



Programmering för Språkteknologer II

Markus Saers
markus.saers@lingfil.uu.se
Rum 9-2040
stp.lingfil.uu.se/~markuss/ht09/pst2



Inledning

Om kursen
Repetition



Om kursen

Tid
Mål
Examination
Förra årets kurs
Upplägg
Innehåll
Planering



Tid

Börjar idag (2009-09-01)
Slutar om en månad (2009-10-01)
Tenta 2009-10-02
Omtenta mitten/sent november



Mål

Efter avslutad kurs skall studenten för att förtjäna betyget Godkänd minst kunna redogöra för följande begrepp och skriva fungerande Javaprogram som exemplifierar och drar nytta av dem:



Mål

- hashtabeller och mappningar
- paket och synlighet
- objektorientering: arv, polymorfism, abstrakta klasser, gränssnitt
- matchning med reguljära uttryck
- stackar, köer och länkade listor
- undantag
- sökning och sortering
- ändliga automater



Examination

Kunskapskontroll sker genom muntligt och/eller skriftligt **redovisade uppgifter** och/eller **tentamen**. Läraren kan som del av examinationen kräva obligatorisk närvaro och aktivt deltagande vid undervisningsmoment. Kurskraven meddelas skriftligen av kursansvarig vid kursens start.



Examination

Examinationsmoment

Laborationer (4 st.)
Stackar och länkade listor
Ändliga automater och reguljära uttryck
Sortering
God programmering

Tenta

Betygskriterier: G

Godkänt på samtliga moment

Betygskriterier: VG

Sammanlagt 3 VG moment (tentan räknas som 2)



Förra årets kurs

Helhetsintryck 3.00

Det har varit bra med genomgång av exempel på tavlan.
Bra också att OH-serierna ligger ute på kursidan i tid.

Undervisning 3.33

Tycker fortfarande att det kan vara lite svårt att komma igång med labbarna. Men det handlar väl till viss del om arbetssättet också.

Jag skulle vilja att fler exempelprogram går igenom steg för steg [...]

Arbetsklimat 3.33

Gärna mindre "tänk lite på det" när man frågar om hjälp på labbar och fler konkreta tips vad man kan göra istället.



Förra årets kurs

Kurslitteratur 2.00

Java med swing saknar väldigt ofta relevanta exempel i kod.

Den bygger på att man läser den pärm till pärm, eftersom den hela tiden hänvisar till tidigare exempel och program. Fungerar alltså ej bra som uppslagsverk.

Examination 4.00

Tentamen och laborationer passar bra.

Administration 3.67

Lite konstig möblering med tanke på användning av tavlan och projektor, svårt att se ibland.



Upplägg

Lektion på förmiddagen

Turing

Laboration på eftermiddagen

Chomsky

Obligatoriska inlämningsuppgifter

Samt frivilliga övningsuppgifter

Tentamen



Innehåll

Mer objektorienterad programmering

Arv, polymorfism, abstrakta klasser, gränssnitt, synlighet, paket och undantag

Datastrukturer

Stackar, köer och länkade listor

Hashtabeller och associativa strukturer

Sökning och sortering

Ändliga automater



Preliminär planering

Arv, polymorfism, abstrakta klasser, gränssnitt, stackar, köer och länkade listor (3–8/9)
 Synlighet, paket och undantag (10/9)
 Ändliga automater, sökning och sortering (15–24/9, Mats)
 Hashtabeller, associativa strukturer och undantag (25–29/9)
 Extra lektion+lab (1/10)
 Tenta



Frågor?



Repetition

Datorer
 Grundläggande Java
 Kontrollstrukturer
 Metoder
 Vektorer
 Klasser/Objekt

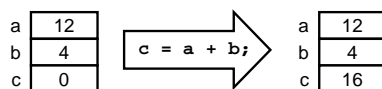


Datorn

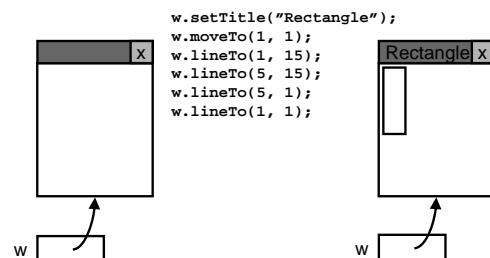
Arbetsminne
 Lagrar kortsiktigt information
 Variabler, värden, objekt, etc.
 Processor
 Förändrar arbetsminnet
 Övrigt
 Skärm, tangentbord, hårddisk, etc.



