Viktiga programmeringsbegrepp: Kontrakt

Vad lovar {klassen, metoden, fältet}?

jonas.kvarnstrom@liu.se – 2018

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?
  - …

Kod kan innehålla (vissa) formella kontrakt

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 ;
   }
}
```

Krav på input: Parametrarna måste vara av typ double

Löfte om resultattyp: Returnerar alltid double

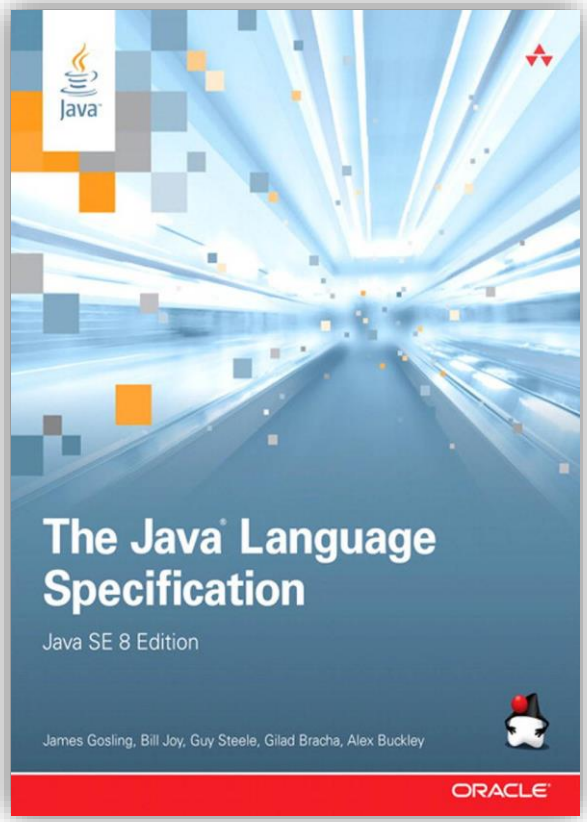
Följer direkt från manifest typning
Så vanligt att man oftast inte ens ser det som kontrakt

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!

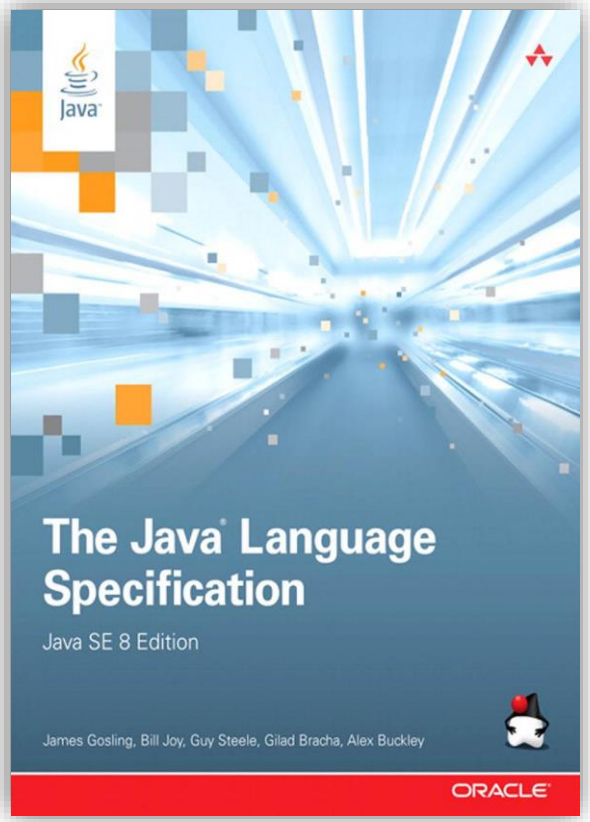
Varför räcker inte detta?

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 ;
   }
}
```

