Kodkonvensioner och god programmeringsstil

Bakgrund och syfte

- Gör rätt från början
  - 80% av kostnaden för ett program är underhåll
  - *I stort sett inget program skrivs/underhålls av en författare*
  - ***Att kod är konsekvent skriven underlättar läsbarheten***

Kodkonvention

- Ett standardiserat sätt att skriva kod
- Det finns *fler* standarder
- Ofta tvingas man hålla sig till en standard som gäller på företaget
- Var konsekvent!

- Tänk noggrant på *struktur* och *namngivning*!
  (ett skrämmande exempel)

```
if (lortgrisar.size() > 0) {
    RT.updateNejbor();
    RT.doChimpans();
    RT.doGorilla()
}
```


- Suns kodkonvention:
  http://java.sun.com/docs/codeconv/

- Exempel på kommentarer:

```
/*
 * Here is a block comment.
 */
if (condition) {

    /* Handle the condition. */
    ...
}
if (a == 2) {
    return true;        /* special case */
} else {
    return isPrime(a);  /* works only for odd a */
}
```


if-satser

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```


Undvik onödig kod

```
if (booleanExpression) {
    return true;
} else {
    return false;
}
```

- ska skrivas som

```
return booleanExpression;
```

Klassnamn, gränssnitt, variabler och konstanter

```
class MyClass { ... }
interface Interfacable { ... }
int variable = 4;
double getDouble () { ... }
static final int MY_CONSTANT = 3;
```


Dokumentation

- Viktigt
- Viktigt
- Viktigt
- javadoc

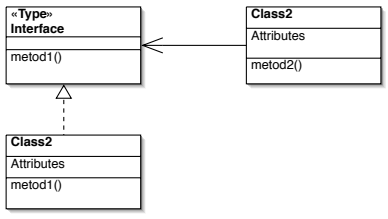

```
/**
 * Vilket pris har en viss vara
 *
 * @param id Den vara man vill veta priset på.
 *
 * @return Priset på varan
 *
 * @throws myshop.ItemNotFoundException Det fanns ingen vara med
 *                                      den givna identifikationen.
 */
public float getPrice( String id )
    throws ItemNotFoundException {
    // Leta reta på varan finns. Om varan inte finns så genereras ett
    // undantag och exekveringen av denna metod avslutas på en gång
    // samtidigt som felet skickas vidare uppåt i hierarkin.
    int pos = findStockItem(id);

    return storage[pos].price();
}
```


Programmera mot interface

Interface

- Snabba att implementera
- Bryter beroenden
- Gör det enklare att sammarbeta
- Gör det enkelt att i ett senare skede byta ut implementationer mot effektivare

Interface kan bryta beroenden



Skriv generell kod

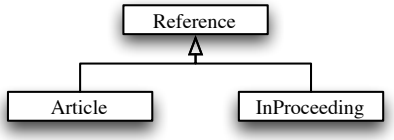
- Skriv alltid kod så att den använder sig av klasser/interface så högt upp i hierarkin som möjligt
- Ska ni bara göra saker som finns definierade i Figure-klassen så använd er då av en Figure-referens i stället för tex en mer specifik Triangle, så får ni mer generell kod

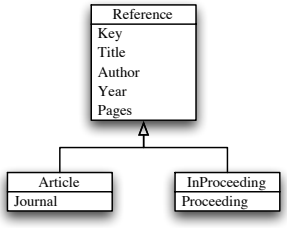
```
moveFigure(new Triangle())
void moveFigure(Figure f) {
   f.moveVertical(10);
   f.moveHorizontal(10)
}
```

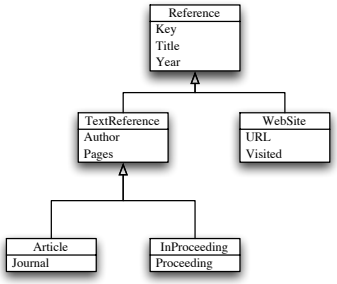

Ett större exempel

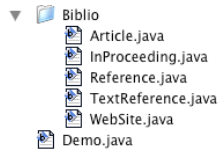
Programexempel

- Referenshantering
- Tidskriftsartiklar och konferensartiklar
  - Samma information till stor del









Reference

```
package Biblio;

abstract public class Reference
{
    private String refKey;
    private String refTitle;
    private String publishedYear;

    public Reference(String key)
    {
        refKey = key;
    }

    public String key( )
    {
        return refKey;
    }

    public void title( String newTitle )
    {
        refTitle = newTitle;
    }

    public String title( )
    {
        return refTitle;
    }

    public void year( String newYear )
    {
        publishedYear = newYear;
    }

    public String year( )
    {
        return publishedYear;
    }

    public String toString( )
    {
        return "Title = " + refTitle + "\nÅr = "
            + publishedYear + "\nKey = " + refKey;
    }
}
```