Arbetsminne/Processor



Arbetsminne/Processor





Grundläggande Java

Variabler

Deklaration
Tilldelning

Enkla typer

Heltal (**byte**, **short**, **int**, **long**)
Decimaltal (**float**, **double**)
Sanningsvärden (**boolean**)
Index i teckentabeller (**char**)



Variabler

En "minneslåda" med...

Namn (så att vi kan använda namnet istället för värdet)

Typ (så att vi vet vad det är för sorts värde)

int a



Variabler

Deklareras genom att man skriver...

vilken sorts variabel man vill ha (typ)
vad man vill att den ska heta

```
int a;
```

Kan anta alla värden av den deklarerade typen

```
a = 12;  
a = a + 1;  
a = 12 * 4 + 2;
```

Deklaration och tilldelning kan kombineras

```
int a = 12;
```



Programsatser

Instruktioner

Utförs i den ordning de står

Avslutas med semikolon

```
int a = 12;  
a = a + 1;  
System.out.println(a);
```



Att köra programsatser

När man kör ett Javaprogram

```
$ java MyClass
```

Letar Java efter en metod i klassen som heter **main**, vars programsatser utförs

```
public class MyClass {  
    public static void main(String[] args) {  
        // Programsatserna här utförs  
    }  
}
```



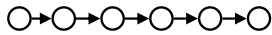
Kontrollstrukturer

Normalt körs programsatser i den ordning de står

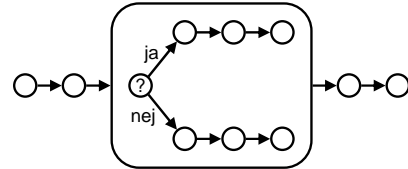
Ibland vill man villkora satser

Ibland vill man upprepa satser

Normalt flöde



Villkorat flöde



Villkorat flöde

```

if (/* Villkor */) {
  // Övre flödet
} else {
  // Nedre flödet
}

```

Villkorat flöde

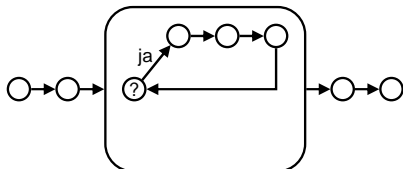
```

if (/* Villkor */) {
  // Övre flödet
} else {
  // Nedre flödet
}

if (tal < 10) {
  tal = 10;
} else {
  System.out.println(tal);
}

```

Upprepat flöde



Upprepat flöde

```

while (/* villkor */) {
  // Övre flödet
}

```



Upprepat flöde

```

    while (/* villkor */) {
    // Övre flödet
    }

    int i = 0;
    while (i < 10) {
        System.out.println(i);
        i++;
    }

```



Vanligt upprepningsflöde

```

    init
    while ( cond ) {
        ...
        step
    }

    for (init ; cond ; step) {
        ...
    }

```



Vanligt upprepningsflöde

```

    int i = 0;
    while ( cond ) {
        ...
        step
    }

    for (int i = 0; cond ; step) {
        ...
    }

```



Vanligt upprepningsflöde

```

    int i = 0;
    while (i < 10) {
        ...
        step
    }

    for (int i = 0; i < 10; step) {
        ...
    }

```



Vanligt upprepningsflöde

```

    int i = 0;
    while (i < 10) {
        ...
        i++;
    }

    for (int i = 0; i < 10; i++ ) {
        ...
    }

```



Vanligt upprepningsflöde

```

    int i = 0;
    while (i < 10) {
        System.out.println(i);
        i++;
    }

    for (int i = 0; i < 10; i++ ) {
        System.out.println(i);
    }

```



UPPSALA
UNIVERSITET

Metoder

Ett antal programsatser som beskriver en metod för att utföra något

Kan ta ett antal argument/parametrar som är meningsfulla

Kan ge tillbaka ett resultat

Måste inte



UPPSALA
UNIVERSITET

Metoder

Metoden addition

Har två tal som argument

Utförs genom att det ena talet läggs till det andra

Resulterar i summan av dessa två tal

```
int add(int tal1, int tal2) {  
    int sum = tal1 + tal2;  
    return sum;  
}
```



UPPSALA
UNIVERSITET

Metoder

Metoden addition

Har två tal som argument

Utförs genom att det ena talet läggs till det andra

Resulterar i summan av dessa två tal

```
int add(int tal1, int tal2) {  
    return tal1 + tal2;  
}
```



UPPSALA
UNIVERSITET

Metoder

Metoden addition

Har två tal som argument

Utförs genom att det ena talet läggs till det andra

Resulterar i summan av dessa två tal

```
public int add(int tal1, int tal2) {  
    return tal1 + tal2;  
}
```



