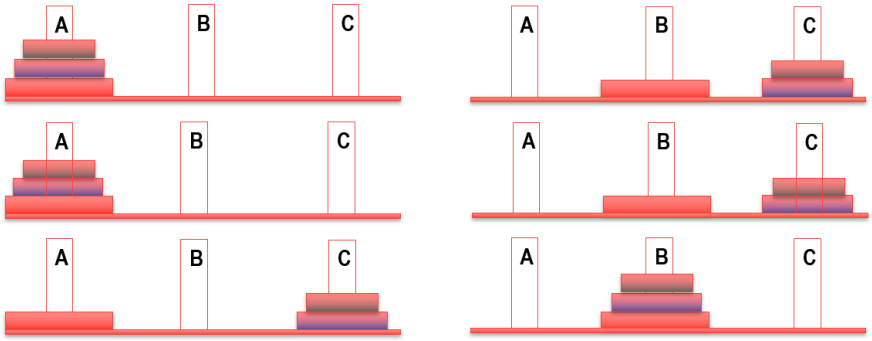
REKURSIJA UN REKURENCE
27.nodarbība

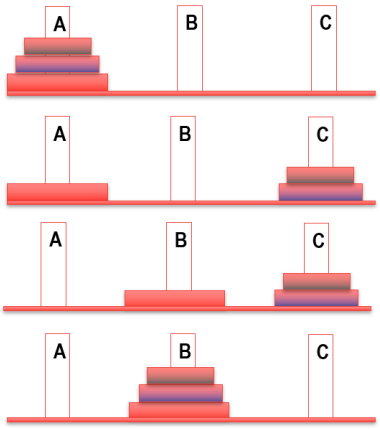
REKURSIJA – DEFINĪCIJA

- Algoritma veidošanas paņēmiens, kas paredz, ka process atkārto pats sevi.
- Rekursīva programma izsauc pati sevi:

```
public int uzdevums(int n){
      if(n>9)return 1+uzdevums(n/10);
      else return 0;
}
```

HANOJAS TORŅI





- Algoritms - kā pārlikt n ripas no A uz B, izmantojot C. Hanoja(n,A,B,C)
- Izpilda Hanoja(n-1,A,C,B)
- Pārliek ripu no A uz B
- Izpilda Hanoja(n-1,C,B,A)

FAKTORIĀLS n!

Naturāla skaitļa $n \geq 1$ faktoriāls = visu naturālo skaitļu no 1 līdz n reizinājums:

$1!=1$

$2!=1 \cdot 2=2$

$3!=1 \cdot 2 \cdot 3=6$

$4!=1 \cdot 2 \cdot 3 \cdot 4=24$

$5!=1 \cdot 2 \cdot 3 \cdot 4 \cdot 5=120$

FAKTORIĀLS – REKURENTĀ FORMULA

$n!=n(n-1)!$

$4!=4 \cdot 3!$

$=4 \cdot 3 \cdot 2!$

$= 4 \cdot 3 \cdot 2 \cdot 1!$

$= 4 \cdot 3 \cdot 2 \cdot 1$

$=24$

FAKTORIĀLS – JAVA KODS

```
class faktorial{
public static int fakt(int n){
     if (n>1) return n*fakt(n-1);
     else return 1;
}
public static void main(String args[]) {
System.out.println(fakt(4));
 }
}
```

FAKTORIĀLS – DARBĪBĀ

```
1. class faktorial{
2. public static int fakt(int n){ if (n>1)
3.       return n*fakt(n-1);
4.       else return 1;
5.  }
6.  public static void main(String args[]) {
7.       System.out.println(fakt(4));
8.  }}
```

Koda rinda	N	Iesāktie, bet nepabeigtie darbi
6	-	-
7	-	System.out.println(fakt(4))
2	4	System.out.println(fakt(4))
3	4	System.out.println(fakt(4)), 4*fakt(3)
2	3	System.out.println(fakt(4)), 4*fakt(3)
3	3	System.out.println(fakt(4)), 4*fakt(3), 3*fakt(2)
2	2	System.out.println(fakt(4)), 4*fakt(3), 3*fakt(2)
3	2	System.out.println(fakt(4)), 4*fakt(3), 3*fakt(2), 2*fakt(1)
2	1	System.out.println(fakt(4)), 4*fakt(3), 3*fakt(2), 2*fakt(1)
4	1	System.out.println(fakt(4)), 4*fakt(3), 3*fakt(2), 2*1
3	2	System.out.println(fakt(4)), 4*fakt(3), 3*2
3	3	System.out.println(fakt(4)), 4*6
3	4	System.out.println(24)
7	-	-