TextReference

```
package Biblio;

abstract public class TextReference
    extends Reference
{
    private String theAuthor;
    private int pageStart, pageEnd;

    public TextReference( String ID)
    {
        super(ID);
    }

    public void author( String newAuthor )
    {
        theAuthor = newAuthor;
    }

    public String author( )
    {
        return theAuthor;
    }

    public void pages( int start, int end )
    {
        pageStart = start;
        pageEnd   = end;
    }

    public int firstPage( )
    {
        return pageStart;
    }

    public int lastPage( )
    {
        return pageEnd;
    }

    public String toString( )
    {
        return super.toString() + "\nAuthor = " + theAuthor +
            "\nPage start = " + pageStart + "\nPage end = " + pageEnd;
    }
}
```


Article

```
package Biblio;

public class Article
    extends TextReference
{

    private String journalName;

    public Article( String ID )
    {
        super(ID);
    }

    public void journal( String theJournal )
    {
        journalName = theJournal;
    }

    public String journal( )
    {
        return journalName;
    }

    public String toString( )
    {
        return super.toString() + "\nJournal = " + journalName;
    }
}
```


Library

```
package Biblio;

public class Library
{
    final static int LIBRARY_SIZE = 100;

    private Reference ref[];
    private int refCount = 0;

    public Library( )
    {
        ref = new Reference[LIBRARY_SIZE];
    }

    public boolean add( Reference r )
    {
        for( int i = 0 ; i < refCount ; i++ ){
            if( ref[i].key() == r.key() ){
                return false;
            }
        }

        ref[refCount++] = r;
        return true;
    }

    public String toString()
    {
        String s="";

        for(int i = 0 ; i < refCount; i++){
            s = s + "Key: " + ref[i].key() +
                "    Titel: " + ref[i].title() + "\n";
        }

        return s;
    }
}
```

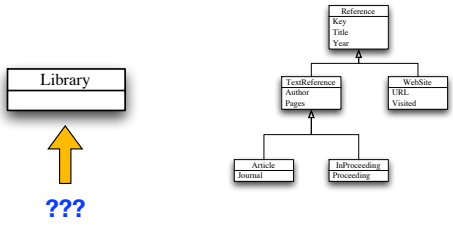
Inte bra


```
abstract public class Reference
{
    ...

    public boolean sameKey( Reference otherReference )
    {
        return refKey == otherReference.refKey;
    }

    ...
}
```

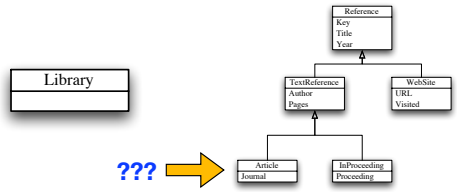
equals !!!

```
for( int i = 0 ; i < refCount ; i++ ){
    if( ref[i].sameKey(r) ){
        return false;
    }
}
```

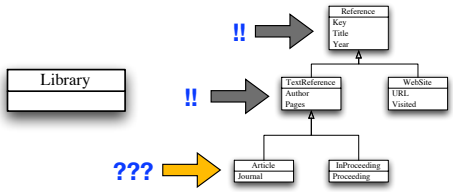

Export



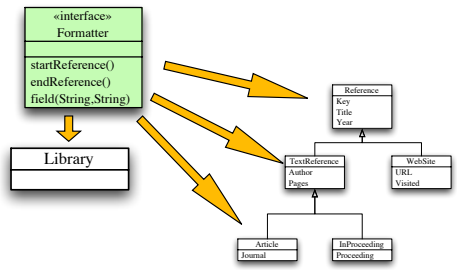
Export



Export



Export




```
package Biblio;

public interface Formatter
{

    public void startReference();

    public void endReference();

    public void field( String fld, String value );
}
```

```
public void export( Formatter format)
{
    for( int i = 0 ; i < refCount ; i++ ){
        format.startReference();
        ref[i].listFields(format);
        format.endReference();
    }
}
```


Article.listFields

```
public void listFields( Formatter format )
{
    format.field("JournalName",journalName);
    super.listFields(format);
}
```


