En list-klass före generics

- En lista för String-objekt

```
public class ListOfStrings {
  private String[] theList;
  private int noOfElements;
  …
  public ListOfStrings() {
     theList = new String[10];
     noOfElements = 0;
  }
  public void addString(String s) { …
  public String getString(int index) { …
```

En list-klass före generics

- Supertypen Object, får referera till alla (sub-) typer

```
public class ListOfObjects {
  private Object[] theList;
  private int noOfElements;
  …
  public ListOfObjects() {
     theList = new Object[10];
     noOfElements = 0;
  }
  public void addObject(Object o) { …
  public Object getObject(int index) { …
```

En list-klass före generics

- En instans av listan

```
ListOfObjects list = new ListOfObjects();
list.addObject(new String("Forty-two"));
list.addObject(new Integer(42));

String s = (String) list.getObject(0);
Integer i = (Integer) list.getObject(1);
```

- Är det refererade objektet av rätt typ? Kontrolleras först run-time(!)

En list-klass *med* Generics

```
List<String> list = new List<String>();
list.add(new String("Forty-two"));
String s = list.get(0);
```

- <String> anger vilken typ listan ska instansieras för (typparameter)
- Kompilatorn kan kontrollera att rätt typ används. Följande ger kompileringsfel:
  `list.add(new Integer(42));`

Generics, syntax för klass

- T är en s.k. typparameter

```
public class DataSample<T> {

   private T theData;

   public DataSample(T theData) {
       this.theData = theData;
   }

   public T getData() {
       return theData;
   }

   public String toString() {
       return theData.toString();
   }
}
```

Generics, syntax för klass

- Instansiera klassen

```
DataSample<String> dss =
  new DataSample<String>("Forty-two");
String s = dss.getData();

DataSample<Integer> dsi =
  new DataSample<Integer>(42);
int i = dsi.getData();
```

Generics, syntax för metod

- Metoden

```
public static <T> T getMiddleElement(T[]
  arr) {
  return arr[arr.length/2];
}
```

- Anrop

```
String[] words = {"C#", "Android", "Java ME"};
  …
String middle =
  Generics.<String>getMiddleElement(words);
```

Generisk sortering m.h.a. Comparable

```
public static <T extends Comparable<T>> void sortObjects(T[] arr){
   int minindex;
   T temp;

   for(int i = 0; i < arr.length - 1; i++) {
        minindex = i;
        for(int j = i + 1; j < arr.length; j++) {
               if(arr[j].compareTo(arr[minindex]) < 0) { // compare
                      minindex = j;
               }
        }
        // Swap
        temp = arr[i];
        arr[i] = arr[minindex];
        arr[minindex] = temp;
   }
}
```

En generisk list-klass

```
public class List<T> {

    private T[] theList;
    private int noOfElements;

    public Queue(int size) {
      theList = (T[]) new Object[size];
      noOfElements = 0;
    }
    …
     public void add(T element) {
      if(noOfElements >= theList.length) {
            this.resize();
      }
      theList[noOfElements++] = element;
    }
    …
```