UPPSALA
UNIVERSITET

Metoder

Metoden addition

Har två tal som argument

Utförs genom att det ena talet läggs till det andra

Resulterar i summan av dessa två tal

```
public static int add(int tal1, int tal2)  
{  
    return tal1 + tal2;  
}
```



UPPSALA
UNIVERSITET

Vektorer

En variabel kan användas för att komma åt en serie värden

Antalet måste deklarerats

Skiljs åt med hjälp av index



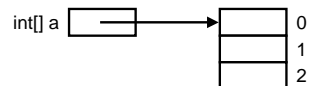
Vektorer

En variabel kan användas för att komma åt en serie värden

Antalet måste deklarerats

Skiljs åt med hjälp av index

```
int[] a = new int[3];
```



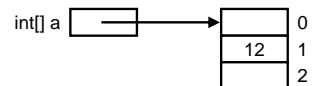
Vektorer

En variabel kan användas för att komma åt en serie värden

Antalet måste deklarerats

Skiljs åt med hjälp av index

```
int[] a = new int[3];
a[1] = 12;
```



Vektorer

Kan också initialiseras explicit

```
int[] b = { 1, 2, 3 };
```

Håller själva reda på hur långa de är

```
int lengthOfB = b.length;
```

Indexeras från 0

Giltiga index i för vektor b : $0 \leq i < b.length$

```
Vi kan löpa igenom alla giltiga index med
for (int i = 0; i < b.length; i++) {
    ...
}
```



Klasser/Objekt

Klasser

En samling variabler och metoder

"Ritning" för hur objekt ska se ut

Objekt

En samling variabler och metoder



Klasser

Allt som är statiskt finns i klassen själv

Klassvariabler

Klassmetoder

Allt annat utgör en ritning för objekt av klassen

Attribut

Metoder

Konstruktörer



Objekt

En samlig data (attribut) som kan förändra sig självt (metoder)

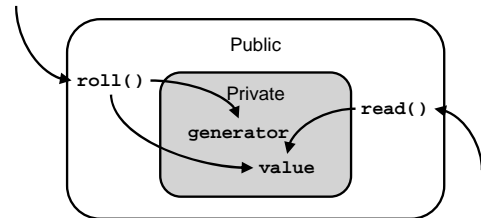
Bra exempel kommer senare på kursen!

Programmeraren behöver inte känna till hur den underliggande informationen ska hanteras

Objekt

```
import java.util.Random;
public class Die {
    private Random generator;
    private int value;
    public Die() {
        generator = new Random();
        roll();
    }
    public void roll() {
        value = generator.nextInt(6);
    }
    public int read() {
        return value + 1;
    }
}
```

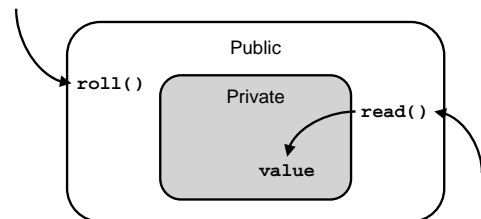
Objekt: Die



Objekt

```
public class LoadedDie {
    private int value;
    public LoadedDie(int value) {
        this.value = value;
    }
    public void roll() {
    }
    public int read() {
        return value + 1;
    }
}
```

Objekt: LoadedDie



Frågor?

Bonusbit av nästa lektion!

Abstrakta klasser
Lite synlighet



Tärningar

```

import java.util.Random;
public class Die {
    private Random generator;
    private int value;
    public Die() {
        generator = new Random();
        roll();
    }
    public void roll() {
        value = generator.nextInt(6);
    }
    public int read() {
        return value + 1;
    }
}

public class LoadedDie {
    private int value;
    public LoadedDie(int value) {
        this.value = value;
    }
    public void roll() {
    }
    public int read() {
        return value + 1;
    }
}

```



Tärningar

```

import java.util.Random;
public class Die {
    private Random generator;
    private int value;
    public Die() {
        generator = new Random();
        roll();
    }
    public void roll() {
        value = generator.nextInt(6);
    }
    public int read() {
        return value + 1;
    }
}

public class LoadedDie {
    private int value;
    public LoadedDie(int value) {
        this.value = value;
    }
    public void roll() {
    }
    public int read() {
        return value + 1;
    }
}

```



Tärningar

```

public class AbstractDie {
    private int value;
    public AbstractDie() {
    }
    public void roll() {
    }
    public int read() {
        return value + 1;
    }
}

```



En abstrakt tärning

```

public abstract class
AbstractDie {
    private int value;
    public AbstractDie() {
    }
    public abstract void roll();

    public int read() {
        return value + 1;
    }
}

