Java, klasser, objekt (Skansholm: Kapitel 2)

Sven-Olof Nyström

Uppsala Universitet

11 mars 2005

Vad är en klass?

- En klass är ett sätt att beskriva en mängd objekt och deras gemensamma egenskaper.
- En klassdefinition innehåller datafält ...
- ... och metoder (= “funktioner” i C).
- Man kan också säga: En klassdefinition beskriver kod och data.
- Ett Javaprogram består av en eller flera klassdefinitioner (och ingenting kan finnas utanför en klassdefinition).

Klasser och objekt

- Varje objekt tillhör en viss klass. (Objektets klass är detsamma som objektets typ.)
- Vi säger att objektet är en *instans* av klassen.
- Alla objekt av en viss klass kan användas på samma sätt–de har samma “gränssnitt”.
- En definition av en viss klass kan ses som en mall för objekt av den typen.

Klassdefinition: Exempel

```
class Person {
    int ålder;
    String namn;
}
```

- Konvention: Namnet på en klass skrivs med stor bokstav.
- ’namn’ och ’ålder’ är *fält* i objekt av typen Person.
- Skansholm: instansvariabler
- Engelska: field

Klassdefinition: Körexempel

```
Person x;

x = new Person ();

x.ålder = 42;
x.namn = "Kalle Karlsson";

System.out.println(
"Personen " + x.namn + " är " + x.ålder
    + " år gammal");
```

ger utskriften Personen Kalle Karlsson är 42 år gammal

Referensvariabler, exempel 1

- Givet tidigare klassdefinition för Person
- och deklarationen Person x;
- Variablen x får värdet null (default-värde)

Referensvariabler, exempel 2

```
Person x;
Person y;

x = new Person ();
y = x;

x.ålder = 42;
x.namn = "Kalle Karlsson";
```

Vilken ålder har y?

Referensvariabler och referenssemantik

- Givet en klassdefinition

  class X { ... }

- och en variabeldeklaration

  X a;

- Variabeln a lagrar en *referens*.

Referensvariabler (forts)

- Exempel:

  ```
  X a, b;
  a = new X();
  b = a;
  ```

- Nu refererar a och b till samma objekt.

Referensvariabel, regel

- Variabler av klasstyp har *referenssemantik*.
- Nya objekt skapas med new.
- Tilldelning kopierar referensen, inte objektet.

Metoder (“Funktioner i klasser”)

```
class A {
   returtyp metodnamn () {
      ...
   }
}
```

Enkelt exempel: metod och fält

```
class A {
   int x = 7;
   int x_plus_ett() {
      return x + 1;
   }
}
```

Om variabeln a är deklarerad och initializerad enligt

A a = new A();

Vad returnerar uttrycket

a.x_plus_ett()?

Metoder (forts)

- Om inget värde ska returneras, skriv void i stället för returtyp.
- En variabel kan deklareras lokalt inom en metod (ungefär som inom en C-funktion). Kallas “lokala variabler”.
- Lokala variabler måste **alltid** ges startvärden.

En return-sats gör två saker:

1. anger vilket värde som ska ges som resultat från metoden
2. avslutar metoden

- “return;” avslutar anropet för en metod som inte returnerar nåt värde (void).
- “return Uttryck;” beräknar uttrycket och avslutar anropet genom att returnera det beräknade värdet.
- Bara void-metoder får nå slutet av metodkroppen utan att stöta på ett return. Kontrollen återvänder automatiskt till anroparen.

Metoder: exempel

```
class A {
  int x = 5;
  int get_x() { return x; }
  }

class B {
  int m1() { return 7; }
  int m2() {
    A a = new A();
    return m1() + a.get_x();
    }
}
```

Parametrar och argument

- En metod kan ges en parameterlista, tex

  int f(int x, float y) { ... }

- En metods *parametrar* skrivs inom parenteserna i definitionen. Fungerar som lokala variabler, men får sina startvärden av anroparen.
- Med *argumenten* till ett anrop menas de faktiska värden som anroparen beräknar och skickar till metoden.
- När ett objekt passas som argument kopieras referensen (precis som vid tilldelning)

Parametrar, exempel

```
class A {

   int add(int a, int b) {
      return a + b;
}

   int m() {
      int x = 4;
      return add(x,3) + 10;

}}
```

Referensvariabler i metodanrop

- Med en metoddefinition

  void f(X c) { ... }

  och ett anrop

  f(a)

  kommer variabeln c inom metodkroppen att referera till samma objekt som a

- Regel: Objekt kopieras *aldrig* (om man inte ber uttryckligen om det).

Konstruktorer: Att skapa nya objekt

Exempel:

```
class Person {
    int ålder;
    String namn;
    Person (String n, int å) {
        ålder = å;
        namn = n;
      }
}
```

Vi skapar en ny person med (tex)

```
Person x = new Person("Kalle", 42);
```

Konstruktorer (forts)

- En konstruktor har alltid samma namn som klassen
- En konstruktor deklareras utan resultattyp (inte ens void)
- En konstruktor ska initialisera objektets fält (instansvariabler)
- Om en klass inte har någon konstruktor, kommer Javakompilatorn att definiera en parameterlös konstruktor (defaultkonstruktorn)

Namns räckvidd

- Alla namn på instansvariabler och metoder kan kommas åt direkt inom hela den egna klassen
- Namn på lokala variabler inom ett block (metodkropp) kan bara kommas åt inom blocket (metodkroppen) De existerar bara då!

Namns räckvidd (forts)

- Om samma namn används för flera olika saker gäller den användning som är närmast".
- Instansvariabler och metoder kan alltid kommas åt genom prefixa dem med this:

  this.namn ger instansvariabeln i nuvarande klassen även om namn skulle ha flera betydelser.

Vad vi vet om objekt och klasser

- En klass definierar en typ av objekt.
- Varje klass har
  - ett antal fält (instansvariabler)
  - ett antal metoder
  - en konstruktor
- Idé: Fälten beskriver den interna representationen. Metoderna beskriver gränssnittet till omvärlden.
- Allt data är kopplat till ett visst objekt.