

OOMPA 2001

Föreläsning 2
Introduktion till Java del b

Mer om Java
bla this och konstruktörer
något super- och subklasser

◀ previous | next ▶

DOMPAE-010903 b Introduktion Java, del b

Exempel: lampa

Light1
isOn : boolean
+ turnOn() : void
+ turnOff() : void
+ isOn() : boolean
+ toggle() : void
+ <u>main(String []) : void</u>

```
graph LR; Start(( )) --> off[off]; off -- toggle --> on[on]; on -- toggle --> off;
```

◀ previous | next ▶ 2

DOMPAE-010903 b Introduktion Java, del b

... Light1, Javakod

```
public class Light1 {  
    protected boolean isOn;  
    public void turnOn() {isOn = true;}  
    public void turnOff() {isOn = false;}  
    public boolean isOn() {return isOn;}  
    public void toggle() {isOn = !isOn();}  
  
    public static void main (String args[]) {  
        Light1 light;  
        light = new Light1();  
        System.out.println(light.isOn()); //ska ge false  
        light.toggle();  
        System.out.println(light.isOn()); // ska ge true  
    }  
}
```

◀ previous | next ▶ 3

DOMPAE-010903 b Introduktion Java, del b

Exempel: lampa2, med start i on-läge

```
graph LR  
    Start(( )) --> on  
    on -- toggle --> off  
    off -- toggle --> on
```

◀ previous | next ▶ 4

DOMPAE-010903 *Introduktion Java, del b*

... Light2, Javakod

```
public class Light2 {
    // defaultvärde för isOn kan anges vid deklarationen

    // medlemsmetoderna som förut...

    public static void main (String args[]) {
        Light2 light;
        light = new Light2();
        System.out.println(light.isOn()); //true
        light.toggle();
        System.out.println(light.isOn()); //false
    }
}
```

◀ previous | next ▶

5

DOMPAE-010903 *Introduktion Java, del b*

Klassvariabel

```
// Klassvariabel anges med
public class Circle {
```

- klassvariabel
- vi kan använda klassvariabeln för att tex initiera instanvariabler
- konstant (som ej kan ändras) med final

```
}
```

◀ previous | next ▶

6

DOMPAE-010903 Introduktion Java, del b

Exempel: Light3

```
public class Light3 {  
  
    public static void main (String args[]) {  
        Light3 light;  
        light = new Light3();  
        System.out.println(light.isOn());  
        light.toggle();  
        System.out.println(light.isOn());  
    }  
}
```

◀ previous | next ▶ 7

DOMPAE-010903 Introduktion Java, del b

Konstruktör

- En konstruktör är en speciell rutin som instansierar en klass
 - I Java har en **konstruktör samma namn som klassen**
 - Exempel: konstruktorn för klassen `Circle` heter `Circle()` och objekt av klassen skapas med hjälp av den på följande sätt

```
Circle c1 = new Circle();
```
 - En konstruktör skrivs på samma sätt som en metod

```
public Circle() {  
}
```
 - Observera att i Java anges inget returvärde för en konstruktör utan detta är alltid (underförstått) en instans av aktuell klass
 - Detta gör också att vi syntaktiskt kan skilja på konstruktörer och andra metoder

◀ previous | next ▶ 8

DOMPAE-010903 Introduktion Java, del b

... konstruktor

- Skriver vi ingen konstruktor för klassen så "låter" Java oss ändå använda en konstruktor utan argument för klassen
 - Det finns dock vissa undantag som vi diskuterar nedan under diskussion om flera konstruktörer
- Varför skriva egen konstruktor?
 - Jo, vi vill kontrollera hur ett objekt skapas och se till att dom rätta initieringarna görs

◀ previous | next ▶

9

DOMPAE-010903 Introduktion Java, del b

Exempel: konstruktor för som initierar instansvariabler

- Om vi definierar följande konstruktor


```

                {
                x = 10;
                y = 20;
                r = (x + y) / 2;
                }
            
```
- Instansierar


```

                Circle c = new Circle();
            
```
- Så får `c.x` värdet 10, `c.y` värdet 20 och `c.r` värdet 15 i samband med instansieringen

◀ previous | next ▶

10

DOMPAE-010903 Introduktion Java, del b

Pseudovariabeln

- För att referera aktuell instans så kan pseudovariabeln `this` användas
 - Om vi tex vill skicka aktuellt objekt som argument vid en viss medelandesändning gör vi i stil med följande


```
mottagare.meddelande(this);
```
 - exempelvis vid utmatning på terminalen


```
System.out.println(this)
```
 - Om vi explicit vill förtydliga att ett meddelande skickas till aktuellt objekt kan vi skriva


```
this.meddelande();
```
 - Vi kan också explicit referera objektets instansvariabler, tex


```
this.r
```