22

FIBONAČI SKAITĻI

- Fibonači skaitļi – rekurenta vienādojuma atrisinājums

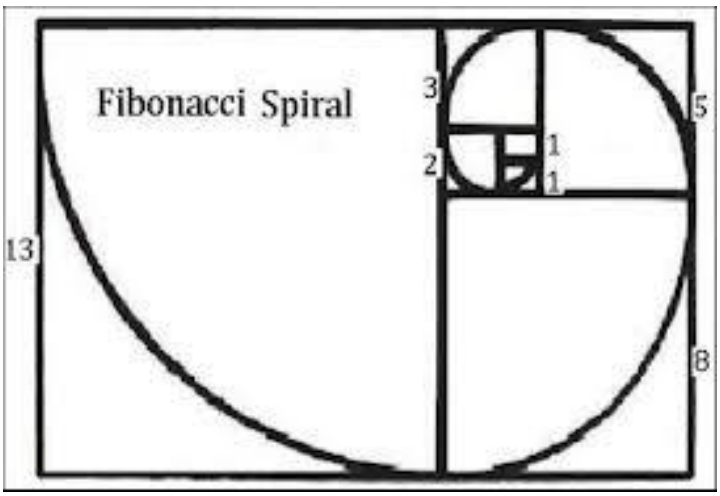
$$f_n = \begin{cases} 1, & \text{ja } n = 1 \text{ vai } n = 2; \\ f_{n-1} + f_{n-2}, & \text{ja } n > 2. \end{cases}$$

- To var pierakstīt arī šādi:

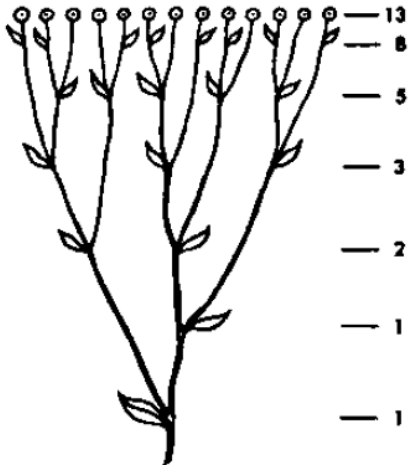
$$f_1 = f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2}$$

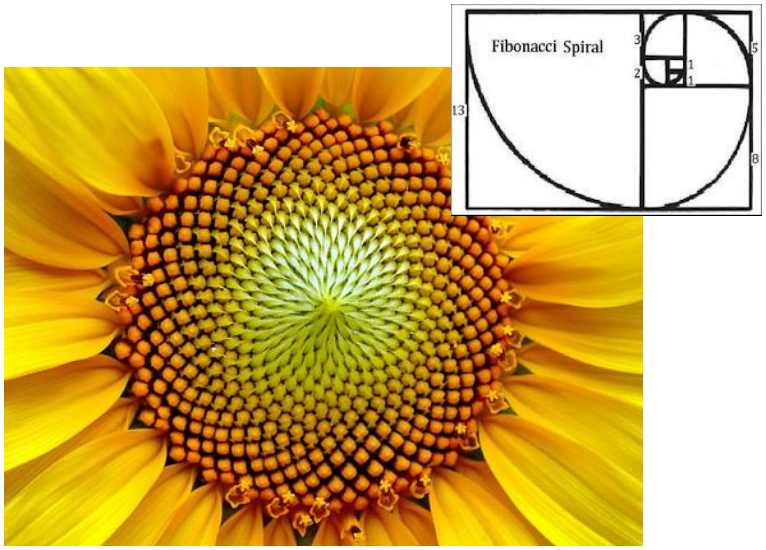
- F_1=1
- F_2=1
- F_3=2
- F_4=3
- F_5=5
- F_6=8
- F_7=13



