Laboration 1 - Rekursion

Grundläggande

1. Metoden **print** skriver ut samtliga tal i ett intervall. Beskrivning av metoden:

   Metodens namn: print
   Parameterlista: (int min, int max)
   Returvärde: void
   Halvkod:
   om min<=max
   skriv ut min följt av ” ”. Utskriften ska ske med System.out.print-metoden
   rekursivt anrop med argumenten: (min + 1, max)

   Skriv metoden och testa den med anropen:

```
print( 10, 15 );          // Resultat: 10 11 12 13 14 15
print( 15, 10 );          // Resultat:
print( -3, 4 );           // Resultat: -3 -2 -1 0 1 2 3 4
```

2. Metoden **everySecondReverse** skriver ut vartannat tecken i ett String-objekt baklänges. Metoden har följande parameterlista och halvkod:

   Parameterlista: (String txt, int pos)
   om pos>=0 och pos<txt.length()
   skriv ut tecknet i position pos
   rekursivt anrop med argumenten (txt, pos-2)

   Skriv metoden och testa den med anropen:

```
everySecondReverse( "Student", 6 );         // Resultat: teuS
everySecondReverse( "Lärare", 3 );          // Resultat: aä
everySecondReverse( "Förälder", 17 );       // Resultat:
everySecondReverse( "Barn", -2 );           // Resultat:
```

3. Vad blir utskriften vid anropet `mystery1(10);` ?

```
public void mystery1( int n ) {
    if( n > 0 ) {
        System.out.println( n );
        mystery1( n - 2 );
    }
}
```

4. Vad blir utskriften vid anropet `mystery2( 10, 20 );` ?

```
public void mystery2( int a, int b ) {
    if( a <= b ) {
        System.out.println( a + " + " + b + " = " + (a+b) );
        mystery2( a+1, b-1 );
    }
}
```

5. Skriv den rekursiva metoden **reverse**, vilken ska skriva ut talen i intervallet min – max (se metodhuvud nedan) baklänges , dvs. börja på det andra argumentet. Exempel:

```
reverse( 4, 10 ); // Resultat: 10 9 8 7 6 5 4
reverse( 5, 2 );  // Resultat:
reverse( -2, 1 ); // Resultat: 1 0 -1 -2
```

Komplettera metoden med kod

```
public void reverse( int min, int max ) {
    // komplettera här
}
```

6. Skriv den rekursiva metoden **print**, vilken ska skriva tecknen i en sträng, från en angiven position i strängen till slutet av strängen, t.ex.

```
print( "Student" , 0);                    // Resultat: Student
print( "Student" , 3);                    // Resultat: dent
print( "Malmö högskola!" , 6);            // Resultat: högskola!
print( "Hubert" , 10);                    // Resultat:
print( "Negativ position" , -6);          // Resultat:
```

Komplettera metoden med kod

```
public void print( String txt, int pos ) {
    // komplettera här
}
```

7. Öppna en browser och gå till sidan: **chortle.ccsu.edu/CS151/cs151java.html**
Studera länkarna **Chapter 70** och **Chapter 71**.

8. Metoden *factorial* beräknar multiplikationer från 1 upp till ett bestämt tal, t.ex. 1 · 2 · 3 · 4 · 5 · 6

Returntyp: long
Parameterlista: (long n)
om n <= 1
  returnera 1
annars
  returnera n * rekursivt anrop med argumentet (n – 1)

Skriv metoden *factorial* och testa den med anropen:

```
System.out.println( factorial( 3 ) );       // Resultat: 6
System.out.println( factorial( -3 ) );      // Resultat: 1
System.out.println( factorial( 6 ) );       // Resultat: 720
```

9. Metoden *member* kontrollerar om ett tal (*nbr*) finns i en array. Kontrollen börjar i den position som anges i tredje argumentet (ska vara minst 0).

