Viktiga programmeringsbegrepp: Kontrakt

Vad lovar {klassen, metoden, fältet}?

Kontrakt

jonkv@ida

- **Kontrakt**: Överenskommelse som anger
  - Vad som ska **tillhandahållas**
  - Vad som förväntas i **utbyte**
  - Allmänna **regler** runt utbytet
  - …

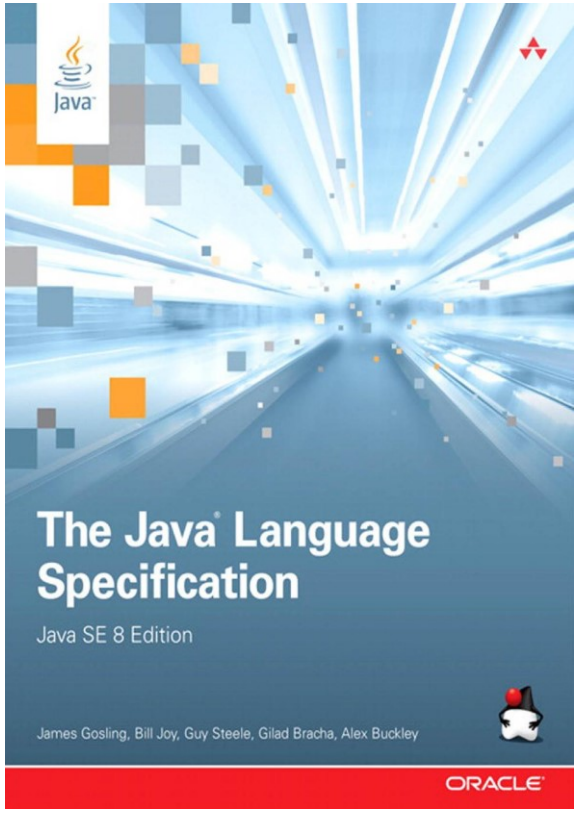
- Inom **objektorienterad programmering**:
  - Vilka värden kan en metod **ta emot**?
  - Vad **gör** metoden, om den får sådana värden? Vad gör den ***inte***?
  - Vad **returnerar** den?
  - Vad **garanterar** en **klass** angående sitt tillstånd och beteende?
  - …

Varför räcker inte koden?

jonkv@ida

```
class Circle {
  private double x, y, r;
  public Circle(double x, double y, double r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }
  public double getArea() {
     return Math.PI * this.r * this.r ;
   }
}
```

+



Definierar exakt vad programmet gör!

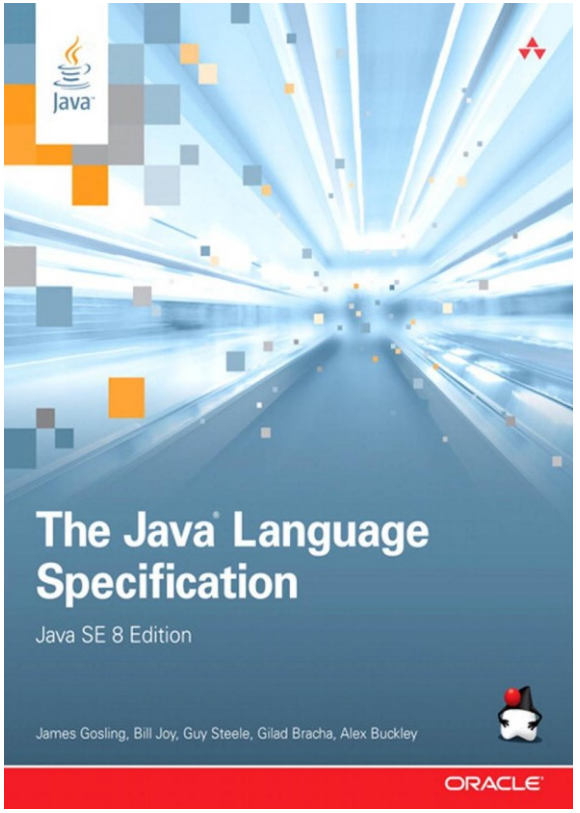
Varför räcker inte detta?

