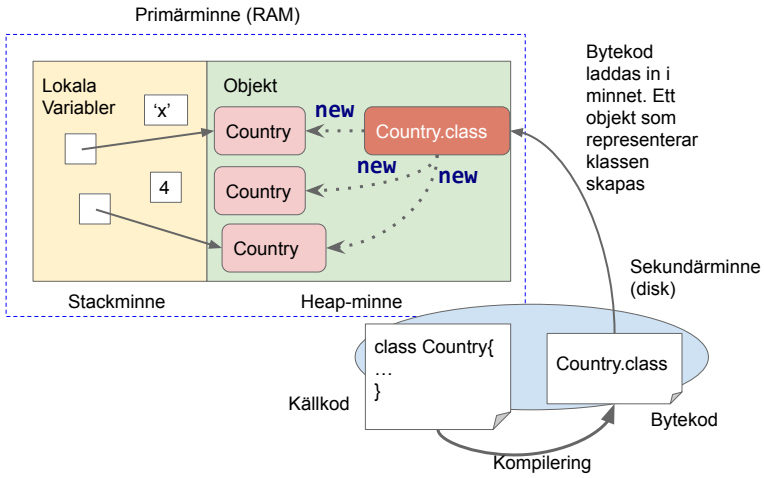
ex3static

Joachim von Hacht

Klassladdning och Instansiering



Lite mer i detalj vad som händer under körning

- När vi försöker skapar en instans kontrolleras om klassen för objektet finns i minnet …
  - … om ej så söker Java rätt på .class-filen för klassen och läser in den i minnet (det skapas ett objekt som representerar själva klassen, Country.class)
- Instanserna skapas med klassobjektet som mall då vi använder new-operatorn
  - I samband med detta körs konstruktorn
  - Returvärdet, referensen till objektet, kan sparas i en variabel på stacken.

Klassvariabler

```
class GameCharacter {

    // A class variable.
    final static Random rand = new Random();
    ...
    void turnRandom() {
        dx = 1 - rand.nextInt(3);
        dy = 1 - rand.nextInt(3);
    }
}

// No object needed, use class to access.
GameCharacter.rand.nextInt(4);
```

3

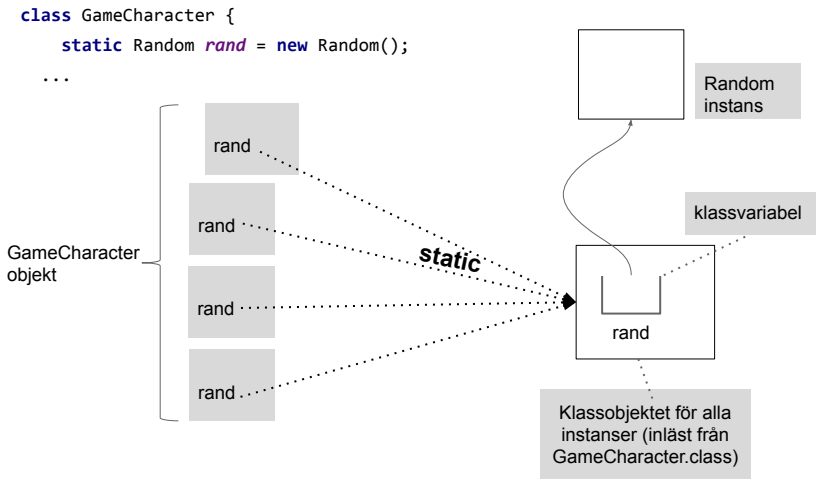
En klassvariabel tillhör inte något objekt, … den delas av alla objekt av samma klass

- Anges med static vid variabeldeklarationen
- Används ganska sällan…
  - Ett exempel i bilden: Alla GameCharacter behöver en slumpgenerator, … men de kan dela på den (onödigt att alla har en egen).
  - Slipper skicka in referens via konstruktor.
- Att en klassvariabel delas av alla objekt är riskabelt
  - På samma sätt som att instansvariabler delas av alla metoder i en klass, delas en klassvariabel av alla instanser
  - … om det blir fel, vart uppstod felet (vilket objekt som helst kommer åt variabeln)?!?!?
  - Klassvariabler bör alltid vara vara final!

För att komma åt variabeln använder man punktnotation direkt på klassnamnet (det behövs inget objekt).

- Om man skriver ett objekt före så ignoreras detta.

Klassvariabler i Minnet



Alla static variabler tillhör klassen (klassobjektet)

- Om vi anger ett namn på en klassvariabel så syftar den implicit på variabeln i klassobjektet.

Alla instanser kan komma åt ett gemensamt icke muterbart klassobjekt (objektet som skapades utifrån class-filen).