Returtyp: boolean
Parameterlista: (int nbr, int[] array, int pos)
om pos < 0 eller pos >= array.length
  returnera false
annars om nbr == array[pos]
  returnera true
annars
  returnera rekursivt anrop med argumenten (nbr, array, pos + 1)

Skriv metoden medlem och testa den med anropen:

```
int[] arr = { 23, -45, -20, 10, 8 };
System.out.println( member( 17, arr, 0 ) );        // Resultat: false
System.out.println( member( 23, arr, 0 ) );        // Resultat: true
System.out.println( member( 23, arr, 2 ) );        // Resultat: false
System.out.println( member( 10, arr, 0 ) );        // Resultat:  true
```

10. I anropen nedan är *arr* samma array som i Uppgift 9. Vad blir utskriften vid anropen
    a) `System.out.println( mystery3( arr, 0 ) );` ?
    b) `System.out.println( mystery3( arr, 1 ) );` ?

```
public int mystery3( int[] array, int pos ) {
    if( pos >= array.length ) {
        return 0;
    }
    else {
        return array[ pos ] + mystery3( array, pos + 2 );
    }
}
```

11. I anropen nedan är *arr* samma array som i Uppgift 9. Vad blir utskriften vid anropet
    `System.out.println( mystery4( arr, 0 ) );` ?

```
public int mystery4( int[] array, int pos ) {
    if( pos >= array.length )
        return 0;
    else if( array[ pos ] < 0 )
        return 1 + mystery4( array, pos + 1 );
    else
        return mystery4( array, pos + 1);
}
```

12. Skriv den rekursiva metoden *sum*, vilken ska summera heltalen i intervallet min – max (se metodhuvud nedan). Exempel:

```
System.out.println( sum( 4, 8 ) );         // Resultat: 30
System.out.println( sum( 5, 2 ) );         // Resultat: 0
System.out.println( sum( -2, 1 ) );        // Resultat: -2
```

    Komplettera metoden med kod

```
public int sum( int min, int max ) {
    // komplettera här
}
```

13. Skriv den rekursiva metoden *reverse* vilken ska returnera en sträng i omvänd ordning. Metoderna *charAt* och *substring* i klassen String är användbara i lösningen.

```
System.out.println( reverse( "Student" ) );       // Resultat: tnedutS
```

    Komplettera metoden med kod

```
public String reverse( String str ) {
    // komplettera här
}
```

14. Vad blir utskriften vid anropet `System.out.println( mystery5( 4, 1 ) );` ?

```
public int mystery5( int n , int res ) {
    if( n == 1 )
        return res;
    else
        return mystery5( n – 1, n * res );
}
```

Fördjupande

15. Skriv den rekursiva metoden **everySecond**, vilken ska skriva tecknen i en sträng från startposition till slutposition.
    Utskrift ska ske om:
    * startposition är större eller lika med 0
    * slutposition är mindre än antalet tecken i strängen
    * startpositionen är mindre eller lika med slutposition

    Exempel:
```
everySecond( "Student", 0, 6 );                 // Resultat: Student
everySecond( "Student", 3, 5 );                 // Resultat: den
everySecond( "Malmö högskola!", -4, 6 );        // Resultat:
everySecond( "Hubert", 2, 10 );                 // Resultat:
everySecond( "Hubert", 5, 2 );                  // Resultat:
```

    Komplettera metoden med kod
```
public void everySecond(String txt, int startPos, int endPos) {
    // komplettera här
}
```

16. Skriv den rekursiva metoden **printString**, vilken ska skriva tecknen i en sträng från angiven position till första positionen / slutpositionen (beror på n), men endast vart n:te tecken, t.ex.

```
printString( "Student", 0 , 2);                 // Resultat: Suet
printString( "Student", 3 , -1);                // Resultat: dutS
printString( "du", 0 , 2);                      // Resultat: d
printString( "Malmö högskola! ", -2 , 1);       // Resultat:
printString( "Hubert", 10 , 0);                 // Resultat:
```

    Komplettera metoden med kod
```
public void printString(String txt, int pos, int n) {
    // komplettera här
}
```