Varför räcker inte koden? (2)

jonkv@ida

```
class Circle {
  private double x, y, r;
  public Circle(double x, double y, double r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }
  public double getArea() {
     return Math.PI * this.r * this.r ;
   }
}
```

+



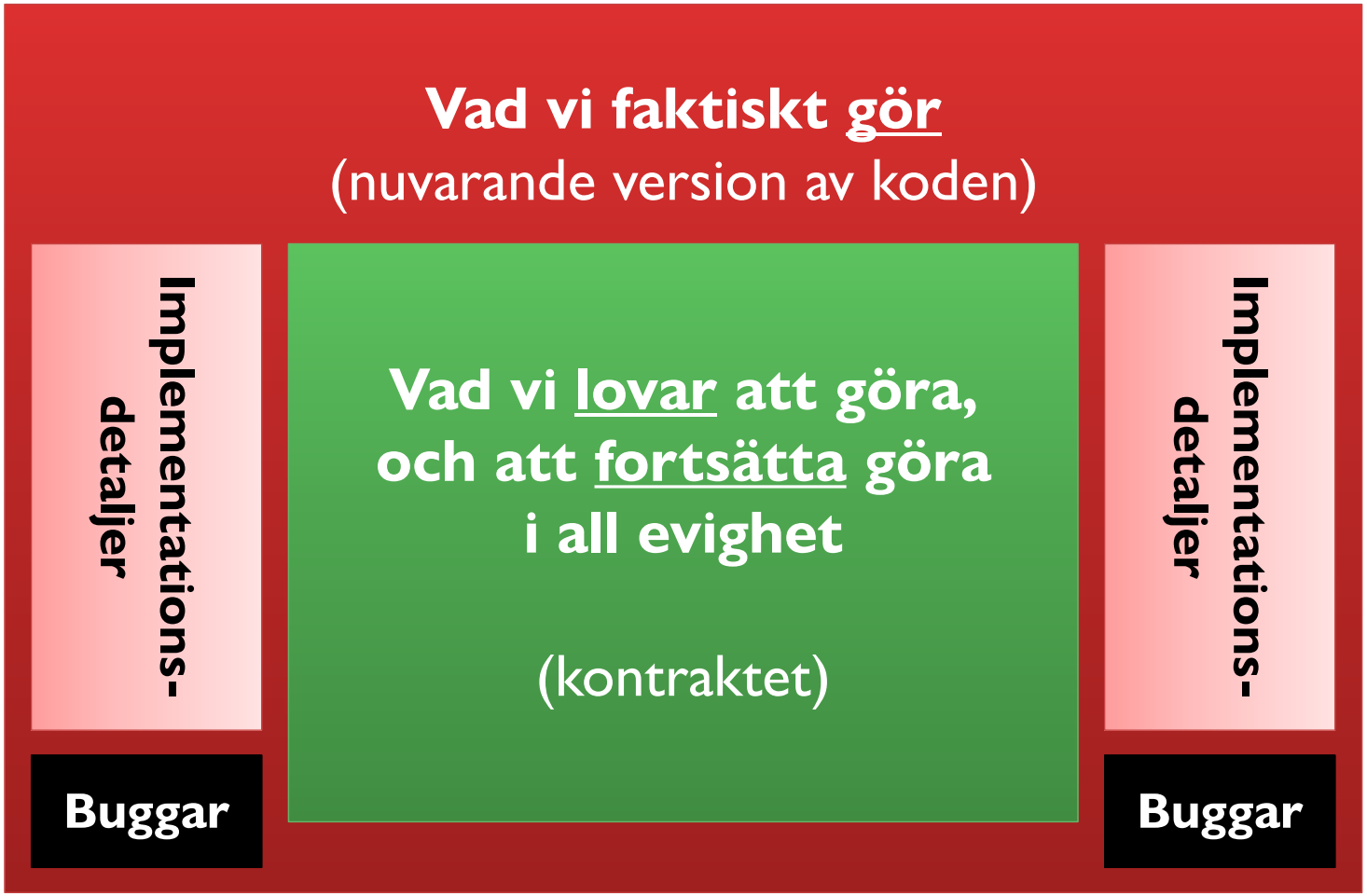
Svårt att inse alla konsekvenser av tusentals rader kod

Kontraktet ska sammanfatta vad som garanteras, krävs

Varför räcker inte koden? (3)

jonkv@ida

- Vi är inte intresserade av vad koden gör!
  - Mycket viktig **konceptuell skillnad** mellan:



Exempel: Sortering

- Exempel: Sortera strängar efter **längd**
  - Skicka in: [”abc”, ”de”, ”fghi”]
  - Resultat: [”de”, ”abc”, ”fghi”] – längd 2, 3, 4