◀ previous | next ▶ 11

DOMPAE-010903 Introduktion Java, del b

Flera konstruktörer

- Man kan skapa flera konstruktörer för en viss klass
- Konstruktörerna skiljer sig åt genom olika signatur, dvs olika antal argument eller olika typ på argumenten
- Tex är följande olika (samtidiga) konstruktörer möjliga


```
public Klass() {}
public Klass(T a) {}
public Klass(X a) {}
public Klass(A a, B b) {}
public Klass(A a, C c) {}
```
- Vilken konstruktör som används beror på vilka argument som ges vid instansieringen

◀ previous | next ▶ 12

DOMPAE-010903 Introduktion Java, del b

... Flera konstruktorer

- Argumenten kan ha samma namn som instansvariablerna


```
public Circle (double x, double y, double r)
// men då måste vi referera instansvariablerna mha this.IVAR

    this.r = r;}

```
- definition av en konstruktor som tar en annan cirkel som argument


```
public Circle(Circle c)

```
- olika typer av argument ger olika metoder (överlagring)


```
public Circle(double r)

```

◀ previous | next ▶ 13

DOMPAE-010903 Introduktion Java, del b

Exempel: Light4 (med def av konstruktorer)

```
public class Light4 {

    public static void main (String args[]) {
        Light4 light1 = new Light4();
        Light4 light2 = new Light4(true);
        System.out.println("Lampa 1: " + light1.isOn() + ",
                           Lampa 2: " + light2.isOn());

        light1.toggle();
        light2.toggle();
        System.out.println("Lampa 1: " + light1.isOn() + ",
                           Lampa 2: " + light2.isOn());
    }
}

```

◀ previous | next ▶ 14

DOMPAE-010903 Introduktion Java, del b

Konstruktionen för att anropa annan konstruktor

- Ibland har man flera konstruktörer i en klass
- I många fall vill man då anropa en konstruktor från en annan
 - tex skulle `Light4()` kunna utnyttja `Light4(boolean bool)` i föregående exempel
 - Fördelen är att man bättre isolerar ett visst beteende till ett ställe. Om man senare behöver ändra något så görs det på så få ställen som möjligt
- I java kan man göra detta genom att använda konstruktionen `this(...)`, tex skulle vi kunna skriva om `public Light4()` på följande sätt:


```
public Light4() {this(false);}
```
- Rätt konstruktor anropas beroende av antal argument och deras typ.
- OBSERVERA! `this(...)` kan bara användas en gång per konstruktor och måste stå först!

◀ previous | next ▶

15

DOMPAE-010903 Introduktion Java, del b

Exempel: Light5 (en konstruktor använder en annan)

```
public class Light5 {

    public static void main (String args[]) {
        Light5 light1 = new Light5();
        Light5 light2 = new Light5(true);
        System.out.println("Lampa 1: " + light1.isOn() + ",
                           Lampa 2: " + light2.isOn());

        light1.toggle();
        light2.toggle();
        System.out.println("Lampa 1: " + light1.isOn() + ",
                           Lampa 2: " + light2.isOn());
    }
}
```

◀ previous | next ▶

16

DOMPAE-010903 Introduktion Java, del b

Exempel: Book

```
public class Book {
    protected String title, author, isbn;

    public Book(String name, String author, String isbn) {
        title = name; this.author = author; this.isbn = isbn;
    }
    public Book(String name) {this(name, "");}
    public Book(String name, String author){
        this(name, author, "");
    }
}

public String toString()
{return "Titel: " + title + " Förf: " + author + " ISBN: " +
    isbn;}

public static void main (String args[]) {
    Book book1 = new Book("XXX with Java", "Budd");
    Book book2 = new Book("Y Undistilled");
    System.out.println("Bok 1: " + book1);
    System.out.println("Bok 2: " + book2);
}
}
```

◀ previous | next ▶ 17

DOMPAE-010903 Introduktion Java, del b

Konstruktörer: några saker man bör vara medveten om

- Skriver man en egen konstruktör som tar argument kan man inte längre använda den från början givna konstruktören utan argument, dvs skriver man

```
public Klass(Typ t) {}
```
- Så kan man inte instansiera på följande sätt

```
Klass k = new Klass();
```
- Om man fortfarande vill använda en konstruktör utan argument så måste man explicit implementera en sådan i stil med

```
public Klass() {}
```

◀ previous | next ▶ 18

DOMPAE-010903 Introduktion Java, del b

...

