RAFLEBÆGER 2 (DIECUP 2)

I denne opgave skal I arbejde videre med jeres raflebægerprojekt.

Start med at tage en kopi af jeres projekt fra sidste opgave (inklusiv de feltvariable og metoder, som I implementerede der). Så har I altid originalen at falde tilbage på, hvis noget går helt i koks.

Opgave 1

Det er ofte hensigtsmæssigt at lave en separat klasse, som udelukkende bruges til at teste de klasser og metoder, som man er i færd med at implementere.

Opret klassen ***TestDriver*** og implementer nedenstående metode.

void test()

Metoden skal skabe et raflebæger med to terninger, foretage et kast med bægeret og dets to terninger, for til sidste at udskrive antallet af øjne på terminalen ved hjælp af ***System.out.println*** metoden, der blev beskrevet i en af forelæsningerne.

Hvis I foran metodens navn skriver ***public static*** i stedet for blot ***public***, kan metoden kaldes ved at højreklikke på den lysebrune boks, der repræsenterer ***TestDriver*** klassen. Det er lettere end først at skulle skabe et ***TestDriver*** objekt og så kalde metoden ved at højreklikke på den røde boks, der repræsenterer dette objekt. I slipper for at skabe et ***TestDriver*** objekt, som I ikke har andet at bruge til end at kalde ***test*** metoden.

Opgave 2

I ***TestDriver*** klassen skal I dernæst implementere nedenstående metode

void testMultiple(int noOfRolls)

Hvis I løste opgave 6 i Raflebæger 1, har I allerede løst store dele af indeværende opgave, og I kan derfor starte med at flytte metoden ***multipleRolls*** fra ***DieCup*** klassen til ***TestDriver*** klassen, ændre dens navn og så arbejde videre på det, som I allerede har lavet.

Metoden skal skabe et raflebæger med to terninger, foretage ***noOfRolls*** kast med bægeret og dets to terninger og udskrive resultatet af disse kast på terminalen (med en linje for hvert kast). Til sidst udskrives det gennemsnitlige antal øjne i de foretagne kast, således at terminalen får et udseende, der svarer til nedenstående, hvor ***noOfRolls*** er sat til 5.

```
Throw no 1: 8
Throw no 2: 12
Throw no 3: 7
Throw no 4: 3
Throw no 5: 6
Average no of eyes: 7.2
```

Når man dividerer to heltal med hinanden rundes ned til nærmeste heltal, dvs. at 36/5 evalurer til heltallet 7. Hvis man i stedet vil have et reelt tal, kan man skrive 1.0 * 36/5, hvilket evaluerer til det reelle tal 7.2.

Også for denne metode vil det være en fordel for jer at skrive ***public static*** i stedet for ***public***, således at I ikke behøver at skabe et ***TestDriver*** objekt for at kunne kalde metoden interaktivt.

Kald ***testMultiple*** et antal gange med ***noOfRolls*** sat til 1000 og disktutér, hvordan resultatet varierer. Hvis I ikke kan se alle linjerne vælges *Unlimited buffering* i den *Options* menu, som findes i terminalens øverste venstre hjørne.

Opgave 3

I skal nu generalisere situationen, således at terninger kan have et vilkårligt antal sider (større end eller lig med 2). Antallet af sider angives som en heltalsparameter ***noOfSides*** til ***Die*** klassens konstruktør, som bruger parameteren til at initialisere en ny feltvariabel ved navn ***sides***.

Modificér konstruktøren for ***Die*** klassen, som beskrevet ovenfor. Brug en ***if*** sætning til at tjekke, at konstruktørens parameter ***noOfSides*** har en fornuftig værdi, dvs. er et heltal større end eller lig med 2. Hvis det ikke er tilfældet, udskrives en passende besked på terminalen.

Modificér ***Die*** klassens ***roll*** metode, således at denne nu sætter feltvariablen ***eyes*** til at være et tilfældigt heltal i intervallet ***[1, sides]***.

Er der behov for at modificere ***Die*** klassens ***getEyes*** metode?

Opgave 4

Modificér konstruktøren for ***DieCup*** klassen, så den får to parametre, hvormed man kan angive, hvor mange sider de to terninger hver især skal have.

DieCup(int sides1, int sides2)

Er der behov for at ændre nogle af ***DieCup*** klassens metoder?

Opgave 5

Forsyn de to metoder i ***TestDriver*** klassen med to ekstra parametre ***sides1*** og ***sides2***, således at man i kaldet kan angive, hvor mange sider de to terninger skal have. Afprøv de nye metoder på nogle forskellige raflebægre. Diskutér om resultaterne ser fornuftige ud.

Opgave 6

I skal nu afprøve det, som I har lavet i opgave 3 og 4, ved at kalde klassemetoden ***test*** i ***TestServer*** klassen med parameteren "DC2". Dette gøres ved at højreklikke på ***TestServer*** klassen, vælge ***test*** og indtaste "DC2" i den dialogboks, der kommer op (husk anførelsestegnene). Metoden uploader jeres progjekt til vores testserver.

Hvis testserveren finder fejl, skal I gennemgå jeres kode og forsøge at rette dem.

Opgave 7 (til dem der har mod på mere)

Implementer nedenstående metode i ***TestDriver*** klassen.

void compareDieCups(int s1, int s2, int s3, int s4, int noOfRolls)

Metoden skal skabe to raflebægre, hvor det første har terninger med ***s1*** og ***s2*** sider, mens det andet har terninger med ***s3*** og ***s4*** sider. Dernæst foretages ***noOfRolls*** kast med de to raflebægre, og det udskrives, hvordan raflebægrene ser ud og hvor mange gange hvert af de to raflebægre vinder på nedenstående form:

DieCup 1 with 6 and 6 sides is highest: 40 times
DieCup 2 with 8 and 4 sides is highest: 47 times
Same score in both: 13 times

Brug metoden til at undersøge, hvem der oftest vinder: et raflebæger, hvor begge terninger har 6 sider, eller et raflebæger hvor den ene terning har 8 sider, mens den anden har 4 sider. Diskutér om resultatet er som forventet.