- Vad händer om flera strängar är lika långa?
  - Skicka in: [”abc”, ”def”, **”ab”**, ”ef”, **”cd”**]
  - Kan bli: [**”cd”**, **”ab”**, ”ef”, ”abc”, ”def”] eller andra resultat

Får byta ordning på ”ab” och ”cd”
eftersom även detta blir korrekt sorterat

  - **Stabil** sortering: [”ab”, ”ef”, ”cd”, ”abc”, ”def”]

Byt inte ordning på ”ab” och ”cd”
eftersom ursprungliga ordningen var OK

Exempel: Vad utlovas?

- Får jag som **anropar sort()** anta att sorteringen är stabil?
  - Titta på sort()…

```
static void sort(String[] arr) {
    int n = arr.length;
    String temp;
    for (int i=0; i < n; i++) {
        for (int j=1; j < n - i; j++) {
            if (arr[j-1].compareTo(arr[j]) > 0) {
                //swap elements
                temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

**Aha – det är bubble sort,
som är stabil!**

**Men tänk om någon byter
till en effektivare algoritm…
Kan jag lita på att den blir stabil?**

Exempel: Vad får jag ändra?

jonkv@ida

- Får jag som **implementerar sort()** byta till QuickSort?
  - Den är **inte** stabil… Kan någon ha antagit stabilitet?

```
static void sort(String[] arr) {
    int n = arr.length;
    String temp;
    for (int i=0; i < n; i++) {
        for (int j=1; j < n - i; j++) {
            if (arr[j-1].compareTo(arr[j]) > 0) {
                //swap elements
                temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

Kontrakt 1

jonkv@ida

- Ange löften genom kontrakt!
  - Vissa språk har **explicita kontrakt** – t.ex. **Eiffel**

```
set_hour (new_hour: INTEGER)
  -- Set `hour' to `new_hour'
  require
    valid_argument: 0 <= new_hour and new_hour <= 23
  do
    hour := new_hour
  ensure
    hour_set: hour = new_hour
  end
```

precondition

postcondition

  - Ofta får man istället använda **dokumentationen**
  - Idealet: **Bara** det som står i kontraktet gäller – titta aldrig på koden!

```
class List {
  /** Sorterar listan efter ordlängd. */
  void sort() { … }
}
```

Här står inget om *stabilitet*.

Då kan vi inte förutsätta det!

Kontrakt 2

jonkv@ida

- Ju tydligare, desto bättre…

```
class List {
  …

  /** Sorterar listan efter ordlängd.
      Listan får inte vara tom.
      Stabilitet garanteras inte.
  */
  void sort() {
    …
  }
}
```

Javadoc (som docstring):
Beskriv gärna mer om
vad koden lovar och *inte* lovar

Bryter man mot krav på indata
(tom lista)
får vad som helst hända…

Kontrakt 3: Javadoc

- Javadoc i **rätt format**, på **rätt plats** ➔ verktyg kan hitta den!

```
/**
 * Add a component to a container and set its layout constraints, assuming that the component does not grow
 * margins.
 *
 * @param container The container to which the component should be added
 * @param component The component that is to be added
 * @param gridx     The logical x-coordinate of the component (zero-based).
 * @param gridy     The logical y-coordinate of the component (zero-based).
 * @param gridw     The number of columns that the component should span.
 * @param gridh     The number of rows that the component should span.
 */
public static void add(final Container container, final int gridx, final int gridy, final int gridw, final i
{
    add(container, g                                                                    ttom: 0,
}

////////////////////

static class PLAFMen
{
    private final Fr

    PLAFMenu(final F
```

```
gui.AWTdemo
public static void add(Container container,
                       int gridx,
                       int gridy,
                       int gridw,
                       int gridh,
                       Component component)
```

Add a component to a container and set its layout constraints, assuming that the component does not grow and has no margins.

Params: container – The container to which the component should be added
gridx – The logical x-coordinate of the component (zero-based).
gridy – The logical y-coordinate of the component (zero-based).

Grundregel för kontrakt:

Dokumentera alltid vad koden **kräver**
och vad den **lovar**.

Relaterad grundregel:

Dokumentera inte vad koden **gör**

Specificera vad den **ska göra**
så man vet **varför man vill anropa den**
och **om den gör det man är ute efter**