17. Vad blir utskriften vid anropet `mystery6(10);` ?

```
public void mystery6(int a) {
    if( a >= 0) {
        System.out.println("a=" + a);
        mystery6(a-4);
        mystery6(a-3);
    }
}
```

18. Vad blir utskriften vid anropet `mystery7( arr );` om arr = { 3, 7, -2, 6, 9 }?

```
public void mystery7( int[] arr ) {
    mystery7( arr, 0 );
}

private void mystery7( int[] arr, int pos ) {
    if( ( pos >= 0) && ( pos < arr.length ) ) {
        mystery7( arr, pos + 1 );
        System.out.print( arr[ pos ] + " " );
    }
}
```

19. Skriv den rekursiva metoden *digits*, vilken ska returnera antalet siffertecken i en sträng. Du kan kontrollera om ett tecken är en siffra med:
(chr >= ’0’) && (chr <= ’9’)

```
System.out.println( digits( "Student" ) );            // Resultat: 0
System.out.println( digits( "RDS 435" ) );            // Resultat: 3
System.out.println( digits( "Pw TT54W41" ) );         // Resultat: 4
```

Komplettera metoden med kod. Metoderna charAt och substring är användbara i lösningen.

```
public int digits( String str ) {
    // komplettera här
}
```

20. Skriv den rekursiva metoden *digits* vilken ska returnera antalet siffror i ett heltal av typen int. Heltalet ska vara större än 0.

Returtyp: int
Parametrar: (int nbr)
Om nbr == 0
  returnera 0
Annars
  returnera 1 + Rekursivt anrop med argumentet (nbr / 10)

```
System.out.println( digits( 95004 ) );      // Resultat: 5
System.out.println( digits( 32 ) );         // Resultat: 2
```

21. Fibonacci är talföljden 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... Varje tal är summan av de två föregående. Detta gäller inte de två första talen i serien.
fib(1) = 1
fib(2) = 1
fib(3) = fib(2) + fib(1);

Skriv metoden *fib* vilken ska beräkna det n:te fibonacci-talet.
Exempel:

```
System.out.println( fib( 1 ) );             // Resultat: 1
System.out.println( fib( 4 ) );             // Resultat: 3
System.out.println( fib( 8 ) );             // Resultat: 21
```

Komplettera metoden med kod

```
public long fib( int n ) {
    // komplettera med kod
}
```

22. Testa hur lång tid det tar att göra 1000 anrop av *fib*-metoden med argumentet 30.

```
 long starttid = System.currentTimeMillis();
// 1000 anrop till fib( 30 );
long stopptid = System.currentTimeMillis();
System.out.println( stopptid – starttid );
```

23. Det går att skriva en effektivare fibonacci-metod genom att använda fler parametrar. De två föregående talen i serien lagras i extraparametrarna n1 och n2.
    Du ska skriva metoden

```
    private long fib2(int n, int n1, int n2)
```

    vilken anropas av metoden fib2(int n):

```
public long fib2( int n ) {
    return fib2( n, 1, 1)
}
```

Algoritm för fib2

Om n<=2
  returnera n2
Annars
  returnera rekursivt anrop med argumenten (n – 1, n2, n1 + n2)

Testa denna version av fibonacci så att korrekt resultat returneras. Testa sedan hur lång tid det tar att göra 1000 anrop till metoden (du kan även testa 10000,...)

Extra

Interfacet *IntModifier* är givet:

```
public interface IntModifier {
    public int modifyInt( int number );
}
```

24. Skriv den rekursiva metoden **changeIntArray(int[] array, IntModifier mod)** vilken ska modifiera talen i array genom anrop till *mod.modifyInt* för varje element.

