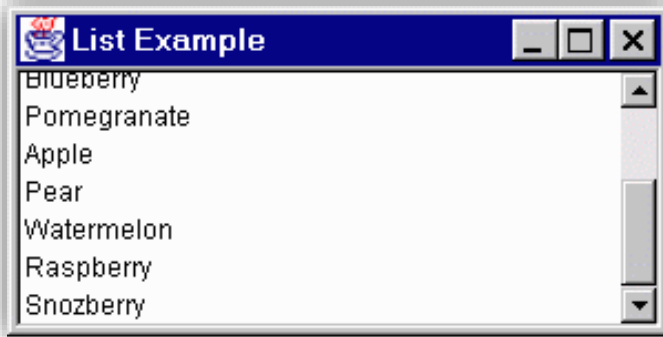


# Objektorientering och namnrymder

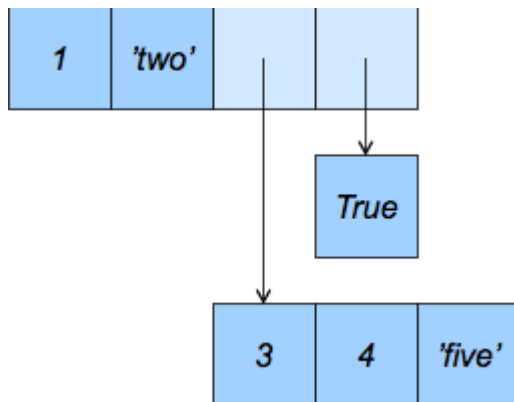
När ett namn har många betydelser

# Unika klassnamn? (1)

- Klassnamn bör vara lättförståeliga, enkla och entydiga
  - Java 1.0 införde en klass för grafiska list-komponenter: **List**



- Java 1.2 utökade stödet för listor som sekvenser.  
Den nya typen *borde* heta **List**...



**Hur gör vi nu?**

**Döper om till  
GuiList, ListStructure?**

**Då måste vi alltid använda långa namn**

# Unika klassnamn? (2)



- Min nya applikation har en egen implementation av listor
  - **JonasListStructure**? Tänk om det finns andra Jonas?

Eller kan det finnas flera klasser som heter List, i samma program?

```
def circle_area(radius):  
    result = 3.14159;  
    r2 = square(radius)  
    result * = r2;  
    return result
```

```
def square(x):  
    result = x * x  
    return result
```

Här är två olika variabler med samma namn...

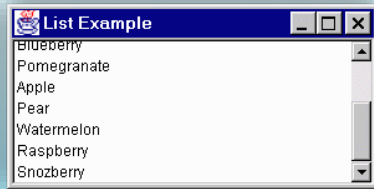
Samma namn kan betyda olika saker i olika sammanhang...  
Betydelsen beror på namnrymden

# Unika klassnamn? (3)

Även för klasser kan man ofta skapa distinkta namnrymder

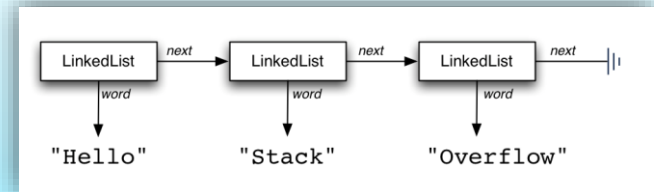
## “GUI-namnrymd”

List →



## “Datatyps-namnrymd”

List →



Enklare namngivning: Vid behov kan vi skilja på `gui.List`, `datatypes.List`

Organisation via gruppering: `datatypes.List` och `datatypes.Set` hör ihop!

- I Java ingår varje klass i ett paket (som är en namnrymd)

```
package se.liu.ida.jonkv.graphics.shapes;  
  
public class Circle {  
    ...  
}
```

Den deklarerar vilket paket innehållet tillhör

Standard för unika namn:  
Börja med domännamn,  
baklänges

- Varje klass har ett fullständigt namn: paketnamn.klassnamn
  - Detta namn räcker för att hitta klassen eller dess källkod

```
package se.liu.ida.something.else;  
public class Painter {  
    private se.liu.ida.jonkv.graphics.shapes.Circle c1;  
}
```

I Python hade en import krävts...

Hur hittar Java klassen? Måste vi skriva så här mycket?

