Lab 1 – Tärningspel

Godkänd-del

Uppgift 1 - Die

I den första uppgiften skall vi modellera en tärning med hjälp av en klass Die. Vi skall sedan använda den här tärningsklassen för att skapa ett enkelt tärningsspel.

Tärningen skall ha två variabler: nuvarande värde och antal sidor (dvs max-värde för tärningen).

- Antal sidor skall sättas i klassens konstruktor. Skall vara av typen **int**. Skall ha en gettermetod.
- Nuvarande värde skall förändras när man kallar på roll()-metoden (se nedan). Skall vara av typen **int**. Skall ha en getter-metod.

Tärningen skall också ha en privat klassvariabel (dvs statisk): en slumpgenerator.

- Slumpgeneratorn skall vara av typen Random och kommer att användas i vår roll()- metod.
    - Det räcker med en kopia av Random-klassen som kan delas av alla instanser - det är därför den här variabeln är static.

För att kunna använda klassen kommer vi att behöva en metod:

- public void roll();
    - Den här metoden skall använda sig av slumpgeneratorn för att förändra tärningens nuvarande värde.
    - Tips: ni bör använda standardklassen Random för detta. Random har en metod **public int nextInt(int maxValue)** som returnerar ett slumpmässigt heltal mellan 0 och (maxValue-1).

Uppgift 2 – Player

För att kunna göra ett simpelt tärningsspel skall vi nu också skapa en klass **Player** för att representera en spelare.

En spelare skall ha tre instansvariabler: ett namn, en poäng och en lista av tärningar.

- Namnet skall vara av typen **String**. Ska ha en getter-metod.
- Poängen skall vara av typen **int** och skall representera hur många poäng spelaren har lyckats samla ihop. Ska ha en getter-metod.
- Listan med tärningar är vad spelaren kommer att använda för att spela tärningsspelet. Skall vara av typen **ArrayList<die>.**

En spelare skall också ha ett antal metoder:

- **public void rollDice();**
  - Skall rulla alla tärningar i spelarens tärnings-lista.
- **public int getDieValue();**
  - Skall summera och returnera värdet på spelarens alla tärningar i form av ett heltal.
- **public void increaseScore();**
  - Skall öka spelarens poäng med ett.
- **public void addDie(int sides);**
  - Skapar en ny tärning med sidor sides och lägger till den till spelaren.

Väl Godkänd-del

Uppgift 3 – SimpleDiceGame

Vi skall nu skapa vårt enkla tärningsspel. Skapa en ny klass **SimpleDiceGame**. Denna klass skall kunna köras, så vi behöver en **main()**-metod. Nedan följer en beskrivning av hur spelet ska se ut, samt några metoder ni ska skapa. Ni måste sedan själva skriva main-metoden utifrån den här informationen, dvs ni måste klura ut hur allting hänger ihop själva!

Beskrivning av spelet

Programmet frågar först hur många spelare som vill spela, hur många tärningar varje spelare skall ha och hur många sidor tärningarna skall ha. Det frågar sedan efter namnen på spelarna. Ni kan anta att användaren matar in korrekta värden, dvs ni behöver ej ha med felhantering. Spelet kommer sedan att spelas i fem omgångar. Varje omgång består av att spelarna, en efter en, gissar på ett värde och sedan rullar sina tärningar. Om värdet spelaren gissade på är lika med det sammanlagda värdet på tärningarna, så får spelaren en poäng. En omgång är slut då en sista spelaren har gissat och rullat sina tärningar. Spelaren som har flest poäng efter 5 omgångar vinner spelet.

Hjälpmetoder
För att förenkla det här skall vi skapa några hjälpmetoder:

- **private static ArrayList initialize();**
    - Skall initialisera spelet genom att ta emot användarinput och skapa alla instanser som behövs, samt en lista av spelare som sedan returneras.
- **private static void takeTurn(ArrayList players);**
    - Skall ta emot en lista av spelare och spela färdigt en hel omgång. Den skall gå igenom listan av spelare, rulla varje spelares tärningar, fråga efter en gissning och öka spelarens poäng om hen gissat rätt.
- **private static ArrayList getWinners(ArrayList players);**
    - Tar emot en lista av spelare och skall returnera en lista med vinnare. En vinnare är man om man har mest poäng. Vi har en lista eftersom flera spelare kan vinna, då flera spelare kan ha samma poäng.