    Exempel:

```
public class Adder implements IntModifier {
    private int nbr;

    public Adder( int nbr ) {
        this.nbr= nbr;
    }

    public int modifyInt( int number ) {
        return ( number + nbr );
    }
}
```

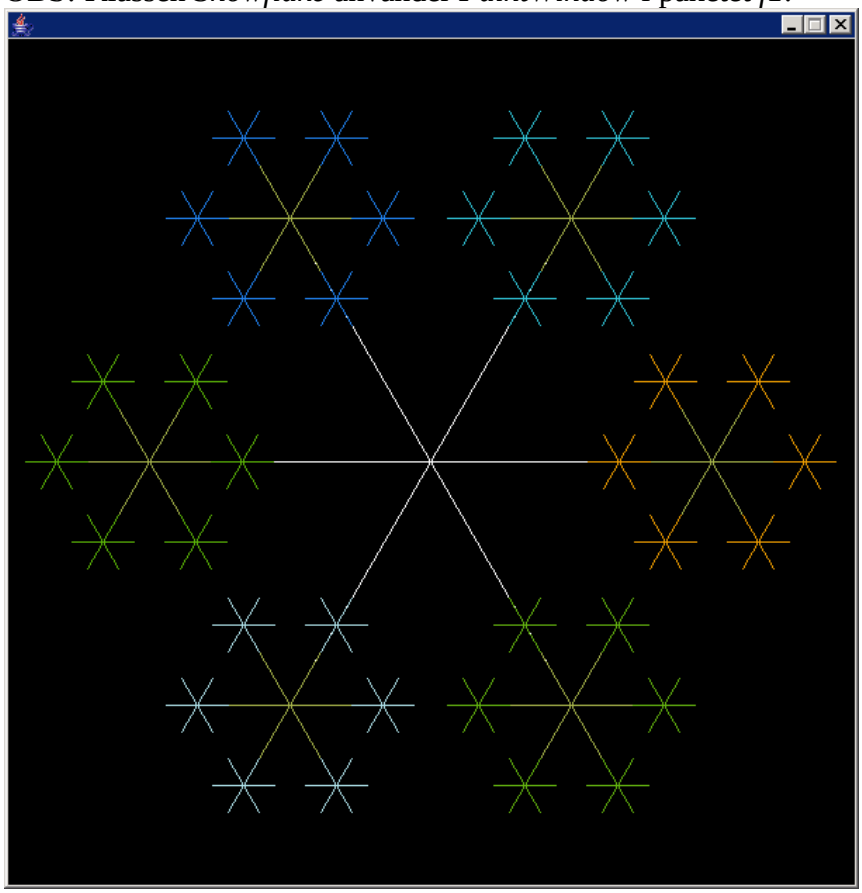
    Testkod:

```
Adder adder = new Adder( 9 );
int[] arr = { 17,-4, 6 };
changeIntArray( arr,  adder );
for( int i : arr ) {
    System.out.print ( arr + " " ); // Utskrifterna 26 5 15
}
```

25. Skriv klassen *Even* vilken implementerar *IntModifier* på följande sätt:

    Ett udda heltal görs jämnt genom att 1 adderas till dess värde. Heltal som redan är delbara med 2 ändras inte.

26. Komplettera metoden *newSnowflake*, i klassen *Snowflake*, med kod så att du får en figur liknande nedanstående när du kör programmet (färgerna är slumpmässiga).
    OBS! Klassen *Snowflake* använder *PaintWindow* i paketet *f1*.



27. Hitta på någon egen självliknande figur. Om du vill kan du utgå från Rectangles (föreläsning) eller Snowflake.

Lösningar

Uppgift 1

```
    public void print(int min, int max) {
        if(min<=max) {
            System.out.print(min+" ");
            print(min+1,max);
        }
    }
```

Uppgift 2

```
    public void everySecondReverse(String txt, int pos) {
        if(pos>=0 && pos<txt.length()) {
            System.out.print(txt.charAt(pos));
            everySecondReverse(txt, pos-2);
        }
    }
```

Uppgift 2

```
    public void everySecondReverse(String txt, int pos) {
        if(pos>=0 && pos<txt.length()) {
            System.out.print(txt.charAt(pos));
            everySecondReverse(txt, pos-2);
        }
    }
```