TextReference.listFields

```
public void listFields( Formatter format )
{
    format.field("TheAuthor",theAuthor);
    format.field("PageStart",String.valueOf(pageStart));
    format.field("PageEnd",String.valueOf(pageEnd));
    super.listFields(format);
}
```



```
public class SimpleFormatter
    implements Formatter
{
    public void startReference( )
    {
        System.out.println(
            " ================= start ==============");
    }

    public void endReference( )
    {
        System.out.println(
            " ================= slut ==============\n\n\n");
    }

    public void field( String fld, String value )
    {
            System.out.println(fld + " : " + value);
    }

}
```



```
import Biblio.*;
import java.lang.System;

public class Demo
{
    public static void main( String[] args )
    {
        Article art = new Article("anka:2000");
        art.year("ca 2000");
        art.title("Vad kul detta är");
        art.journal("Disney's samlade verk");
        art.author("Kalle Anka");
        art.pages(34,90);

        InProceeding ip = new InProceeding("anka:2002");
        ip.year("2002");
        ip.title("Ankeborgs ekonomi");
        ip.author("Kalle Anka");
        ip.pages(12,78);
        ip.proceeding("Ankornas Världskonferens");

        Library lib = new Library();
        if( lib.add(art) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }
        if( lib.add(ip) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }

        System.out.println(lib);

        lib.export(new SimpleFormatter());
    }
}
```



```
import Biblio.*;
import java.lang.System;

public class Demo
{
    public static void main( String[] args )
    {
        Article art = new Article("anka:2000");
        art.year("ca 2000");
        art.title("Vad kul detta är");
        art.journal("Disney's samlade verk");
        art.author("Kalle Anka");
        art.pages(34,90);

        InProceeding ip = new InProceeding("anka:2002");
        ip.year("2002");
        ip.title("Ankeborgs ekonomi");
        ip.author("Kalle Anka");
        ip.pages(12,78);
        ip.proceeding("Ankornas Världskonferens");

        Library lib = new Library();
        if( lib.add(art) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }
        if( lib.add(ip) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }

        System.out.println(lib);

        lib.export(new SimpleFormatter());
    }
}
```

```
import Biblio.*;
import java.lang.System;

public class Demo
{
    public static void main( String[] args )
    {
        Article art = new Article("anka:2000");
        art.year("ca 2000");
        art.title("Vad kul detta är");
        art.journal("Disney's samlade verk");
        art.author("Kalle Anka");
        art.pages(34,90);

        InProceeding ip = new InProceeding("anka:2002");
        ip.year("2002");
        ip.title("Ankeborgs ekonomi");
        ip.author("Kalle Anka");
        ip.pages(12,78);
        ip.proceeding("Ankornas Världskonferens");

        Library lib = new Library();
        if( lib.add(art) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }
        if( lib.add(ip) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }

        System.out.println(lib);

        lib.export(new SimpleFormatter());
    }
}
```



```
import Biblio.*;
import java.lang.System;

public class Demo
{
    public static void main( String[] args )
    {
        Article art = new Article("anka:2000");
        art.year("ca 2000");
        art.title("Vad kul detta är");
        art.journal("Disney's samlade verk");
        art.author("Kalle Anka");
        art.pages(34,90);

        InProceeding ip = new InProceeding("anka:2002");
        ip.year("2002");
        ip.title("Ankeborgs ekonomi");
        ip.author("Kalle Anka");
        ip.pages(12,78);
        ip.proceeding("Ankornas Världskonferens");

        Library lib = new Library();
        if( lib.add(art) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }
        if( lib.add(ip) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }

        System.out.println(lib);

        lib.export(new SimpleFormatter());
    }
}
```



```
import Biblio.*;
import java.lang.System;

public class Demo
{
    public static void main( String[] args )
    {
        Article art = new Article("anka:2000");
        art.year("ca 2000");
        art.title("Vad kul detta är");
        art.journal("Disney's samlade verk");
        art.author("Kalle Anka");
        art.pages(34,90);

        InProceeding ip = new InProceeding("anka:2002");
        ip.year("2002");
        ip.title("Ankeborgs ekonomi");
        ip.author("Kalle Anka");
        ip.pages(12,78);
        ip.proceeding("Ankornas Världskonferens");

        Library lib = new Library();
        if( lib.add(art) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }
        if( lib.add(ip) ){
            System.out.println("Kunde lägga till");
        } else {
            System.out.println("Kunde inte lägga till");
        }

        System.out.println(lib);

        lib.export(new SimpleFormatter());
    }
}
```


Generics och Collections

Generics

Ännu ett sätt att lösa ett gammalt problem:
skriva så lite kod som möjligt

Hur implementerar vi då egna generiska klasser/metoder?

- Vill egentligen ha:
  - Lista av heltal
  - Lista av strängar
  - Lista av ...