+



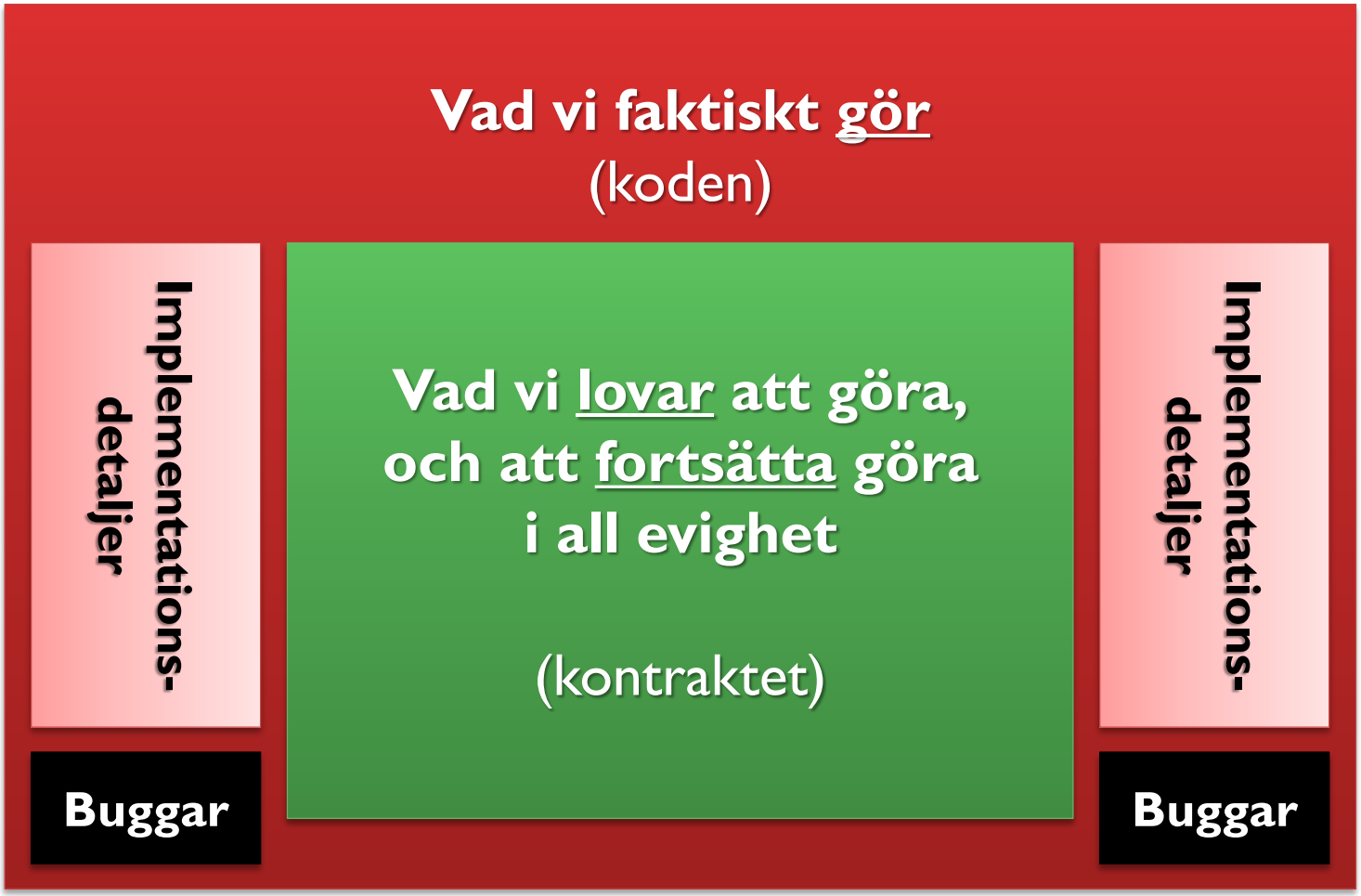
Svårt att inse alla konsekvenser av tusentals rader kod

Kontraktet bör 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: Vad utlovas?

jonkv@ida

- Får jag ("Cirkel-användare") skapa en cirkel med radie 0?
  - **Ser** inget som skulle orsaka problem…
  - Men vem vet hur klassen kan **ändras i framtiden**?

```
class Circle {
  double x, y, r;

  Circle(double x, double y, double r) {

    this.x = x;
    this.y = y;
    this.r = r;
  }
}
```

Exempel: Vad är tillåtet?

jonkv@ida

- Får jag ("Cirkel-utvecklare") ändra min kod så här?
  - Verkar rimligt att cirklar ska ha positiv radie
  - Men kod som förut fungerade kommer nu att krascha!

```
class Circle {
  double x, y, r;

  Circle(double x, double y, double r) {

    this.x = x;
    this.y = y;
    this.r = r;
  }
}
```

```
class Circle {
  double x, y, r;

  Circle(double x, double y, double r) {
    if (r <= 0.0) {
      // Signalera fel...
      throw ...;
    }
    this.x = x;
    this.y = y;
    this.r = r;
  }
}
```

Kontrakt 1

jonkv@ida

- Ange löften genom kontrakt!
  - Vissa språk har formellt stöd – 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

- Ju tydligare, desto bättre…

```
class List {
  …

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

Beskriv gärna mer om
vad koden lovar och *inte* lovar

Exempel 2: Kontrakt?

jonkv@ida

- Med kontrakttänkande:

```
class Circle {
  double x, y, r;
  /** Skapar en cirkel med angivna
      koordinater och radie.
   */
  Circle(double x, double y, double r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }
}
```

Metoden tar emot double.

Det står inget om begränsningar.

Alltså lovar den att klara godtycklig double.

Exempel 2: Kontrakt?

jonkv@ida

- Med kontrakttänkande:

```
class Circle {
  double x, y, r;
  /** Skapar en cirkel med angivna
      koordinater och radie. Radien
      måste vara strikt positiv.
   */
  Circle(double x, double y, double r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }
}
```

Lovar att klara strikt positiva radier.

Kräver sådan input.

Skickar du in annat är beteendet **odefinierat** (kan krascha, …)

Grundregel för kontrakt:

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

Relaterad grundregel:

Dokumentera inte vad koden **gör**
utan vad den **ska göra**
(och gärna **varför**)