Uppgift 3

```
10
8
6
4
2
```

Uppgift 4

```
10 + 20 = 30
11 + 19 = 30
12 + 18 = 30
13 + 17 = 30
14 + 16 = 30
15 + 15 = 30
```

Uppgift 5

```
    public void reverse( int min, int max ) {
        if( min <= max ) {
            System.out.print(max + " ");
            reverse(min,max-1);
        }
    }
```

Uppgift 6

```
    public void print( String str, int pos ) {
        if(pos>=0 && pos<str.length()) {
            System.out.print(str.charAt(pos));
            print(str,pos+1);
        }
    }
```

Uppgift 8

```
    public long factorial( long n ) {
        if(n<=1)
            return 1;
```

```
        else
            return n * factorial(n-1);
    }
```

Uppgift 9

```
    public boolean member( int nbr, int[] array, int pos ) {
        if( pos<0 || pos>=array.length)
            return false;
        else if( nbr == array[pos] )
            return true;
        else
            return member( nbr, array, pos+1 );
    }
```

Uppgift 10

a) 11
b) -35

Uppgift 11

2

Uppgift 12

```
    public int sum( int min, int max ) {
        if( min>max )
            return 0;
        else
            return min + sum( min+1, max );
    }
```

Uppgift 13

```
    public String reverse( String str ) {
        if(str.length()==0)
            return "";
        else
            return reverse(str.substring(1)) + str.charAt(0);
    }
```

Uppgift 14

24

Uppgift 15

```
    public void everySecond( String str, int startPos, int endPos ) {
        if(startPos>=0 && startPos<=endPos && endPos<str.length()) {
            System.out.print(str.charAt(startPos));
            everySecond(str,startPos+1,endPos);
        }
    }
```

Uppgift 16

```
    public void printString( String str, int pos, int n ) {
        if(pos>=0 && pos<str.length()) {
            System.out.print(str.charAt(pos));
            printString(str,pos+n,n);
        }
    }
```

Uppgift 17

a=10
a=6
a=2
a=3
a=0

```
a=7
a=3
a=0
a=4
a=0
a=1
```

Uppgift 18

```
9 6 -2 7 3
```

Uppgift 19

```
    public int digits( String str ) {
        if(str.length()==0)
            return 0;
        else if(str.charAt(0)>='0' && str.charAt(0)<='9')
            return 1 + digits(str.substring(1));
        else
            return digits(str.substring(1));
    }
```

Uppgift 20

```
    public int digits( int nbr ) {
        if( nbr==0 )
            return 0;
        else
            return 1 + digits(nbr/10);
    }
```

Uppgift 21

```
    public long fib( int n ) {
        if( n == 1 || n == 2 )
            return 1;
        else
            return fib(n-1) + fib(n-2);
    }
```

Uppgift 23

```
    public long fib2( int n ) {
        return fib2(n,1,1);
    }

    private long fib2( int n, int n1, int n2) {
        if(n<=2)
            return n2;
        else
            return fib2( n-1, n2, n1+n2 );
    }
```

Uppgift 24

```
    public void changeIntArray( int[] array, IntModifier mod ) {
        changeIntArray(array,mod,0);
    }
    private void changeIntArray( int[] array, IntModifier mod, int pos ) {
        if(pos>=0 && pos<array.length) {
            array[pos] = mod.modifyInt(array[pos]);
            changeIntArray(array,mod,pos+1);
        }
    }
```

Uppgift 25

```
public class Even implements IntModifier {
    public int modifyInt( int number ) {
```

```
        if(number%2==-1 || number%2==1 )
            number++;
        return number;
    }
}
```

Uppgift 26

```
    public void newSnowflake( PaintWindow frame, int x, int y, int len,
Color color ) {
        Color randomColor;
        Point point;
        if( len >= 10) {
            snowflake( frame, x, y, len , color );
            randomColor = randomColor();
            for( int pos=0; pos<=5; pos++ ) {
                point = getTip(x,y,len,pos);
                newSnowflake(frame,point.x,point.y,len/3,randomColor);
            }
        }
    }
```