Återanvända algoritmer för olika datatyper

```
package lists;

public class GenericList<T>
{
    private GenericListItem<T> headItem = null;
    private GenericListItem<T> tailItem = null;

    public GenericList( )
    {

    }

    public void append( T o )
    {
        if( headItem == null ){
            headItem = new GenericListItem<T>( o );
            tailItem = headItem;
        } else {
            tailItem = tailItem.append( o );
        }
    }

    public T head( )
    {
        if( headItem == null ){
            return null;
        } else {
            T tmp = headItem.value();

            headItem = headItem.next();
            return tmp;
        }
    }
}
```



```
package lists;

class GenericListItem<T>
{
    private GenericListItem<T> nextItem  = null;
    private T itemValue = null;

    public GenericListItem ( T o )
    {
        itemValue = o;
    }

    public T value( )
    {
        return itemValue;
    }

    public GenericListItem<T> append( T o )
    {
        if( nextItem != null ){
            return nextItem.append( o );
        } else {
            nextItem = new GenericListItem<T>( o );
            return nextItem;
        }
    }

    public GenericListItem<T> next( )
    {
        return nextItem;
    }
}
```



```
ol.append( new Object() );
String s = sl.head();
```

```
import lists.GenericList;

public class Demo3
{
    public static void main( String argv[] )
    {
        GenericList<String> sl = new GenericList<String>();
        GenericList<Object> ol = sl;
    }
}
```

```
javac Demo3.java
Demo3.java:8: incompatible types
found   : lists.GenericList<java.lang.String>
required: lists.GenericList<java.lang.Object>
        GenericList<Object> ol = sl;
```


Men kod som fungerar på alla generiska klasser?

- Till exempel skriva ut alla saker i listan


```
import lists.List;

public class Demo4a
{
    public static void main( String argv[] )
    {
        List l = new List();

        l.append( "Hello" );
        l.append( "World" );
        l.append( new Integer( 1234 ) );

        printStuff( l );

    }

    public static void printStuff( List sl )
    {
        Object o = null;

        o = sl.head();
        while( o != null ){
            System.out.println( o );
            o = sl.head();
        }
    }
}
```



```
import lists.GenericList;

public class Demo4
{
    public static void main( String argv[] )
    {
        GenericList<String> l = new GenericList<String>();

        l.append( "Hello" );
        l.append( "World" );

        printStuff( l );

    }

    public st
    {
        Object

        o = s
        while
            Sy
            o
    }
}
```

```
Demo4.java:12:
    printStuff(lists.GenericList<java.lang.Object>)
    in Demo4 cannot be applied to
    (lists.GenericList<java.lang.String>)
        printStuff( l );
        ^
```

```
import lists.GenericList;

public class Demo4b
{
    public static void main( String argv[] )
    {
        GenericList<String> l = new GenericList<String>();

        l.append( "Hello" );
        l.append( "World" );

        printStuff( l );

    }

    public static void printStuff( GenericList<?> sl )
    {
        Object o = null;

        o = sl.head();
        while( o != null ){
            System.out.println( o );
            o = sl.head();
        }
    }
}
```



```
public static void doStuff( GenericList<?> sl )
{
    Object o = sl.head();
    sl.append( o );
}
```

```
public static void doStuff( GenericList<?> sl )
{
    String o = sl.head();
    sl.append( o );
}
```



```
import lists.GenericList;
import shapes.*;

public class Demo5
{
    public static void main( String argv[] )
    {
        GenericList<Circle>    cl = new GenericList<Circle>();
        GenericList<Rectangle> rl = new GenericList<Rectangle>();

        // Do something useful

        doUsefulShapeStuff( cl );
        doUsefulShapeStuff( rl );



    }

    public static void doUsefulShapeStuff( GenericList<?> sl )
    {
        // xxxxxxxx
    }
}
```



```
import lists.GenericList;
import shapes.*;

public class Demo5
{
    public static void main( String argv[] )
    {
        GenericList<Circle>    cl = new GenericList<Circle>();
        GenericList<Rectangle> rl = new GenericList<Rectangle>();

        // Do something useful

        doUsefulShapeStuff( cl );
        doUsefulShapeStuff( rl );

        GenericList<String>    sl = new GenericList<String>();
        doUsefulShapeStuff( sl );
    }

    public static void doUsefulShapeStuff( GenericList<?> sl )
    {
        // xxxxxxxx
    }
}
```



```
import lists.GenericList;
import shapes.*;

public class Demo5
{
    public static void main( String argv[] )
    {
        GenericList<Circle>    cl = new GenericList<Circle>();
        GenericList<Rectangle> rl = new GenericList<Rectangle>();

        // Do something useful

        doUsefulShapeStuff( cl );
        doUsefulShapeStuff( rl );

        GenericList<String>    sl = new GenericList<String>();
        doUsefulShapeStuff( sl );
    }

    public static void doUsefulShapeStuff( GenericList< ? extends Shape > sl )
    {
        // xxxxxxxx
    }
}
```


