dicegame

Tärningspel - Dokumentation

Detta program är skapat med hjälp av fyra olika klasser:

- Die
- Player
- GameFunctions
- Dicegame (main)

Fördelen med klasser och metoder är att gömma undan kod så att main-metoden blir mer lättöverskådlig. Jag kommer förklara lite kort om varje klass.

Die

I Die-klassen skapar vi vår tärning och möjligheten att spara alla variabler som har med tärningen att göra, så som vilka värden och hur många sidor den ska ha. För att underlätta skapar vi en metod med standardvärden och med hjälp av en scanner senare i main-metoden kan vi modifiera värdena utifrån vad spelaren väljer.

I och med att detta program går ut på att spelaren ska gissa ett nummer beroende på de parametrar som ställs så behöver vi en funktion som slumpar en siffra och det gör vi genom den importerade klassen Random och lägger till den i vår klass och skapar ett objekt.

CurrentValue() använder vi oss av för att hämta det slumpmässigt genererade värdet som behövs för att kunna jämföra om spelaren gissat rätt

Sides() behöver vi för att veta hur många sidor tärningen ska ha.

Roll()-metoden rullar tärningen med hjälp av randomizern och sparar resultatet i CurrentValue.

Player

I player-klassen sparar vi undan all information som har med spelaren att göra. Vi börjar med att deklarera ett par variabler som vi behöver till vår spelare. Efter det skapar vi metoder för att kunna referera till och modifiera under spelets gång.

Vi frågar efter spelarens namn och sparar det i “name”. Därefter vill vi också kunna skriva ut det och använder oss därför av metoden “getName()” som returnerar “name”.

För att i slutet kunna återge resultatet av spelet använder vi oss av “getPoint();” som har poängen sparade under variabeln “point”.

setPoint() i detta fall använder vi för att modifiera värdet av “point”.

I programmet vill vi också kunna ge feedback efter varje rullning så vi behöver jämföra random-objektets resulutat mot spelarens gissning som sparas under “guess”.

addDie(int sides) Här skapar vi vår tärning till spelaren med konstruktorn “sides” som bestämmer hur många sidor tärningen ska ha och som väljs av spelaren.

När vi har allt vi behöver så är det dags och rulla tärningen och det gör vi med “rollDice()”.

Efter att tärningen har rullats med hjälp av random funktionen sparas resultatet i “CurrentValue()”.

Nu har vi ett värde på tärningen samt spelarens gissning och kan jämföra resultatet. Om spelaren gissat rätt använder vi “increaseScore()” för att öka poängen med +1.

GameFunctions

Syftet med denna klass är för att gömma undan mycket av den kod som fanns i main så dokumentationen blir lite konstig när 90% av den redan va klar när jag bestämde mig för att städa i den. Jag kommer beskriva kort vad varje metod gör.

guessEvaluations() = Denna metod tar spelarens gissning och jämför den spelarens slumpmässigt rullade tärning och skriver ut ett meddelande.

displayScore() = Denna metod skriver ut ett av tre olika meddelanden beroende på hur många poäng spelaren fick.

gameStart() = Själva kärnan i programmet som itererar beroende på antal rundor spelaren valt. Det finns också en try-carch funktion som hindrar spelaren från att ange felaktiga värden. Här finns även metoden att rulla tärningen och guessEvaluation som behöver ingå i loopen. Jag försökte lägga till replay-funktionen här men fick inte till det. 😟

Main

I main-metoden exekverar och skriver vi själva programmet. Vi börjar med att importera en Scanner för att kunna ta emot inmatning av användaren.

I början av programmet frågar den efter ett namn. Efter det skapas en ny spelare, som refererar till Player-klassen, med namnet som valts.

Därefter frågar programmet hur många rundor spelaren vill spela och hur många sidor tärningen ska ha. Dessa värden sparas under “rounds” respektive “sides”.

Nu ber vi spelaren gissa på ett nummer under varje iteration. Detta gör vi med hjälp av en for-loop som loopar baserat på antalet rundor angivet. Här finns det även en try-catch funktion som hjälper oss att fånga upp om spelaren skriver in ogiltiga värden. (siffra större eller mindre än antal sidor, bokstäver eller andra tecken). “InputMismatchException” låter oss tala om för programmet vad som ska hända om ett ogiltigt värde upptäckts. Vi behöver även lägga try-catch funktionen i en while-loop och ange den som true för att loopen ska fortsätta köras tills boolean säger annat.

Tärningen rullas.

Under spelets gång ska det ges feedback på spelarens gissningar och det gör vi genom en att använda en if-sats. Först jämför vi om spelarens gissning är samma som värdet på tärningen och om så är fallet tilldelas en poäng.

När det är dags att presentera resultatet finns det tre olika meddelanden som kan skrivas ut beroende på hur många poäng spelaren fick. Ännu en gång med hjälp av if-satsar bestämmer vi vilket meddelande som ska skrivas ut. Det finns tre gränsvärden: >50%, <50% (0.5*rounds) och 100%.

Slutligen finns det även en replay-funktion! Den funkar som så att man skapar en while-loop av den kod man vill köra igen, i detta fall 95% av koden förutom där vi deklarerar våra variabler och vi förmodar att spelarens namn är förblir oförändrad. För att ge spelaren möjligheten att spela igen frågar vi i slutet av programmet om de vill spela igen. Detta gör vi med hjälp av en if-sats. Precis innan deklarerar vi “playAgain” av typen String och initierar den med input från våran Scanner. Om spelaren svarar “yes” (“equalsIgnoreCase” löser så att input inte påverkas av små eller stora bokstäver) kommer programmet gå tillbaka till början av while-loopen. Om svaret är “yes” behöver vi återställa poängen till noll och det gör vi med “setPoint()”. Om svaret är “no” kommer programet gå vidare och avslutas. Om spelaren svarar något annat får de ett felmeddelande och för att programmet inte ska gå vidare här ifrån så måste vi loopa till svaret blir antingen “yes” och då avslutar vi denna loop med en “break;”

Hur man spelar dicegame

- Programmet välkomnar dig och ber dig skriva in ditt namn. Ange önskat namn.
- Programmet frågar hur många rundor du vill spela. Ange antal rundor i sifrror.
- Programmet frågar hur många sidor tärningen ska ha. Ange antal sidor i siffror.

- Programmet ber dig gissa på ett nummer beroende på antal sidor på tärningen. Ange i siffror. Programmet kommer att upprepa sig själv beroende på antal rundor som valts. Feedback återges direkt om du gissat rätt eller fel.
- Till slut kommer programmet att presentera resultatet och du kan välja om du vill spela igen. Om ja så kommer programmet att starta om och poängen återställas. Om nej så kommer programmet att avslutas.