Övning 2.

Uppgift 1

Skriv en klassmetod minGap som tar ett heltalsfält som parameter och returnerar den minsta skillnaden mellan två intill liggande element i detta fält.

Exempel:

Antag att följande deklaration har gjorts

```
int[] vekt =  {1, 4, 6, 9, 12};
```

Ett anrop av minGap(vekt) returnerar värdet 2. Skillnaden mellan 1 och 4 är 3, skillnaden mellan 4 och 6 är 2, skillnaden mellan 6 och 9 är 3 och skillnaden mellan 9 och 14 är 5. Den minsta av dessa skillnader är 2.

I din lösning får du anta att fältet som ges som parameter innehåller minst 2 element.

Uppgift 2

Skriv en klassmetod percentEven som tar ett heltalsfält som parameter och returnerar hur många procent av elementen i fältet som är jämna tal.

Exempel:

Antag att följande deklaration har gjorts

```
int[] vekt =  {1, 4, 6, 9, 12};
```

Ett anrop av percentOfEven(vekt) returnerar värdet 60.0.

Uppgift 3

Skriv en klassmetod swapAll som tar två heltalsfält och byter elementen i fälten.

Exempel:

Antag att följande deklaration har gjorts

```
int[] a = {11, 42, -5, 27, 0, 89};
int[] b = {10, 20, 30, 40, 50, 60};
```

Ett anrop av swapAll(a, b) skall fältet a innehålla elementen {10, 20, 30, 40, 50, 60} och fältet b elementen {11, 42, -5, 27, 0, 89}.

a) Du får anta att fälten som ges som parametrar till metoden swapAll inte är **null** och att de är lika långa.

b) Förklara varför du måste göra antagandena som görs i deluppgift a).

Uppgift 4

Ett *anagram* är ett ord som bildas genom att kasta om bokstäverna i ett annat ord. Till exempel är ordet "katt" ett anagram av "takt" och ordet "allt" är ett anagram av "tall".

Skriva en metod

```
public static boolean isAnagram(String word1, String word2)
```

som tar två strängar word1 och word2. Metoden skall returnera **true** om strängarna word1 och word2 är anagram, annars **false**. Metoden skall inte göra skillnad mellan stora och små bokstäver, dvs orden "katt" och "Takt" skall betraktas som anagram.

Uppgift 5

Skriv en metod

public static double[] removeFaulty(**double**[] items, **double** correctSize, **double** tolerance)

som tar ett fält med reella tal items samt två reella tal correctSize och tolerance. Metoden returnerar ett nytt fält som innehåller de värden i fältet items som avviker från värdet av corretSize med mindre än tolerance.

Exempel:

Antag att följande deklaration har gjorts

double[] bolts = {50.01, 50.02, 49.99, 50.03, 50.01, 50.00, 49.99};

Ett anrop av removeFaulty(bolts, 50.0, 0.02) skall returnera fältet

{50.01, 49.99, 50.01, 50.0, 49.99}

Uppgift 6

Klass GenericDie som används för att avbilda tärningar med ett godtyckligt antal sidor finns att tillgå. Klassen innehåller följande konstruktorer och metoder:

public GenericDie()	konstruktor som ger en 6 sidig tärning
public GenericDie(**int** nrOfSides)	konstruktor som ger en tärning med nrOfSides sidor
public void roll()	kastar tärningen
public int getValues()	returnerar värdet av tärningen
public int getSides()	returnerar hur många sidor tärningen har

a) Skriv en metod

public static int[] roll(GenericDie theDice, **int** nrRoll)

somen tärning theDice av klassen GenericDie, kastar denna tärning nrRoll antal gånger och returnerar ett heltalsfält som innehåller hur många gånger var och en av valörerna på tärningen kommit upp.

Tips: Fältet som metoden roll returnerar innehåller lika många element som det finns sidor på tärningen theDice. I klassen GenericDie finns en metod för att ta reda på antalet sidor på tärningen.

Valörerna på en tärning är numrerade från 1 till max, medan indexen i ett fält är indexerade från 0 till max-1. Var i fältet är det lämpligt att lagra valören 1, valören 2, . . , valören max?

b) Använd klassen GenericDie för att skriva ett program som genom simulering beräknar sannolikheten för att få en kåk (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 två av tärningarna. En metod skall användas för att utföra själva simuleringen och ett fält skall användas för att handha 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 kan använda vid sorteringen.