Generic arrays

- Java tillåter inte skapandet av arrayer av generisk typ
- Eg.:

```
public class Stack<E>{
    private E[] data =new E[10];
    ...
}
```

är ej tillåtet

- Lösningen: Skapa en array av objekts och typkonvertera denna

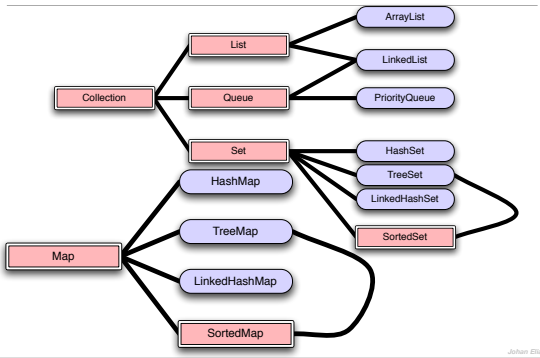
```
public class Stack<E>{
    private E[] data = (E[]) (new Object[10]);
    ...
}
```

- Fast det genererar en varning från kompilatorn :-(

Värt att veta

- Alla instanser av en generisk klass delar samma kod
- En del andra språk kompilerar upp en ny version av koden för varje instans av den generiska klassen (och ibland värre än så!!)

Collections




```
import java.util.HashMap;

public class Demo7
{
    public static void main( String[] argv )
    {
        HashMap<String, Integer> hi = new HashMap<String, Integer>();

        hi.put( "apa", new Integer( 12 ) );
        hi.put( "banan", new Integer( 23 ) );
        hi.put( "dask", new Integer( 239 ) );

        Integer i = hi.get( "banan" );

        System.out.println( i );
    }
}
```



```
import java.util.HashMap;

public class Demo7a
{
    public static void main( String[] argv )
    {
        HashMap<String, Integer> hi = new HashMap<String, Integer>();

        hi.put( "apa", 12 );
        hi.put( "banan", 23 );
        hi.put( "dask", 239 );

        int i = hi.get( "banan" );

        System.out.println( i );
    }
}
```

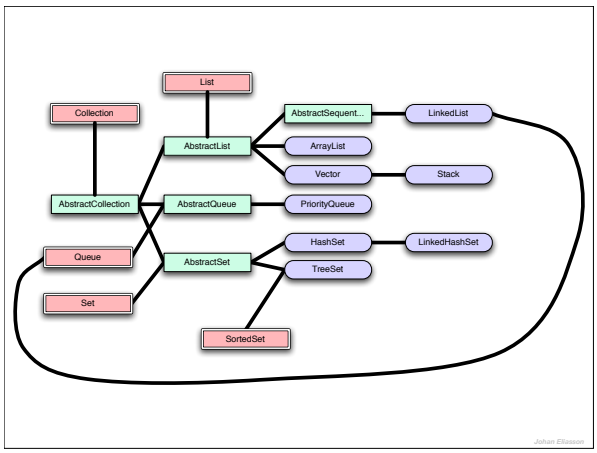
Autoboxing


```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;

public class Demo7b
{
    public static void main( String[] argv )
    {
        HashMap<String, Integer> hi = new HashMap<String, Integer>();

        hi.put( "apa", 12 );
        hi.put( "banan", 23 );
        hi.put( "dask", 239 );

        Set<String> keys;
        keys = hi.keySet();

        Iterator<String> it = keys.iterator();

        while( it.hasNext() ){
            System.out.println( it.next() );
        }


    }
}
```

Iteratorer istället för indexering

Nya for-loopen

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;

public class Demo7b
{
    public static void main( String[] argv )
    {
        HashMap<String, Integer> hi = new HashMap<String, Integer>();

        hi.put( "apa", 12 );
        hi.put( "banan", 23 );
        hi.put( "dask", 239 );

        Set<String> keys;
        keys = hi.keySet();

        Iterator<String> it = keys.iterator();

        while( it.hasNext() ){
            System.out.println( it.next() );
        }

        for( String thisKey : keys ){
            System.out.println( thisKey );
        }
    }
}
```


List
Collection
AbstractSequent...
LinkedList
AbstractList
ArrayList
Vector
Stack
AbstractCollection
AbstractQueue
PriorityQueue
HashSet
LinkedHashSet
Queue
AbstractSet
TreeSet
Set
SortedSet
Johan Ellsson