

Collections and Comparators

the Collection class


- manages Lists (ArrayLists and LinkedLists)
- offers a very useful method:

```
.sort( List );
```

but.... how can you “sort” a “list” of “objects”

well, sometimes it's easy

```
List myList = new ArrayList( );  
for (x=1000; x > 0; x = x-1)  
{  
  myList.add( x ); // 1000, 999, 998, 997, etc  
}  
Collections.sort( myList );  
Collections.sort( myList, Collections.reverseOrder() );
```


what's this?

that second parameter....

```
Collections.sort ( List );
```

```
Collections.sort ( Comparator );
```

the Comparator Interface

- allows you to design a custom comparison (like, comparing music records by song).

```
import java.util.Comparator;  
public class songComparator implements Comparator  
{  
    public int compare( Object obj1, Object obj2 )  
    {  
        musicRecord mr1, mr2;  
  
        mr1 = (musicRecord)obj1;  
        mr2 = (musicRecord)obj2;  
        return ( mr1.getSong().compareTo( mr2.getSong() ) );  
    }  
}
```

now, use it with any list

```
List musicDatabase = new ArrayList();  
musicDatabase.add( new musicRecord( ...  
musicDatabase.add( new musicRecord( ...  
musicDatabase.add( new musicRecord( ...  
musicDatabase.add( new musicRecord( ...  
  
songComparator sc = new songComparator();  
  
Collections.sort( musicDatabase, sc );
```
