- Universities maintain records of students. Each academic year adds new records to the collection, while the records of those who have left are moved to an archive collection. Listing subsets of the collection will be common: all the students taking first-year CS, or all the students due to graduate this year, for instance.

The number of items stored in a collection will vary from time to time. So far, we have not met any features of Java that would allow us to group together arbitrary numbers of items. We could, perhaps, define a class with a lot of individual fields to cover a fixed but very large number of items, but programs typically have a need for a more general solution than this provides. A proper solution would not require us either to know in advance how many items we wish to group together, or to fix an upper limit to that number.

So we will start our exploration of the Java library by looking at a class that provides the simplest possible way of grouping objects, an unsorted but ordered flexible-sized list: **`ArrayList`**. In the next few sections, we shall use the example of keeping track of a personal music collection to illustrate how we can group together an arbitrary number of objects in a single container object.

4.3 An organizer for music files

We are going to write a class that can help us organize our music files stored on a computer. Our class won't actually store the file details; instead, it will delegate that responsibility to the standard **`ArrayList`** library class, which will save us a lot of work. So, why do we need to write our own class at all? An important point to bear in mind when dealing with library classes is that they have not been written for any particular application scenario—they are general-purpose classes. One **`ArrayList`** might store student-record objects, while another stores event reminders. This means that it is the classes that we write for using the library classes that provide the scenario-specific operations, such as the fact that we are dealing with music files, or playing a file that is stored in the collection.

For the sake of simplicity, the first version of this project will simply work with the file names of individual music tracks. There will be no separate details of title, artist, playing time, etc. That means we will just be asking the **`ArrayList`** to store **`String`** objects representing the file names. Keeping things simple at this stage will help to avoid obscuring the key concepts we are trying to illustrate, which are the creation and usage of a collection object. Later in the chapter, we will add further sophistication to make a more viable music organizer and player.

We will assume that each music file represents a single music track. The example files we have provided with the project have both the artist's name and the track's title embedded in the file name, and we will use this feature later. For the time being, here are the basic operations we will have in the initial version of our organizer:

- It allows tracks to be added to the collection.
- It has no predetermined limit on the number of tracks it can store, aside from the memory limit of the machine on which it is run.
- It will tell us how many tracks are in the collection.
- It will list all the tracks.