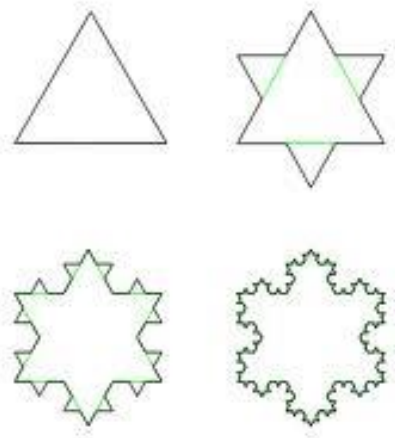
- F1=1
- F2=1
- F3=2
- F4=3
- F5=5
- F6=8
- F7=13

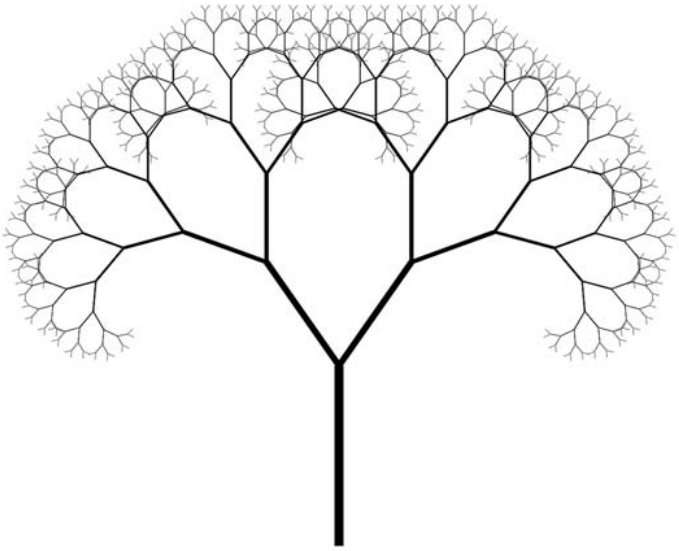


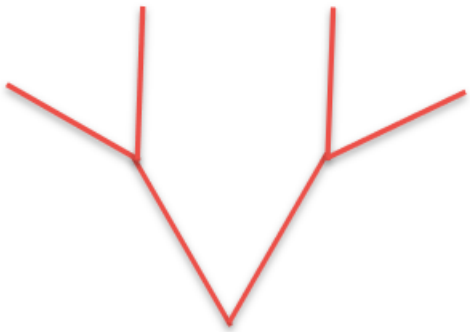
- F1=1
- F2=1
- F3=2
- F4=3
- F5=5
- F6=8
- F7=13



DABAS FORMAS REKURSĪVI – FRAKTĀĻI



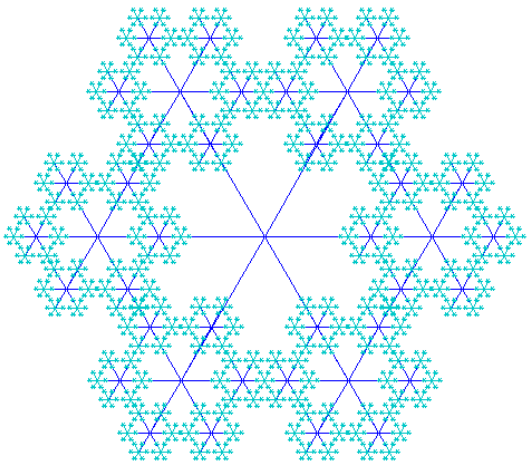




JAUTĀJUMS

Kāds attēla fragments atkārtoti jāzīmē,

lai iegūtu šādu sniegpārslas attēlu?



JAVA ZĪMĒŠANAS IESPĒJAS ANDROID VIDĒ

- Veidosim jaunu Android projektu ar apakšklasi *Drawing*.
- *Drawing* manto visas *View* īpašības, tāpēc tai eksistē metode *onDraw*, kas dabū līdzi objektu *Canvas*, uz kā var zīmēt.
- Lai galvenā aktivitāte izmantotu «apzīmēto» kanvu, jānomaina galvenās aktivitātes skats.

