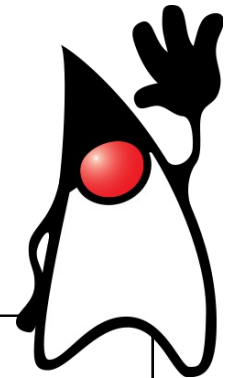


Project 1

Do's and don'ts



Klas	1EO-ICT
Opleiding	Bachelor in de elektronica-ICT
Vak	Java basisconcepten
Lokaal	D0.49 of E0.31
Tijdstip	Dinsdag, LT 2 of 3
Docent	Katja Verbeeck
Contact	Katja.verbeeck@odisee.be



```

/**
 * Schrijf een klasse cirkel, lees straal, bereken omtrek en opp
 *
 * @author Docenten ICT
 * @version okt 2014
 */

public class CirkelNOK {

    double straal; // dataveld van de klasse Cirkel

    public void leesStraal() {
        // lees straal in
        System.out.println("Geef een straal in : ");
        straal = Input.readDouble();

        if (straal >= 0) {
            bepaalOmtrek();
        }
    }

    public void bepaalOmtrek() {
        // bereken omtrek, rond af en druk resultaat op scherm

        double omtrek = 2 * Math.PI * straal;
        System.out.println("De omtrek van de cirkel bedraagt : "
            + Math.round(omtrek * 100) / 100.0);

        bepaalOppervlakte();
    }
}

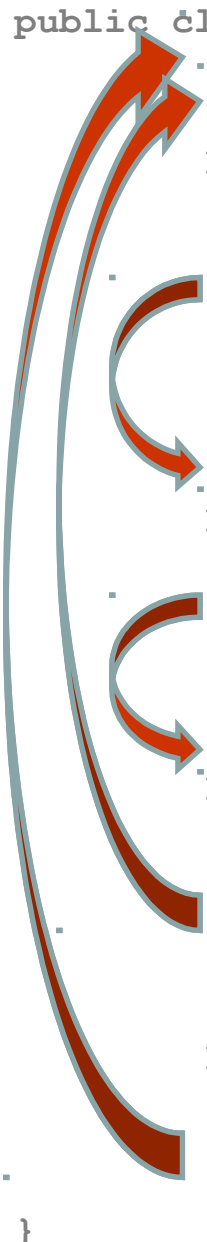
```

```
public void bepaalOppervlakte() {  
    // bereken oppervlakte, rond af en druk resultaat op scherm  
    double oppervlakte = Math.PI * straal * straal ;  
    System.out.println("De omtrek van de cirkel bedraagt : " +  
                        Math.round(oppervlakte * 100) / 100.0);  
    leesStraal();  
}  
  
public static void main(String[] args) {  
    CirkelNOK c = new CirkelNOK();  
    c.leesStraal();  
}
```

```
}
```

OEFENING: Schrijf een klasse cirkel, lees straal, bereken omtrek en opp

```
public class CirkelNok {  
    -double straal;  
  
    public void leesStraal() {  
        // lees straal in  
  
        if (straal >= 0) {  
            bepaalOmtrek();  
        }  
    }  
  
    public void bepaalOmtrek() {  
        // bereken omtrek, rond af en druk resultaat op scherm  
  
        bepaalOppervlakte();  
    }  
  
    public void bepaalOppervlakte() {  
        // bereken oppervlakte, rond af en druk resultaat op scherm  
  
        leesStraal();  
    }  
  
    public static void main(String[] args) {  
        CirkelNok c = new CirkelNok();  
        c.leesStraal();  
    }  
}
```



PROBLEMEN:

- Geen ogenblikkelijk zicht op de set van kerntaken van het programma
- Methodenamen dekken hun 'lading' niet

OPLOSSING:

Gebruik een start methode met hierin een gepaste lusstructuur gebruiken !!!

OVERBELAST DE STACK NIET !!!!

```
public class CirkelNokCircular {
    double straal = 0;

    public void start() {
        bepaalOmtrek();
        bepaalOppervlakte();
        straal++;
        start();
    }

    public void bepaalOmtrek() {
        // bereken omtrek, rond af en druk resultaat op scherm
    }

    public void bepaalOppervlakte() {
        // bereken oppervlakte, rond af en druk resultaat op scherm
    }

    public static void main(String[] args) {
        CirkelNokCircular c = new CirkelNokCircular();
        c.start();
    }
}
```

DEBUG DEMO

De omtrek bedraagt 55618.75 cm

De oppervlakte bedraagt 2.147483647E7 cm²

De omtrek bedraagt 55625.03 cm

De oppervlakte bedraagt 2.147483647E7 cm²

Exception in thread "main" java.lang.StackOverflowError

at sun.nio.cs.SingleByteEncoder.encodeArrayLoop(SingleByteEncoder.java:91)

at sun.nio.cs.SingleByteEncoder.encodeLoop(SingleByteEncoder.java:130)

at java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:544)

at sun.nio.cs.StreamEncoder.implWrite(StreamEncoder.java:252)

at sun.nio.cs.StreamEncoder.write(StreamEncoder.java:106)

at java.io.OutputStreamWriter.write(OutputStreamWriter.java:190)

at java.io.BufferedWriter.flushBuffer(BufferedWriter.java:111)

at java.io.PrintStream.write(PrintStream.java:476)

at java.io.PrintStream.print(PrintStream.java:619)

at java.io.PrintStream.println(PrintStream.java:756)

at CirkelNokCircular.bepaalOmtrek(CirkelNokCircular.java:15)

at CirkelNokCircular.start(CirkelNokCircular.java:5)

at CirkelNokCircular.start(CirkelNokCircular.java:8)