- Källkod lagras i en katalogstruktur motsvarande paketen

- `se.liu.ida.jonkv.graphics.shapes.Circle` →

- **rotkatalog** `~/mysource`

- katalog `se`

- katalog `liu`

- katalog `ida`

- katalog `jonkv`

- katalog `graphics`

- katalog `shapes`

- fil `Circle.java`

- fil `GraphicCircle.java`

**Kan bli många nivåer**

**Utvecklingsmiljöer  
hjälp till att  
hålla reda på detta...**

- Ange rotkatalogen `~/mysource` – så hittas allt annat från den

- På kommandoraden: Miljövariabeln **CLASSPATH**

- I övrigt: Beror på utvecklingsmiljön

# Paket 3: Existerande



- Många paket finns i Javas klassbibliotek

## **java.lang:**

Object, Class, String,  
Thread, ...

## **java.util:**

List, ArrayList, Date,  
Calendar, Timer,  
Formatter, ...

## **java.io:**

Reader, InputStream,  
...

# Paket 4: Import förkortar namnen



- Men nu blir det väl lite jobbigt...
  - `javax.swing.filechooser.FileFilter filter = new javax.swing.filechooser.FileFilter();`

- Undvik med **Javas** variant av **import**

- **package** se.liu.ida.jonkv.io;  
**import** javax.swing.filechooser.FileFilter;

```
public class FileSaver {  
    ...  
    FileFilter filter = new FileFilter();  
    ...  
}
```

Talar bara om var FileFilter finns!

Om jag säger FileFilter  
menar jag  
javax.swing.filechooser.FileFilter

Ungefärlig motsvarighet i Python:

```
from javax.swing.filechooser  
import FileFilter
```

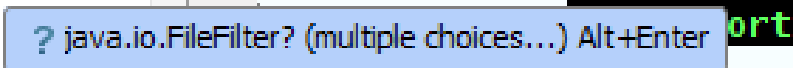
Men: Klassen har samma paket  
oberoende av om man importerar



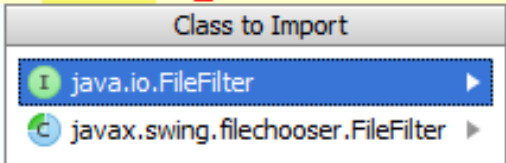
# Paket 5: Automatisk import



- Vissa klasser importeras automatiskt
  - Alla klasser i **samma paket**
  - Alla klasser i **java.lang** ("systemklasser" som **Object**, **String**)
- Moderna utvecklingsmiljöer hjälper till med annan import



```
    {  
        FileFilter filter = _  
    }
```



```
    FileFilter filter = _  
}
```

# Paket 6: Import med "\*"



- Kan importera alla klasser i ett paket med "\*"
  - **import** javax.swing.filechooser.\*;
FileFilter filter = **new** FileFilter();

Motsvarighet i Python:

```
from javax.swing.filechooser
import *
```

Undvik: Kan ge problem om nya klasser adderas till ett importerat paket...

## Skriv i Java 1.0...

```
import gui.*; // Har en List-klass
import java.util.*; // Ingen List-klass...
class MyClass {
    List list; // Unikt!
}
```

## Kompilera med Java 11...

```
import gui.*; // Har en List-klass
import java.util.*; // Har också en...
class MyClass {
    List list; // ??? Kompileringsfel!
}
```

- Om inget paket anges:
  - Klassen hamnar i "unnamed package"
  - **Använd inte detta** – ange alltid paketnamn!
- För att:
  - Undvika namnkollisioner
  - Förbättra läsbarhet, underhåll, organisation:  
Vilka klasser är **relaterade**?
  - Undvika problem med verktyg som antar att paketnamn finns

De flesta av projekten i kursen  
bör ha flera "delpaket"

# Paket 8: Sammanfattning och översikt



se.liu.ida.jonkv.util: Paket som innehåller klasser

List: Klass med fält, metoder, konstruktorer

size: Fält, antalet element

add: Metod

element: Variabel

get: Metod

element: Variabel

Set: Klass med fält, metoder, konstruktorer

size: Fält, antalet element

add: Metod

element: Variabel

get: Metod

element: Variabel

se.liu.ida.jonkv.gui

List: Klass med fält, metoder, konstruktorer