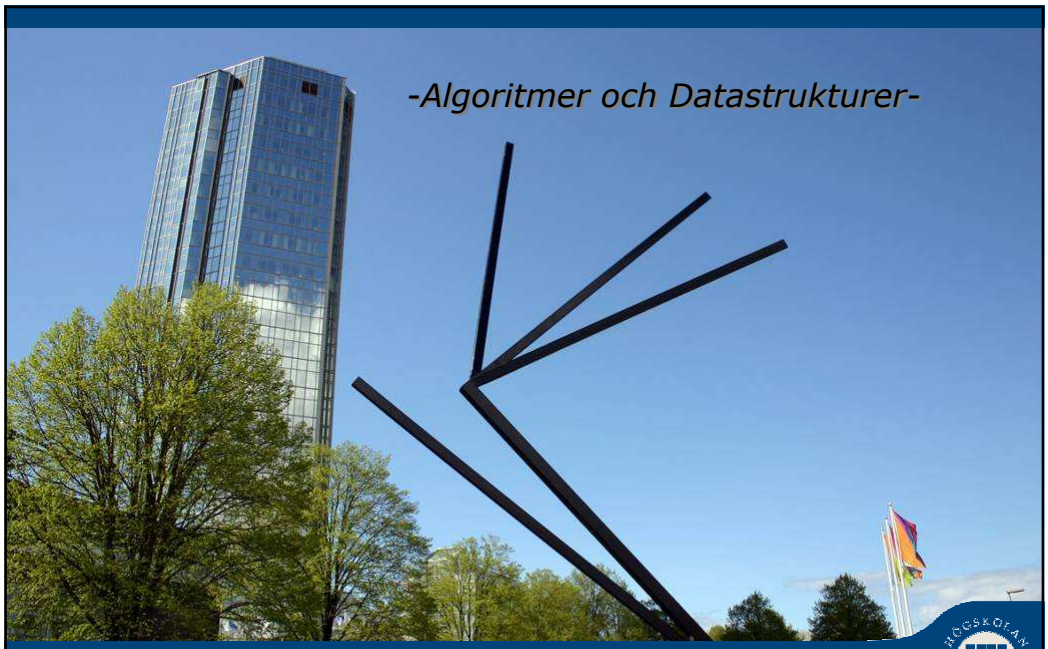


För utveckling av verksamhet, produkter och livskvalitet.

Abstrakt datatyp

- ***Datatyp*** för en variabel

Betecknar i ett programmeringsspråk den mängd värden variabeln får anta. T ex kan en variabel av typ boolean anta värdena true och false.

- ***Abstrakt datatyp (ADT)***

Abstrakt modell tillsammans med operationer definierade på modellen. Finns inte i verkligheten men är ”abstraherad ” från verkligheten.

- ***Datastruktur***

Används för att *representera* ADT. Är *samlingar av variabler av någon datatyp* som *hör ihop* på något sätt. Datastrukturer skapas genom att man ger namn åt (deklarerar) samlingar. Enklaste mekanismen för detta i Java är fält (array). ”Länkade” strukturer via referensvariabler är annan möjlighet.

- ***En del böcker använder beteckningen datastruktur som synonym med ADT. Däribland vår lärobok.***

List, Kö, Stack är exempel på ADT)

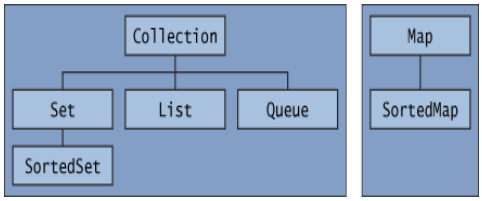
För utveckling av verksamhet, produkter och livskvalitet.

”Ording och reda” på datastrukturer i java med hjälp av olika interface

En datastruktur är:

- data (representerad på olika sätt)
- Metoder

- Finns i java.util http://java.sun.com/j2se/1.5.0/docs/api/



Collection- generell samling data
List- ordnad samlig av data
Set- ordnad samlig av data som inte får innehålla dubbleter
Map- Samling av data i par (objekt+ nyckel)
Queue- samlig data som accesar efter en bestämd ordning

Interface Collection

- Basen är ett interface Collection som representerar en samling element. Det innehåller följande metoder:

```
boolean isEmpty();
boolean add(E x);
boolean contains(Object x);
boolean remove(Object x);
void clear();
int size();
Iterator iterator();
```

En ”SimpleDataStructure”

```
public class SimpleDataStructure implements
      Collection
{
        private int theSize;
        private Object [ ] theItems;

 public void add ( Object obj)
{
   theItems[thesize++]=obj;
}
public boolean remove( Object obj ){

boolean found=false;
// hitta idx-index of objekt som skall bort
if( idx!=-1){
 faund=true;
for( int i = idx; i < size( ) - 1; i++ )
       theItems[ i ] = theItems[ i + 1 ]
}
theSize--;

return found;
}
```

```
public class Program
{
...main()....{
SimpleDataStructure bank=new SimpleDataStructure ();
bank.add( new Customer());
bank.add( new Customer());
.....
bank.add( new Donkey());




}
}
```

För utveckling av verksamhet, produkter och livskvalitet.

