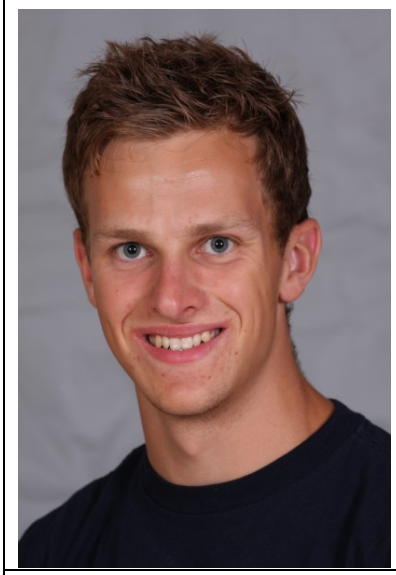
Mads Friis Bornebusch, s123627
Kristian Sloth Lauszus, s123808
Simon Patrzalek, s123401

3. marts 2013

Afleveringsopgave 2

02102 Indledende Programmering

Mads Bornebusch s1233627



Kristian Sloth Lauszus s123808

Simon Patrzalek s123808

Rapporten og øvelsen har været et samarbejde mellem Mads Friis Bornebusch, Kristian Sloth Lauszus og Simon Patrzalek.
Vi har alle lavet opgaverne hver for sig og har derefter tjekket om vi havde de samme løsninger. I forhold til den første afleveringsopgave var denne noget mere tidskrævende da man skulle udtænke hvordan man skulle lave opgaverne. Det lykkedes os dog alle sammen at få lavet alle opgaverne.

Mads Friis Bornebusch, s123627
Kristian Sloth Lauszus, s123808
Simon Patrzalek, s123401

3. marts 2013

Opgave 1

For at løse denne opgave lavede vi en funktion der hed ”intToRomanNumeral” der stod for konverteringen af det positive tal til en streng som den så returnede.

Dette gjorde vi vha. En række while-loops. Fx ses delen der tjekker om tallet er større end 1000 og 900:

```
while(value >= 1000) {
   string += "M";
   value -= 1000;
}
while(value >= 900) {
   string += "CM";
   value -= 900;
}
```

Det specielle ved konverteringen til romertal er at det gælder specielle regler for 4,9,40,90 osv. Derfor bliver man nødt til at konvertere tallet på denne måde.

Derudover har vi tilføjet en try-catch rundt om nextInt(), da den ellers crasher hvis man indtaster noget der ikke er et tal. Derudover tjekker vi også om tallet er større end 0. Hvis nogle af de to tilfælde er gældende melder programmet en fejl.

Nedenfor vises udskrifter fra programkørsler

Eksempel på en konvertering til romertal:	Eksempel på en forkert brugerindtastning:
`Enter positive integer to convert: 1492` `1492 = MCDXCII`	`Enter positive integer to convert: -300` `Please enter a positive integer.`

Opgave 2

I denne opgave skulle vi skrive en kode der tjekkede om en tekst var et såkaldt palindrom. Funktionen der tjekkede dette kan ses nedenfor:

```
private static boolean checkPalindrome(String input) {
   input = input.toLowerCase().replaceAll("[^a-z]", ""); // Remove everything that is not a letter using a regex
   if(input.equals("")) // Check if the string is empty. For instance if the user inputs only numbers
      return false;
   return new StringBuilder(input).reverse().toString().equals(input); // Check if the input is equal to the
reverse using the StringBuilder class
}
```

Det er værd at bemærke at vi benytter det der hedder en regex til at fjerne alle karakterer der ikke er et bogstav. a-z betyder at den skal kigge på alle bogstaver, men da vi jo netop ønsker at beholde alle bogstaver bruger vi ^ der fortæller den at den skal kigge på alle de karakterer der ikke opfylder dette. Dette kunne fx være tal, mellemrum, kommaer, udråbstegn osv.

Nedenfor er der vist en testkørsel af programmet med et palindrom:

```
Enter line to check: A man, a plan, a canal: Panama.
"A man, a plan, a canal: Panama." is a palindrome!
```

Mads Friis Bornebusch, s123627
Kristian Sloth Lauszus, s123808
Simon Patrzalek, s123401

3. marts 2013

Opgave 3

I den sidste opgave skulle man skrive en kode der fandt en tilnærmelse til π vha. det der kaldes Buffons nål. Denne var lidt mere vanskelig da man lige skulle finde ud af præcis hvad det egentlig gik ud på.

Det mest interessante i denne kode er for-loopet som der laver selve "simuleringen". Denne kan ses nedenfor:

```
for (int i = 0; i < iterations; i++) { // Use the for-loop to run the "simulation" the number of iterations
   double distance = Math.random()*MAXDISTANCE; // Get the distance from the bottom of the needle to the line
   double angle = Math.random()*Math.PI; // Calculate the angle in radians from 0-π
   double number = distance + Math.sin(angle)*LENGTH; // Calculate the opposite side using the sine function
   if(number >= MAXDISTANCE) // If it's larger than the maximum distance then it must be crossing the line
      success++; // Increment the counter
}
```

Først findes et tilfældig distance fra nålens nedre del hen til linjen. Derefter findes en tilfældig vinkel mellem 0-180 grader eller 0-π regnet i radianer. Til sidst bestemmer vi først den modstående katete vha. Sinus-funktionen ganget med længen af nålen. Dette lægges derefter til afstanden fra nålens nedre del hen til stregen. Det lidt ironiske ved denne kode er dog, at vi faktisk benytter π til at estimere π.

Nedenfor er vist eksempler på kørsler af koden. Bemærk at outputtet ved en forkert brugerindtastning er det samme som med romertallene da vi tester inputtet på samme måde:

Her passer estimatet på pi ned til 3 decimaler	Her har brugeren indtastet en forkert værdi
`Enter number of iterations: 10000000` `10000000 / 3183222 = 3.1414711257964414`	`Enter number of iterations: 0` `Please enter a positive integer.`