Enter A Word");
57          } else {
58      //          switchScene
59              changeScene(new Stage());
60          }
61      }
62
```

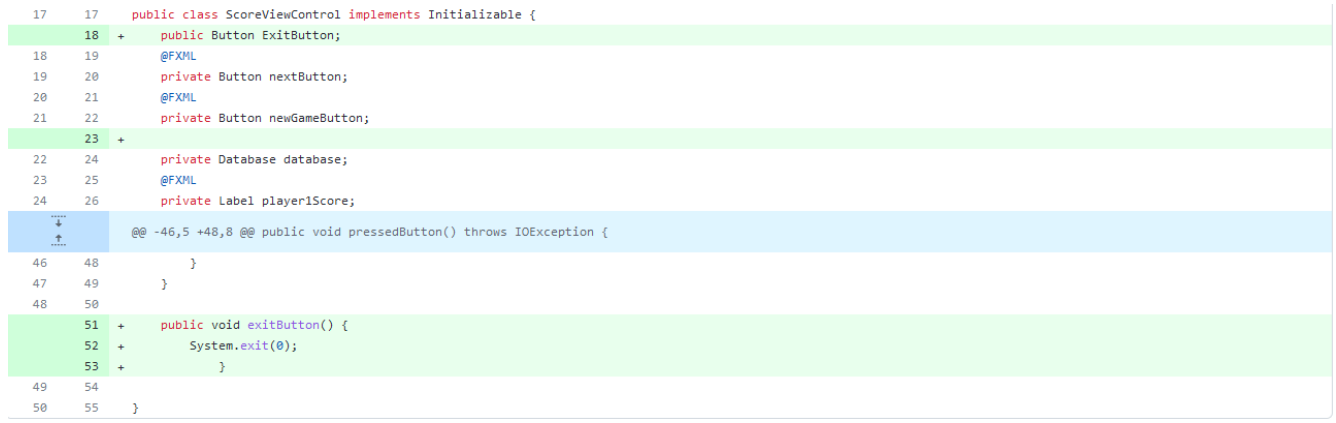
User story 3 formulering:

”As a user, I want to be able to end and turn off the game when I want to. Then I have the option to turn off the program and not play any further.”

Spelare behöver inte spela förevig, och kan avbryta spel på ett enkelt sätt: tryck på knappen EXIT.

För att lösa uppgiften började jag med att först använda det visuella redigeringsprogrammet igen. Där skapade jag en knapp, som jag kallade "EXIT", placerade den på rätt plats, valde en färg för den, och angav var den skulle aktiveras. Efter det, i programmet, skrev jag ett speciellt kommando som kommer att meddela till dator att den kan stänga av programmet när spelaren klickar på "EXIT" knappen.

```
5 src/main/resources/Game/ScoreView.fxml
@@ -33,5 +33,10 @@
                <Font name="Comic Sans MS" size="24.0" />
             </font>
          </Text>
+         <Button fx:id="ExitButton" layoutX="41.0" layoutY="383.0" mnemonicParsing="false" onAction="#exitButton" prefHeight="27.0" prefWidth="81.0" style="-fx-background-color: #ce77d1;" text="Exit">
+            <font>
+               <Font name="Comic Sans MS" size="13.0" />
+            </font>
+         </Button>
       </children>
    </AnchorPane>
```



```
public class ScoreViewControl implements Initializable {
+    public Button ExitButton;
     @FXML
     private Button nextButton;
     @FXML
     private Button newGameButton;
+
     private Database database;
     @FXML
     private Label player1Score;
@@ -46,5 +48,8 @@ public void pressedButton() throws IOException {
         }
     }

+    public void exitButton() {
+        System.exit(0);
+            }

 }
```

3. Avslutning

När vi delades in i grupper, började vi arbeta med projektet. Vi har skapat en särskild Discord-grupp för regelbundna möten utanför lektionstid. Sedan registrerade vi oss alla för Trello board, blev bekanta med plattformen och började arbeta där.

Trello var det första steget mot att använda agila metoder. Jag tycker att det är bekvämt att jobba i Trello. Vi delade in uppgiften i små delar och arbetade successivt med varje del. Vi har skapat flera kolumner som: "product backlog" -> "ready" -> "developing" -> "code review" -> "done" i dem placerade vi vad som behöver göras för projektet som helhet och vad vi arbetar med för tillfället och flyttade sedan de avslutade delarna av arbetet till kolumnen "done", här kan vi mycket tydligt se hur många uppgifter som har redan genomförts och hur mycket som återstår att göra och se att vissa uppgifter prioriteras, vilket innebar att de måste göras i första hand för att kunna fortsätta arbeta med projektet.

Vår grupp fungerade väldigt bra tillsammans tycker jag, jag kände mig som en riktig medlem i gruppen, vi kunde diskutera olika saker med varandra och ta framåt våra egna åsikter och få hjälp vid behov utan problem.

För att hinna med eftersläpningen gjorde vi först det som verkligen behövdes och återgick senare till resten av de mindre viktiga detaljerna i programmet. Vi valde en uppgift för varje deltagare bland prioriteringarna och påbörjade arbetet. När uppgiften var klar rapporterade vi den i gruppen och skickade den till Github. Vi alla gjorde kommit, pull requests och merge så ofta som möjligt till develop branch. Vi flyttade fixat user story till avsnittet "done" och tog nästa uppgift från utvecklingskolumnen, den som var viktigare.

Det hände oftast att var och en tog den uppgift han ville ha bland de som var nödvändiga för genomförandet. Ibland under arbetets gång ställdes vi inför det faktum att vi inte visste hur vi skulle genomföra uppgiften, sedan försökte vi alla hitta den nödvändiga informationen, och de som kunde ta reda på det och förstå hur vi skulle arbeta vidare, implementerade en ny metod i programmet och förklarade sedan för alla i gruppen hur det fungerar. Det var väldigt lärorik. Marcus var ledare i våran grupp.

I början, försökte alla göra allt på egen hand i sin uppgift, och det tog tid, men senare slutade vi vara rädda för att be om hjälp och det gick bra, påskyndade arbetet och fick gruppen att dra ihop sig, tror jag. Vår grupp kommunicerade utan konflikt, alla var redo att hjälpa till om det behövdes och vi behandlade varandra med stor respekt.

Vi träffades ofta och diskuterade uppgifter och hur man implementerar det i kod, delade skärm och arbetade tillsammans. Under kodgranskningen diskuterade och föreslog vi möjliga ändringar för att till exempel förbättra programmets utseende eller funktionalitet. Alla diskussioner var konstruktiva och vi kom alltid fram till en gemensam uppfattning. Det var ett bra beslut att träffas ofta och lösa problem tillsammans, det påskyndade arbetet rejält och vi lärde oss av varandra. Vi tror att vi har jobbat mest med Kanban och lite Scrum.