- Instanserna kan få tag i klassobjektet med metoden getClass()
- Dock kan vi inte använda t.ex. o.getClass().rand (i bilden) för att komma åt klassvariabeln.
- Kan även komma åt klassobjektet med t.ex. GameCharater.class

Klassmetoder

```
class ArrayUtils  {
    static int[] reverse (int[] a){       // Class (static) method
        int[] tmp = new int[arr.length];
        for (int i = 0; i < arr.length; i++) {
            tmp[arr.length - i - 1] = arr[i];
        }
        return tmp;
    }
}
 // Call directly on class name
int[] revArr = ArrayUtils.reverse( arr );

// Class methods in Java classes
Character.isDigit(...)
String.valueOf(...)
Math.sqrt(..)
```


För vissa metoder gäller att man inte behöver något objekt

- Metoden har inget behov av någon data förutom parametrarna
    - De är rena funktioner

Om så, verkar det onödigt att skapa objekt…

- .. detta kan man lösa i Java genom att använda klassmetoder.
- Anges med static i metodhuvudet (samma som för klassvariabler).
- Metoderna tillhör klassen (inte något objekt). På samma sätt som klassvariabler.
- För att komma åt metoderna använder man punktnotation direkt på klassnamnet!

Om man lägger till import static … så slipper man skriva klassnamnet.

- Så har vi gjort med t.ex. metoderna i Math.

Klass kontra Instans

```
int j;  // Instance variable
static int i;  // Class variable

void doIt() {          // Instance method
   out.println(i);
   out.println(j);
   this.doOther();  // Ok
}

static void doOther() {   // Class method
   out.println(i);
   out.println(j);  // Bad, which object?
   this.doIt();     // Bad, no this!
}
```

6

Instansmetoder kan använda klassvariabler och anropa klassmetoder

Klassmetoder kan inte använda instansvariabler eller anropa instansmetoder

- Vilket skulle objektet vara i så fall?? Klassmetoden kan inte veta!
- Kan speciellt inte använda this i klassmetod, finns ingen sådan referens.

Initiering

```
class MyClass {
    int i1 = i2;
    static int i2 = 4;
    final static int i3; // Ok, assigned in constructor
    MyClass() {
        i3 = i1 + i2;
    }
}
MyClass m = new MyClass();
out.println(m.i1);    // 4
out.println(m.i2);    // 4
out.println(m.i3);    // 8
```

7

Objekt initieras enligt

- Klassvariabler i skriven ordning
  - Innan någon instans har skapats, initieras då klassen laddas!
- Instansvariabler i skriven ordning …
- … därefter körs konstruktorn.
- Förenklat mer senare.

Instansvariabler som är final måste ges ett värde vid deklarationen, eller i konstruktorn, eftersom de inte kan ändras.

main-metoden

```
public class MyClass {
    // args is an array of command line arguments
    public static void main(String[] args) {
        out.println(args[0]);   // Hello
        out.println(args[1]);   // world!
    }
}


// Executing program supplying
// command line argument after program name
java MyClass Hello World!
```

8

main-metoden måste finnas i alla Java-program (i någon klass).

- Då vi startar den virtuella maskinen måste vi ange vilken klass som innehåller main
    - IntelliJ visar en grön triangel på klassikonen om klassen innehåller en main-metod
    - Om så kan den exekveras (annars visas inget Run-alternativ i menyn)

Metoden anropas automatiskt först av allt då programmet startas, innan det finns några objekt!

- Dvs metoden måste vara en klassmetod (static).
- Parametern till metoden är en lista av strängar som operativsystemet kan skicka med då programmet startas.
    - Om man startar från kommandoraden skriver man strängarna efter klassnamnet.

I vår programmall har vi alltid instantiserat ett objekt av “programmet” i main-metoden och därefter anropat instansmetoden program().

- Från program() har vi sedan anropat andra instansmetoder
- Hade vi anropat metoder direkt från main hade vi varit tvungna att gör dessa static (dålig stil/vana).

Konstanter

```
// Own constant
public class CatchTheRain {
    public final static int MAX_DROPS = 10;
    ...
}


// Predefined constants
Integer.MAX_VALUE;        // 2147483647
Integer.MIN_VALUE;        // -2147483648
Math.PI
...
```

9

Konstanter är ett speciellt begrepp i Java (till skillnad från konstant variabel. En konstant:

- Deklareras som public static final (en konstant klassvariabel)
- Primitiva variabler inget problem deklarera som ovan
- Referensvariabler
  - Objektet skall motsvara ett fixt värde (icke-muterbara)
  - Objektet skall inte användas som ett objekt d.v.s.inte anropa metoder eller indexera, bara användas som ett värde.
- Konstanter skrivs med stora bokstäver avdelade med "_" t.ex. public static final int MAX_DROPS = …
- Finns en del färdiga konstanter i olika Java-klasser.

Rent Statiska Klasser

```
 // Simplified view of class Math
public final class Math {
   /**
    * Don't let anyone instantiate this class.
    */
    private Math() {}  // private!

    public static double cos(double a) { … }
    public static double tan(double a) { … }
    public static int abs(int a) { … }
    …

}
```


Vissa klasser är bara rena metodsamlingar t.ex. den färdig Java klassen Math.

- Man skall inte kunna skapa några Math-objekt …
- … därför görs konstruktorn private. D.v.s. ingen utanför klassen kommer åt konstruktorn
  - Inga objekt kan skapas
- Alla metoder måste vara static.