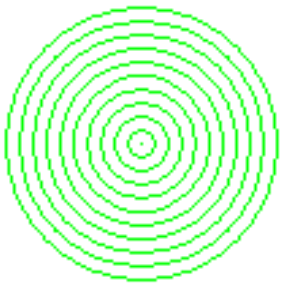
```
public class MainActivity extends Activity {

    class Drawing extends View {
        Canvas canva;
        Paint brush;
        public Drawing(Context context) {
            super(context);
        }
        public void onDraw(Canvas canva){
            this.canva=canva;
            brush= new Paint();
            brush.setStyle(Style.STROKE);
            brush.setStrokeWidth(5);
            brush.setColor(Color.rgb(123, 200, 0));
            canva.drawCircle(150, 150, r, brush);
            canva.drawLine(1,1, 123,200, brush);
            canva.drawRect(1,1, 123,200, brush);
        }
    }
```

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

KĀ UZZĪMĒT 10 APĻUS?

- Papildināsim klasi *Drawing* ar metodi *rekRinki*().
- Apļa zīmēšanas komandu pārcelsim uz šo metodi.
- Kad aplis uzzīmēts, liksim, lai metode *rekRinki* atkārto pati sevi, bet ar mazāku rādiusu.
- Lai rekursija beigtos, ja rādiuss pārāk mazs, pieliekam pārbaudi.
- No metodes *onDraw*() izsaucam metodi *rekRinki*().

- Ja gribam, lai uzzīmē 10 apļus, līdzi dodam sākuma rādiusa vērtību 250.



```
public void rekRinki(int r){
  if (r>0){
      canva.drawCircle(150, 150, r, brush);
      rekRinki(r-25);
  }
}
```

```
public void onDraw(Canvas canva){
    this.canva=canva;
    brush= new Paint(Paint.ANTI_ALIAS_FLAG);
    brush.setStyle(Style.STROKE);
    brush.setStrokeWidth(5);
    brush.setColor(Color.rgb(123, 200, 0));
    rekRinki(250);
}
```

KĀ UZZĪMĒT APĻU KOMPOZĪCIJU?

- Programmai *rekRinki*() vajadzētu sevi atkārtot 4 reizes dažādās vietās.
- Lai *rekRinki*() zinātu to vietu, veidosim otru metodi *rekRinki*(), kam līdzi nāks arī koordinātas xc un *yc*.
- Zīmējam apli ar rādiusu *r* un centru *xc*,*yc*.
- Atkārtojam metodi *rekRinki*() četrās vietās.
- Parūpējamies, lai rekursija savlaicīgi beidzas.
- No metodes *onDraw*() izsaucam metodi *rekRinki*().

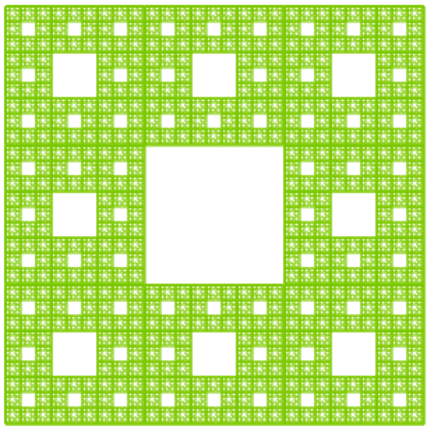
```
public void rekRinki(int xc,int yc,int r){
    if (r>0){
            canva.drawCircle(xc,yc, r, brush);
            rekRinki(xc+r,yc,r/3);
            rekRinki(xc-r,yc,r/3);
            rekRinki(xc,yc-r,r/3);
            rekRinki(xc,yc+r,r/3);
    }
}
```

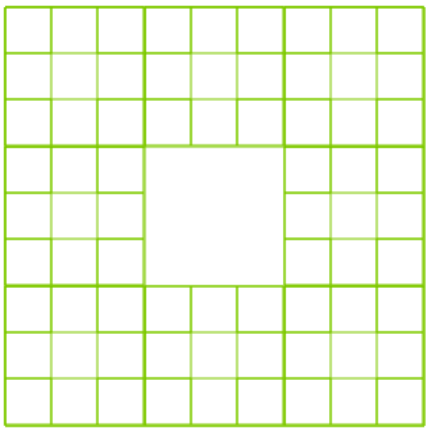
```
public void onDraw(Canvas canva){
    this.canva=canva;
    brush= new Paint(Paint.ANTI_ALIAS_FLAG);
    brush.setStyle(Style.STROKE);
    brush.setStrokeWidth(5);
    brush.setColor(Color.rgb(123, 200, 0));
    rekRinki(200);
    rekRinki(200,500,100);
}
```

Polimorfisms!!