...

at CirkelNokCircular.start(CirkelNokCircular.java:8)

at CirkelNokCircular.start(CirkelNokCircular.java:8)

Druk op een toets om door te gaan. . .

```
public class CirkelOk {  
    double straal;
```

Overzicht van alle kerntaken in één oogopslag !

```
public void start(){  
    leesStraal();  
  
    while (straal >= 0) {  
        double omtrek = bepaalOmtrek();  
        System.out.println("De omtrek van de cirkel bedraagt : " +  
                           Math.round(omtrek * 100) / 100.0);  
        double opp = bepaalOppervlakte();  
        System.out.println("De omtrek van de cirkel bedraagt : " +  
                           Math.round(opp * 100) / 100.0);  
        leesStraal();  
    }  
}
```

```
public void leesStraal() {  
    // lees straal in  
    System.out.println("Geef een straal in : ");  
    straal = Input.readDouble();  
}
```

```
public double bepaalOmtrek() {  
    // bereken omtrek  
    return 2 * Math.PI * straal;  
}
```

```
public double bepaalOppervlakte() {  
    // bereken oppervlakte  
    return Math.PI * straal * straal ;  
}
```

```
public static void main(String[] args) {  
    CirkelOk c = new CirkelOk();  
    c.start();  
}
```

Samengevat

1) Opsplitsen in methoden :

Splits 'zinvol' uit als het aantal instructies voor een functionaliteit te groot wordt // elke methode heeft slechts 1 functionaliteit te vervullen

```
public class Acties {  
    public void start() {  
        functionaliteit1();  
        functionaliteit2();  
    }  
  
    public void functionaliteit1() {  
        //instructies 1..n  
    }  
  
    public void functionaliteit2() {  
        //instructie 1..m  
    }  
  
    public static void main(String[] args) {  
        Acties a = new Acties();  
        a.start();  
    }  
}
```



```
public void functionaliteit1() {  
    functionaliteit1_deelA();  
    functionaliteit1_deelB();  
}  
  
public void functionaliteit1_deelA() {  
    //instructies 1..(n/2)  
}  
  
public void functionaliteit1_deelB() {  
    //instructies (n/2+1)..n  
}
```

Splits niet artificieel...

```
public void functionaliteit2() {  
    functionaliteit2_deelA();  
    functionaliteit2_deelB();  
}  
  
public void functionaliteit2_deelA() {  
    //instructies 1..(m/2)  
}  
  
public void functionaliteit2_deelB() {  
    //instructies (m/2+1)..m  
}
```

```
public void functionaliteit1() {  
    deelfunctionaliteitX();  
    deelfunctionaliteitY();  
}
```

```
public void deelfunctionaliteitX() {  
    //instructies m.b.t. deelfunctionaliteit X  
}
```

```
public void deelfunctionaliteitY() {  
    //instructies m.b.t. deelfunctionaliteit Y  
}
```

**...maar volgens
betekenisvolle
deelfunctionaliteit !**

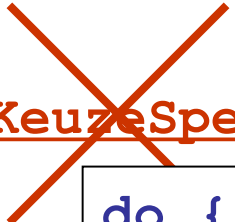
```
public void functionaliteit2() {  
    deelfunctionaliteitZ();  
    deelfunctionaliteitQ();  
}
```

```
public void deelfunctionaliteitZ() {  
    //instructies m.b.t. deelfunctionaliteit Z  
}
```

```
public void deelfunctionaliteitQ() {  
    //instructies m.b.t. deelfunctionaliteit Q  
}
```

2) Gebruik GEEN 'recursieve' oproepen, maar WEL een gepaste lus-structuur!

```
public void vraagKeuzeSpeler() {  
    System.out.println("Kies: Blad, Steen of Schaar");  
    String ingave = Input.readString().toLowerCase();  
  
    if (ingave.equals("blad")) {  
        keuzeSpeler = BLAD;  
    }  
    else if (ingave.equals("steen")) {  
        keuzeSpeler = STEEN;  
    }  
    else if (ingave.equals("schaar")) {  
        keuzeSpeler = SCHAAR;  
    }  
    else {  
        vraagKeuzeSpeler();  
    }  
}
```

