

# EDA016 Programmeringsteknik för D

## Läsvecka 3: Systemutveckling

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2015

## 2 Systemutveckling

- Att göra denna vecka
- Klasser och objekt
- Metoder och parametrar
- Synlighet
- Konstruktorer
- Oföränderlighet (immutability)
- Specifikation versus implementation
- Integrerad utvecklingsmiljö

# Att göra i Vecka 3: Förstå hur systemutveckling går till i en integrerad utvecklingsmiljö

- 1 Läs följande kapitel i kursboken:  
2, 6.3, 7.1, 7.2, 7.3, 5.1, 5.2, 5.3  
Begrepp: klass, objekt, specifikation, referensvariabel, instans, IDE, arbetsområde, brytpunkt, avlusare,
- 2 Gör övning 3: beräkningar, klasser och objekt
- 3 Träffas i samarbetsgrupper och hjälp varandra förstå
- 4 Gör Lab 2: Eclipse

# Klasser och objekt

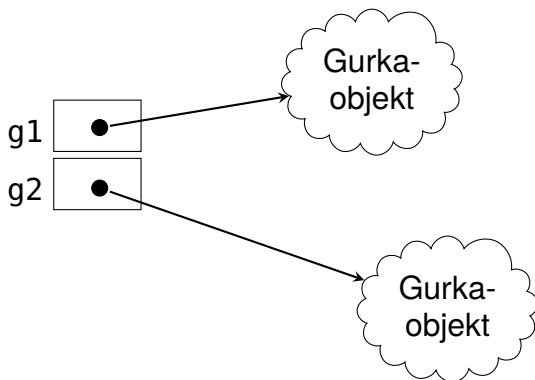
# Objekt och referensvariabler

```
1  class Gurka {
2      public int vikt = 100; //gram
3  }
4
5  public class ReferenceVariables {
6      public static void main(String[] args){
7          Gurka g1 = new Gurka();
8          Gurka g2 = new Gurka();
9          g2.vikt = 200;
10         System.out.println("Gurka 1 väger: " + g1.vikt + "g");
11         System.out.println("Gurka 2 väger: " + g2.vikt + "g");
12         g1.vikt = 200;
13         System.out.println("Gurka 1 väger nu: " + g1.vikt + "g");
14         if (g1 == g2) {
15             System.out.println("samma");
16         } else { // g1 och g2 refererar till OLIKA objekt!
17             System.out.println("olika");
18         }
19     }
20 }
```

# Objekt och referensvariabler

```
7   Gurka g1 = new Gurka();  
8   Gurka g2 = new Gurka();
```

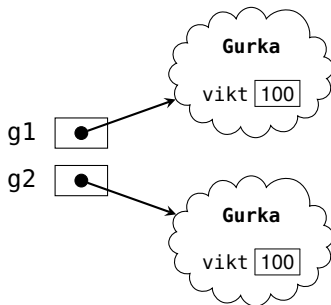
Efter rad 8 ser det ut såhär i minnet:



# Objekt och referensvariabler

```
7   Gurka g1 = new Gurka();  
8   Gurka g2 = new Gurka();
```

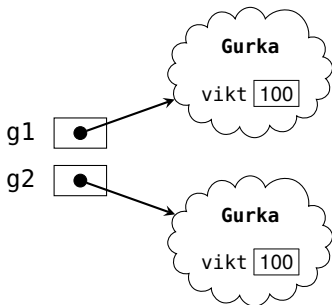
En mer detaljerad bild av minnet efter rad 8:



# Objekt och referensvariabler

```
7      Gurka g1 = new Gurka();  
8      Gurka g2 = new Gurka();
```

En mer detaljerad bild av minnet efter rad 8:



Referensvariablerna g1 och g2 pekar på *olika* objekt, sålunda är uttrycket `g1 == g2` **false**, även om objektens *innehåll* är lika och `g1.vikt == g2.vikt` är **true**.

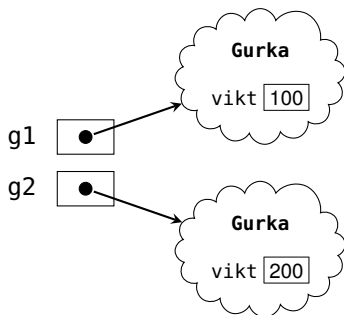


# Punktnotation för att komma åt klassmedlemmar

9

```
g2.vikt = 200;
```

Efter rad 9 ser det ut såhär i minnet:



# Metoder och parametrar

# Deklarera och anropa metoder

```
1  class Gurka {
2      public int vikt = 100; //en variabel i en klass kallas attribut (eller fält/field)
3
4      public void halvera(){ //denna metod är en procedur
5          vikt = vikt / 2;
6      }
7
8      public double kilo(){ //denna metod är en funktion utan sidoeffekt
9          return vikt / 1000.0;
10     }
11
12     public void visa(){ //denna metod är en procedur
13         System.out.println("Gurkan väger " + kilo() + "kg");
14     }
15 }
16
17 public class MethodsExamples {
18     public static void main(String[] args){
19         Gurka g = new Gurka();
20         g.visa();
21         g.vikt = 256;
22         g.visa();
23         g.halvera();
24         g.visa();
25     }
26 }
```

# Parametrar och aktiveringspost

# Lokala namn och överskuggning

# Synlighet

# Public och Private

# Konstruktörer



# This

# Oföränderlighet (immutability)

# Förhindra att variabler **ändras** med **final**

Attributet `latinsktNamn` nedan är en **konstant**.

Kompilatorn hjälper oss att kolla så att vi inte råkar ändra på det vi har deklarerat som **final**.

```
1  class Gurka {
2      public int vikt = 100; //gram
3
4      public final String latinsktNamn = "Cucumis sativus";    /*1
5
6      public String toString() {
7          return "Denna gurka (" + latinsktNamn + ") väger " + vikt + "g";
8      }
9  }
10
11  public class Constant {
12      public static void main(String[] args){
13          Gurka g = new Gurka();
14          g.vikt = 200;
15          g.latinsktNamn = "Tomat"; // ERROR: ger kompileringsfel! Vilket?
16          System.out.println(g.toString()); // *2
17      }
18  }
19
20  // *1: final deklareraras gärna även static om det bara behövs en enda
21  // *2: .toString behövs ej
```

# Specifikation versus implementation

# Krav – Design – Implementation – Test

# Integrerad utvecklingsmiljö

# Eclipse