SERPINSKA PAKLĀJS

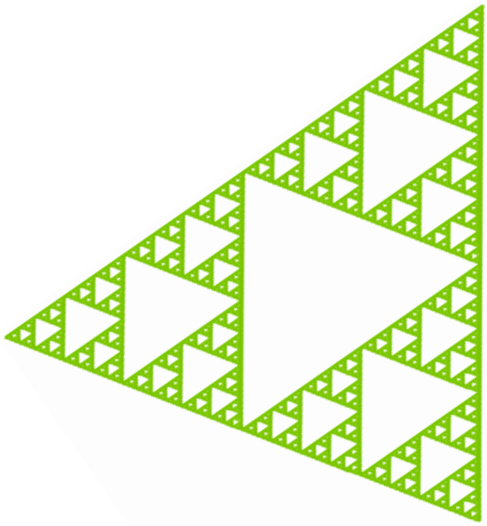
- Vajadzīga metode, kas prot uzzīmēt kvadrātu un atkārtot sevi 8 reizes.

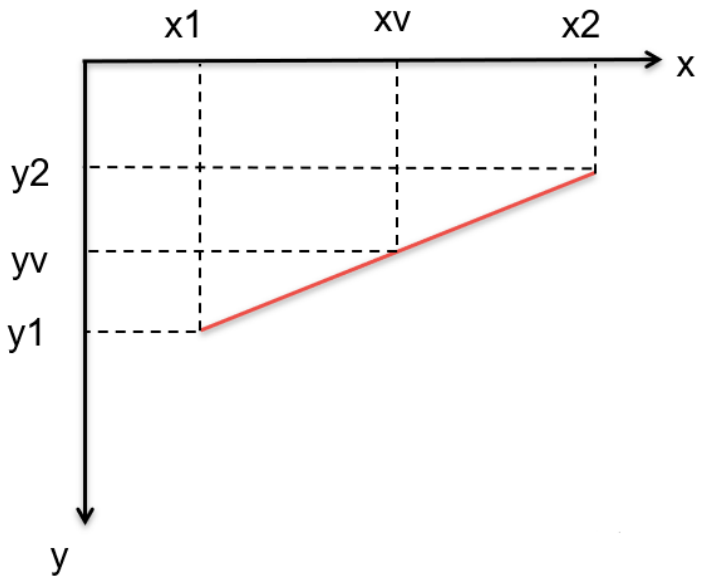




SERPINSKA TRIJSTŪRIS

- Vajadzīga metode, kas prot uzzīmēt 3 līnijas un atkārtot sevi 3 reizes.
- Jauno trijstūru virsotnes – malu viduspunktos.
- xv=x1+(x2-x1)/2
- yv=y2+(y1-y2)/2





SERPINSKA TRIJSTŪRIA KODS

```
public int vidus(int a,int b){
    if(a>b)return b+(a-b)/2; else return a+(b-a)/2;
}
```

```
public void trijsturis(int x1,int y1,int x2,int y2,int x3,int y3){
   if(Math.abs(x1-x2)>0&&Math.abs(x1-x3)>0&&Math.abs(x3-x2)>0){
        canva.drawLine(x1, y1, x2, y2, brush);
        canva.drawLine(x1, y1, x3, y3, brush);
        canva.drawLine(x3, y3, x2, y2, brush);
        trijsturis(x1,y1,vidus(x1,x2),vidus(y1,y2),vidus(x1,x3),vidus(y1,y3));
        trijsturis(x2,y2,vidus(x1,x2),vidus(y1,y2),vidus(x2,x3),vidus(y2,y3));
        trijsturis(x3,y3,vidus(x1,x3),vidus(y1,y3),vidus(x2,x3),vidus(y2,y3));
   }
}
```