- Observera att om en superklass implementerar en konstruktör med argument men ingen utan så kan inte heller subclasser använda "defaultkonstruktören" utan argument, utan måste om dom så önskar explicit implementera en konstruktör utan argument
- Det finns ett problem till och det är att en subclass konstruktör implicit anropar super-klassens konstruktör om inte `this(...)` eller `super(...)` explicit används

◀ previous | next ▶ 19

DOMPAE-010903 Introduktion Java, del b

Inkludera klasser från andra filer

importera Point

```

public class Figure {
    /* med protected kan variabeln enbart läsas av instanser
       eller instanser av subclasser */
    protected Point position = new Point(10, 20);

    // fast vi kan ge en publik inspektor
    public Point position() {return position;}

    // en annan metod kan anropa direkt
    public int x() {return position().x;}

    // eller med this.MEDDELANDE
    public int y() {return this.position().y;}
}
    
```

◀ previous | next ▶ 20

DOMPAE-010903 Introduktion Java, del b

testa

```
public static void main (String args[]) {
    //Vi deklarerar en temporär variabel
    Figure f = new Figure();
    System.out.println("pos: " + f.position() +
        " x: " + f.x() + " y: " +
        f.y());
    }
}
```

- Resultat
pos: java.awt.Point[x=10,y=20] x: 10 y: 20

◀ previous | next ▶ 21

DOMPAE-010903 Introduktion Java, del b

Javaexempel : Enkellänkad lista

```
/* en enkellänkad lista med metoder för att undersöka om objektet är
   sist, stoppa in länk samt ge avstånd till slutet */
class LinkableObject{
    public LinkableObject link;
    public Object value;
    public String toString() {return value.toString();}
    public LinkableObject() {this(null);}
    public LinkableObject(Object value) {
        link = null;
        this.value = value;}
    public boolean atEnd(){return link == null;}
    public void insert(LinkableObject next) {
        link = next;}
    public int distanceToEnd(){
        return atEnd() ? 0 : 1 + link.distanceToEnd();
    }
}
```

◀ previous | next ▶ 22

DOMPAE-010903 Introduktion Java, del b

```

public class LinkableTestB{
    public static void main (String args[]) {
        LinkableObject root = new LinkableObject("ROOT");
        LinkableObject current;
        current = root;
        for(int i=1; i < 5; i++) {
            // För att kunna använda en int som ett Object
            // konstruerar vi en Integer
            LinkableObject newLinkable =
                new LinkableObject(new Integer(i));
            current.insert(newLinkable);
            current = newLinkable;}
        current = root;
        while(current != null) {
            System.out.println(current + " distance to end: " +
                current.distanceToEnd());
            current = current.link;}
    }
}

```

◀ previous | next ▶ 23

DOMPAE-010903 Introduktion Java, del b

Javaexempel : Dubbellänkad lista

```

class DoubleLinkableObject{
    public DoubleLinkableObject next, prev;
    public Object value;
    public String toString() {return value.toString();}

    public DoubleLinkableObject() {this(null);}
    public DoubleLinkableObject(Object value){
        next = prev = this;
        this.value = value;
    }
    public void insert(DoubleLinkableObject aDoubleLinkableObject) {
        this.next.prev = aDoubleLinkableObject;
        aDoubleLinkableObject.next = this.next;
        aDoubleLinkableObject.prev = this;
        next = aDoubleLinkableObject;}
    public int distanceTo(DoubleLinkableObject aDoubleLinkableObject){
        return this == aDoubleLinkableObject ? 0 : 1 +
            next.distanceTo(aDoubleLinkableObject);}
    public int length(){return 1 + next.distanceTo(this);}
}

```

◀ previous | next ▶ 24

DOMPAE-010903 Introduktion Java, del b

```
public class DoubleLinkableTest{
    public static void main (String args[]) {
        DoubleLinkableObject root = new DoubleLinkableObject("ROOT");
        DoubleLinkableObject current;
        current = root;
        for(int i=1; i < 5; i++) {
            // För att kunna använda en int som ett Object
            // konstruerar vi en Integer
            DoubleLinkableObject newLinkable =
                new DoubleLinkableObject(new Integer( i));
            current.insert(newLinkable);
            current = newLinkable;
        }
        current = root.next;
        while(current != root) {
            System.out.println(current + " distance to root: " +
                current.distanceTo(root));
            current = current.next;}
        System.out.println( "length: " + root.length());
    }
}
```

◀ previous | next ▶ 25

DOMPAE-010903 Introduktion Java, del b

Introduktion till arv

- Vi implementerar följande

```
classDiagram
    class Person {
        name
        phone
    }
    class Surfer {
        email
        webaddress
    }
    Person <|-- Surfer
```

◀ previous | next ▶ 26