**Dice Game**

**Kurs:** [Objektorienterad programmering 1](https://newton.learnpoint.se/GroupForms/Group_LearningContent_Item.aspx?Id=2391&ItemId=922)

**Termin och år: ht 2019**

**Författare: Manjit Singh**

**Lärare: Jerry**

1. Sammanfattning

Projektuppgift del 2 handlar om att använda våra JavaFX kunskaper i ett faktiskt program och sättet jag har valt att tillämpa mina JavaFX kunskaper är att skapa ett program som kastar tärningar och därefter jämföra värdena av kasten för att bedöma vem som har kastat högst. Tyvärr så hann jag inte tillämpa alla mina krav men programmets grundfunktionalitet finns där och kan enkelt expanderas.

Innehållsförteckning

[1. Sammanfattning 2](#_Toc25249316)

[2. Inledning 4](#_Toc25249317)

[3. krav 4](#_Toc25249318)

[4. Huvud algoritm 4](#_Toc25249319)

[5. Wireframe 5](#_Toc25249320)

1. Inledning

Till denna projektuppgift har jag fått dispans att jobba själv och då valt att göra ett enkelt GUI program som kastar 5st tärningar och sedan bedöma vilken av de 5 tärningarna som är högst. Alternativt att använda programmet som ett tillbehör till ett spel som t.ex. Yatzee för hur ofta tappar man inte bort tärningar?

1. krav

I och med att tanken med projektuppgiften är att få förståelse om JavaFX samt verktygen runt så har jag valt att sätta krav för egenutveckling då jag är själv i detta arbete.

Mina krav:  
programmet skall hållas simpelt då det skall användas som ett verktyg till spel.  
kunna skifta mellan 2 scener.  
gärna skalbart upp till 5 tärningar.  
En algoritm för att identifiera vinnaren av 5 tärningar.  
Byggas i scenebuilder för att sedan kunna användas i egna program för enklare GUIn.  
Använda bilder på tärningar stället för nummer.  
Använda mig av flera datorer så att jag gör mig mer bekant med Git och hur jag ska hantera koden när den utvecklas på flera datorer.

1. Huvud algoritm

Koden Nedanför visar hur jag identifierar vinnaren samt säkerställer att det inte finns en annan spelare med samma värde.

public int findWinner(int[] roll){  
  
 int biggestValue = 0; //initialize a compare value  
 int arrayIndex =0; //initialize a variable to keep the biggest value in the arrays position  
 for(int i=1;i<6;i++){ //First loop to find the biggest value of the 5 rolls  
  
 if(roll[i]>biggestValue){ //if the new value is bigger than the current biggest value enter if  
 biggestValue = roll[i]; //update the biggest value  
 arrayIndex = i; //save the position of the biggest value in array  
 }  
 }  
  
 for(int i= 1;i<6;i++){ //Second loop to find if the biggest value is alone in the array  
  
 if (roll[i]==biggestValue && i != arrayIndex){ //if there is a second value equal to the biggest value enter loop  
 return 6; // return value 6 which will be handle as TIE in the program  
 }  
  
 }  
  
 return arrayIndex; //if the biggest value was solo return the position of the biggest value in array

1. Wireframe