```



Konkret tärning

```

import java.util.Random;
public class Die extends AbstractDie {
    private Random generator;
    public Die() {
        super();
        generator = new Random();
        roll();
    }
    public void roll() {
        value = generator.nextInt(6);
    }
}

```

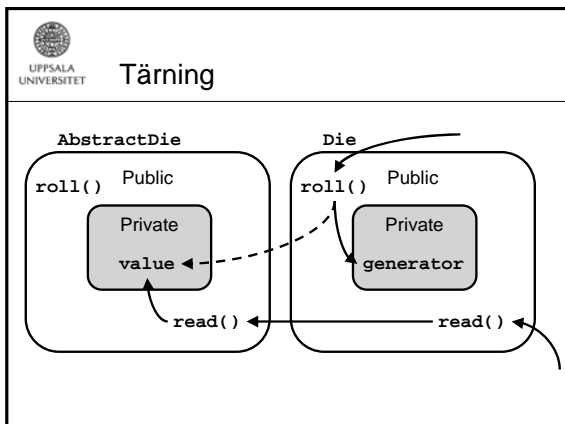


Konkret fusk tärning

```

public class LoadedDie extends
AbstractDie {
    public LoadedDie(int value) {
        super();
        this.value = value;
    }
    public void roll() {
    }
}

```



UPPSALA UNIVERSITET

Att blotta sig för sina barn

Vi vill att `Die` ska kunna se attributet `value` i `AbstractDie`
 Attributet `value` i `AbstractDie` är privat
 Bör man dölja allt för sina barn?
 Det går att blotta sig med det reserverade ordet `protected`

- Gör saker synliga för underklasser (barn)
- Och för andra klasser i samma paket (familjen)
- Mer om paket senare

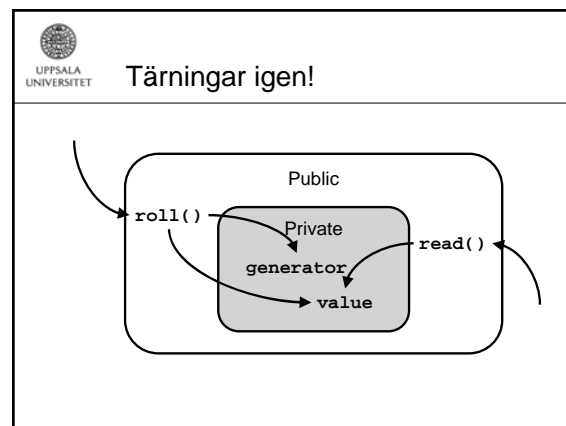
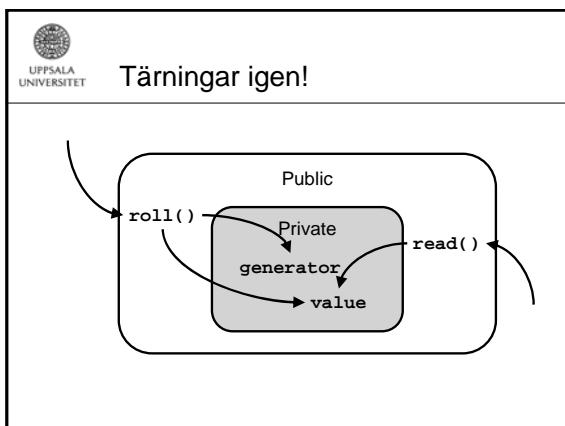
UPPSALA UNIVERSITET

Frågor?

UPPSALA UNIVERSITET

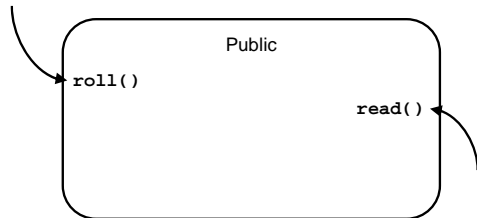
Markus planerar dåligt!

Gränssnitt





Tärningar igen!



Gränssnitt

- Alla publika metoder i en klass
- Kan formaliseras
 - Deklarerar vilka metoder som ska finnas
- Kan implementeras av en klass
 - Definierar hur metoderna utförs



Mera tärningar!

```
public interface DieInterface {
    void roll();
    int read();
}

public abstract class AbstractDie
implements DieInterface {
    private int value;
    public AbstractDie() {
    }
    public int read() {
        return value + 1;
    }
}
```



Multipla arv

- En klass kan bara ärva från en klass
- Men kan implementera hur många gränssnitt som helst.
- Javas version av konceptet multipla arv