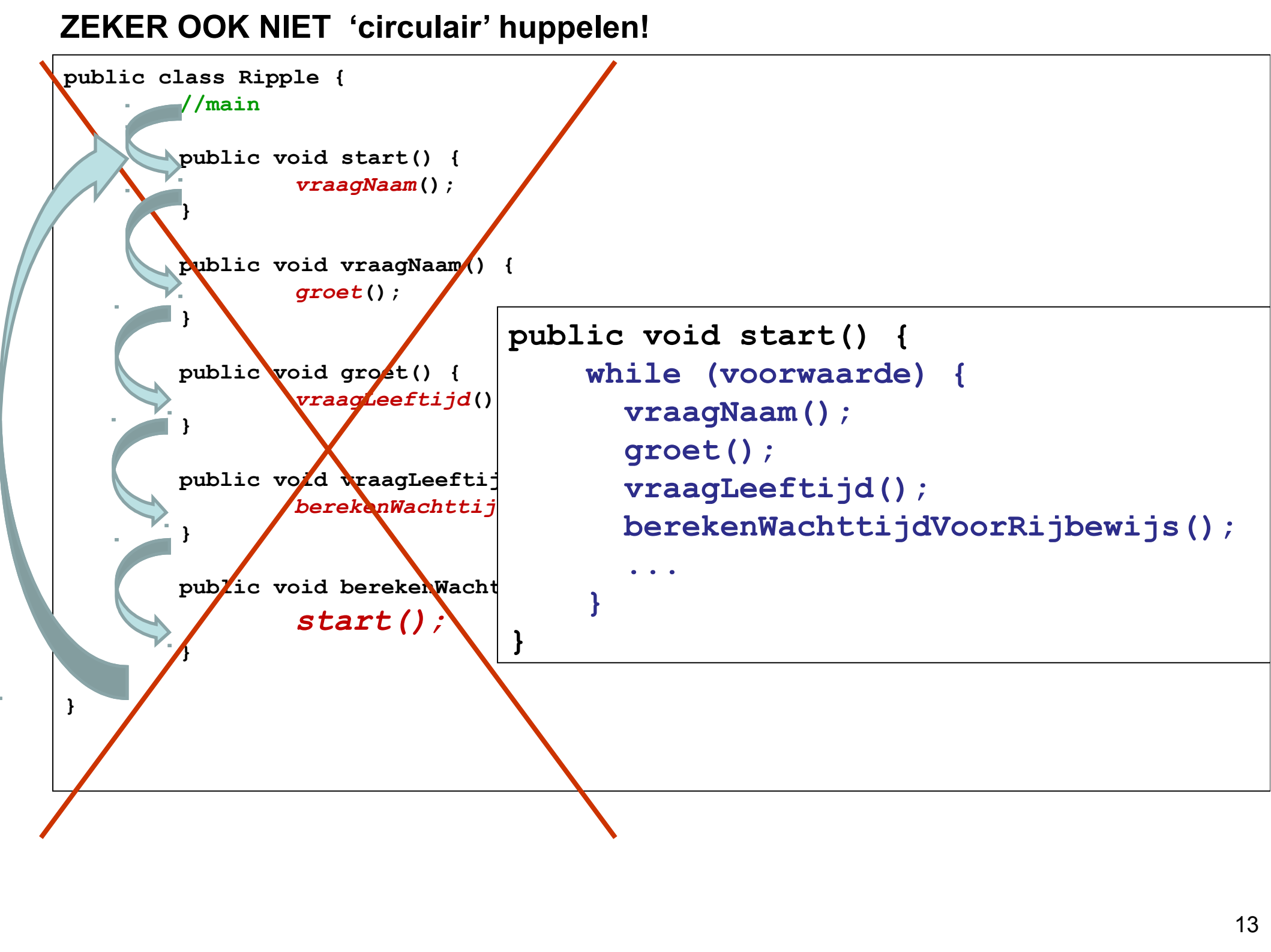


```
do {  
    System.out.println("Kies: ...");  
    ingave = ...;  
} while ( !ingave.equals("blad")  
        && !ingave.equals("steen")  
        && !ingave.equals("schaar") );
```

3) Huppel NIET van de ene methode naar de andere !

```
public class Ripple {  
    String naam;  
  
    public static void main(String[] args) {  
        Ripple dont = new Ripple();  
        dont.start();  
    }  
  
    public void start() {  
        vraagNaam();  
    }  
  
    public void vraagNaam() {  
        System.out.print("Geef je naam: ");  
        naam = Input.readLine();  
        groet();  
    }  
  
    public void groet() {  
        System.out.println("Dag " + naam + "!");  
        vraagLeeftijd();  
    }  
}
```

ZEKER OOK NIET 'circular' huppelen!



```
public class Ripple {  
    //main  
    public void start() {  
        vraagNaam();  
    }  
    public void vraagNaam() {  
        groet();  
    }  
    public void groet() {  
        vraagLeeftijd();  
    }  
    public void vraagLeeftijd() {  
        berekenWachttijd();  
    }  
    public void berekenWachttijd() {  
        start();  
    }  
}
```

```
public void start() {  
    while (voorwaarde) {  
        vraagNaam();  
        groet();  
        vraagLeeftijd();  
        berekenWachttijdVoorRijbewijs();  
        ...  
    }  
}
```