Laboration nr 1

Syfte

Syftet med denna laboration är få en första inblick i objektorienterad programmering i Java.

Redovisning

Källkoden för uppgifterna skall skickas in via Fire. *Posta endast rena källkodsfiler, alltså de som har filändelse* **`.java`**.

Uppgift 1

Din uppgift är att nyttja standardklassen java.util.Random för att skriva en klass Die som modellerar en vanlig 6-sidig tärning.[1] Det man skall kunna göra med en tärning är dels att kasta tärningen, dels att avläsa vilket värde tärningen har (dvs vilken sida på tärningen som ligger upp). Kalla metoderna som utför dessa operationer roll respektive getDots:

public void roll()	kastar tärningen
public int getDots()	returnerar tärningens värde

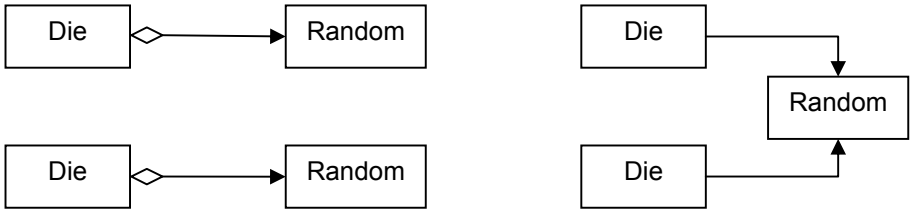
Att kasta en tärning innebär att slumpen skall avgöra vilken sida på tärningen som kommer upp. Därför behöver klassen Die ha en slumptalsgenerator (dvs ha ett objekt av klassen Random), som kan användas vid implementationen av metoden roll.

Klassen Die skall också innehålla en konstruktor utan parametrar (= den tomma konstruktorn):

public Die()	initierar en ny tärning med ett slumpmässigt värde

Att fundera på:

- Bör objektet av klassen Random lagras i en instansvariabel eller i en klassvariabel? Vad händer om två tärningsobjekt har var sitt slumptalsgeneratorobjekt (den vänstra figuren nedan)? Kommer de att producera oberoende utfall? Vore det lämpligare om flera tärningsobjekt *delar* en gemensam slumptalsgenerator (den högra figuren)? Vad blir skillnaden?



- Varför bör klassen inte ha en metod setDots?
- Bör klassen ha fler konstruktorer?
- Bör klassen ha fler metoder?

[1] sing. *die*, pl. *dice* el. *dies*, tärning.

Uppgift 2

I tärningsspelet *Craps* används två sexsidiga tärningar. Innan varje spelrond görs en insats. Det första kastet i en spelrond kallas "the come out roll". Om "the come out roll" blir 7 eller 11 är det automatiskt vinst och om "the come out roll" blir 2, 3 eller 12 är det automatiskt förlust. Om däremot "the come out roll" ger 4, 5, 6, 8, 9 eller 10 blir denna poäng "the point" och spelet fortsätter. Spelaren kastar nu tärningarna antingen tills 7 kommer upp, vilket innebär förlust, eller tills "the point" kommer upp, vilket innebär vinst.

Skriv ett program som simulerar tärningsspelet Craps. Gör 1000 spelomgångar och skriv ut hur många gånger spelaren vinner respektive förlorar.

Uppgift 3

Utgå från koden i klassen Die och skriv en ny mer generell klass GeneralDie för att handha tärningar med ett *godtyckligt antal sidor*.

Att fundera på:

- Behöver GeneralDie ha fler instansvariabler än Die (dvs behöver vi hålla reda på något mer än vilken sida av tärningen som ligger upp då vi har en tärning med godtyckligt antal sidor)?
- Vilka konstruktorer bör GeneralDie ha?
- Behöver GeneralDie ha några fler metoder?
- Vad betyder egentligen *ett godtyckligt antal sidor*? *Vilka antal är meningsfulla?*

Uppgift 4

Använd klassen GeneralDie för att skriv ett program som genom simulering beräknar sannolikheten för att få en *kåk* (*eng.* full house) när man kastar 5 tärningar, där de enskilda tärningarna har 6, 7, 8, 9 respektive 10 sidor. En kåk betyder att endast två olika värden förekommer på de fem tärningarna, och att det ena värdet förekommer på tre av tärningarna och det andra värdet på de övriga två tärningarna.

Ett fält skall användas för att lagra tärningarna.

Tips:

För att avgöra om värdena på tärningarna bildar en kåk är det enklast att först lagra värdena i ett fält och sedan sortera fältet. I klassen java.util.Arrays finns metoden sort som kan användas vid sorteringen.