Introduktionslaboration i Java-programmering

Målsättningen med denna uppgift är att komma igång med Java och bekanta sig med några grundläggande Java-idiom. Du kan *inte* redovisa mål med den här uppgiften, och inte heller använda den som en av de två som ska redovisas med mål Z70.

1. Skapa en arbetskatalog med ett lämpligt namn (till exempel `die`) i din `ioopm`-katalog.
2. Kopiera programmet `Die.java` från de utdelade filerna till din katalog.
3. Kompilera programmet med kommandot `javac Die.java`. Detta skapar en fil i den aktuella katalogen; använd `ls` för att ta reda på dess namn. Prova sedan att köra programmet med kommandot `java Die`.
4. Skriv en annan klass med namnet `MyDieTest` (i en fil som heter `MyDieTest.java` i samma katalog). Klassen skall (endast) innehålla en `main`-metod som frågar användaren om önskat antal sidor och sedan skapar ett sådant `Die`-objekt, slår det 10 gånger och beräknar och skriver ut summan i terminalen.
5. Följande kod är nonsenskod som inte producerar ett vettigt resultat (försök att fundera ut varför innan du skriver ett program som utför uttrycken nedan och ser resultatet):

   ```
   Die die = new Die();
   System.out.println(die.get());
   ```

   Vad är problemet/felet? Ändra i `Die.java` så att ovanstående kod blir meningsfull (”vettig”)!
6. Följande kod är heller inte vettig:

   ```
   Die die = new Die(-12);
   ```

   Ändra i `Die.java` så att det inte går att skapa tärningsobjekt med ett orimligt antal sidor!
7. Vad ger koden

   ```
   Die d = new Die();
   System.out.println(d);
   ```

   för utskrift? Lägg till följande metod i `Die`-klassen:

   ```
   String toString() {
     return "Die(" + value + ")";
   }
   ```

   och se vad det då blir för utskrift. Fundera över varför!
8. Lägg till en metod med signaturen **`boolean`** `equals(Die otherDie)` som returnerar **`true`** om tärningarna ”är lika” (vad betyder det?), annars **`false`**.
9. Skriv en klass `PairOfDice` som representerar ett tärningspar. Klassen skall använda sig av (”aggregera” med OO-terminologi) klassen `Die` , d.v.s. den skall ha två attribut av typen `Die` och metoderna i `PairOfDice` skall använda metoder i klassen `Die`.

   Operationer som skall finnas:

   - Skapa ett tärningspar med givet antal sidor (samma för båda tärningar),
   - slå ett tärningspar,
   - avläsa varje individuell tärning samt
   - en `toString`-metod.

10. Skriv en klass `ColoredDie` som representerar en tärning som har en viss färg (låt färgen representeras som en sträng). Låt den ärva av klassen `Die`. Vilka metoder bör du specialisera (eng. ”override”)? Vilka nya metoder är vettiga att ha? Ändra sedan i klassen `PairOfDice` så att en av de skapade tärningarna är röd.

11. Metoden `equals` i steg 8 har problemet att den bara kan anropas med en tärning som argument. Med andra argument anropas istället motsvarande metod i klassen `Object`, som kanske inte gör vad vi vill. För att specialisera `equals` måste den ha typen **`boolean`** `equals(Object other)`. Skriv om `equals` så att den kan anropas med argument av godtycklig typ. Du kan använda operatorn **`instanceof`** för att ta reda på om ett värde är (en subtyp av) en viss klass (men det här är i princip det enda vettiga tillfället att använda **`instanceof`**!).

12. Gå till online-dokumentationen för Javas klass-API (JDK) och skumma dokumentationen för `String`-klassen. Läs om `compareTo`-metoden och skriv sedan ett program som läser in två namn i strängform (med hjälp av `Scanner`-klassen) och skriver ut dem i bokstavsordning.

13. [**Frivilligt, men rekommenderas (Speciellt till dig som redan kan programmera Java)!**] Skriv ett lämpligt driver-program som kapslar in nedanstående kod i en `main`-metod, och förklara beteendet.

```
String a = "Beefheart";
String b = "Beefheart";
String c = "heart";
String d = "Beef" + c;
int i = 1;
int j = 1;
Integer k = 2;
Integer l = i + j;
System.out.println(a == b);
System.out.println(a.equals(b));
System.out.println(a == d);
System.out.println(a.equals(d));
System.out.println(i == j);
System.out.println(k == i + j);
System.out.println(k.equals(i+j));
System.out.println(k == l);
System.out.println(k.equals(l));
```