En ”SimpleDataStructure” generic

```
public class SimpleDataStructure <AnyType>
implements Collection <AnyType>
{
        private int theSize;
        private AnyType [ ] theItems;

 public void add ( AnyType obj)
{
   theItems[thesize++]=obj;
}

public boolean remove( AnyType obj){
      // idx- hitta index of objekt obj
.....
for( int i = idx; i < size( ) - 1; i++ )
       theItems[ i ] = theItems[ i + 1 ]

theSize--;

return found;
}

}
```

```
public class Program
{
...main()....{
SimpleDataStructure <Customer>bank=new
SimpleDataStructure <Customer>();
bank.add( new Customer());
bank.add( new Customer());
.....
bank.add( new Donkey()); //  compile error




}
}
```

För utveckling av verksamhet, produkter och livskvalitet.

Generiska klasser i Java 5.0

- Typparametrar anges inom tecknen < och > i klassrubriken:

public class ArrayList**<E>** { ... }

- Typparametern kan användas i klassen för att t ex deklarera attribut och parametrar och returtyper för metoder

```
public class ArrayList<E> {
private E[] elements;
public boolean add(E x) { … } // lägg in x sist
public E get(int i) { … } //hämta element på plats i
…
}
```

- Vi skapar instanser genom att ge parametern E ett värde
  ArrayList<Integer> intList = new ArrayList<Integer>();
  I intList kan endast Integer-objekt läggas in

Interface Comparable som ”generic”

- I java biblioteket finns interfacen Comparable som används för att definiera objekt som kan jämföras och rangordnas

```
public interface Comparable<AnyType>
{
  public int compareTo( AnyType other);
}
```

En klass som implementear en ”generic” interface

AnyType ersäts med Person

```
class Person implements Comparable<Person>
{

 // andra metoder  för person
public int compareTo( Person other)
{
 if( name.compareTo(other.getName())==0)  /jämför namn
   { if (adress .....)   // jämför adress


  }
else
  return  name.compareTo(other.getName())==0;
}
}
```

För utveckling av verksamhet, produkter och livskvalitet.

En ”generic” algoritm- findMax metoden

```
public static <AnyType extends Comparable<? super AnyType>>
   AnyType findMax ( AnyType [ ] a )

{
        int maxIndex = 0;

        for( int i = 1; i < a.length; i++ )
            if( a[ i ].compareTo( a[ maxIndex ] ) > 0 )
                maxIndex = i;

        return a[ maxIndex ];
}
```

För utveckling av verksamhet, produkter och livskvalitet.

Iteratorer

```
* Iterator interface for Collections.
*/
public interface Iterator
{
  * Tests if there are more items in the collection.
    * @return true if there are more items in the collection.
    boolean hasNext( );

    * Obtains the next item in the collection.
    * @return the next (as yet unseen) item in the collection.

     Object next( );

    * Remove the last item returned by next.
    * Can only be called once after next.
      void remove( );
}
```


Interfacet List

I java.util finns ett interface för listhantering som ärver interfacet Collection<E>:

```
public interface List<E> extends Collection<E> {

void add(int idx, E element);
boolean add(E element);
E get(int idx);
E set(int idx, E element);

ListIterator<E> listIterator();
…
…
}
```

ListIterator

ListIterator<E> är ett interface som ärver Iterator<E>
och där man lagt till metoder för att röra sig även bakåt i listor
samt för att sätta in och ta bort element:

public interface ListIterator<E> extends Iterator<E> {

boolean hasPrevious();
E previous();
void add(E x);
void remove();
…

}

List-klasser i java

- Det finns två konkreta generiska klasser i Javas API för listhantering. Båda implementerar alltså interfacet List
ArrayList som implementerats med vektor
LinkedList, som använder länkad struktur i implementationen

Man bör bara använda ArrayList om alla insättningar görs sist i listan. Insättningar i andra positioner orsakar flyttningar av element.
De indexerade operationerna **get(int idx)** och **set(int idx, E element)** är däremot effektivare i denna implementation än i LinkedList

Paketet weiss.nonstandard i kursboken

- Paketet weiss.nonstandard innehåller interface och klasser för samlingar, som *inte* direkt motsvarar klasser i java.util.

En del av klasserna utgör alternativa sätt att utforma vissa samlingar.

T ex finns det en **LinkedList** och en tillhörande iterator-klass som har ett annat gränssnitt än sin motsvarighet i java.util resp. weiss.util.
Det finns också klasser för hantering av prioritetsköer (**BinaryHeap** och PairingHeap) som inte har samma gränssnitt som javas prioritetsköklass.