Iterator Pattern

Iterator Pattern adalah sebuah pattern yang memungkinkan kita untuk mengakses sebuah *collection* tanpa kita perlu mengetahui detil implementasi dari *collection* tersebut. *Collection* adalah sebuah objek yang berfungsi untuk menyimpan kumpulan objek-objek. Sebuah *collection* dapat menyimpan objek-objek tersebut dalam sebuah array, linked list, maupun struktur data lainnya. Sebuah "Iterator" adalah sebuah objek yang memungkinkan kita untuk mendapatkan seluruh isi dari sebuah collection dengan cara membacanya satu demi satu. Tujuan penggunaan Iterator pattern adalah agar program kita tidak perlu dimodifikasi bila /ketika sumber data kita (collection yang kita gunakan) berubah. Penjelasan lengkap dari pattern ini dapat dilihat di slide kuliah.

Java memiliki dua buah interface yang digunakan untuk membuat pattern ini yaitu :

- Interface Iterable : kelas yang mengimplementasikan interface Iterable adalah kelas yang memiliki sebuah iterator.
- Interface Iterator : kelas yang mengimplementasikan interface Iterator adalah sebuah iterator untuk sebuah kelas *collection*.

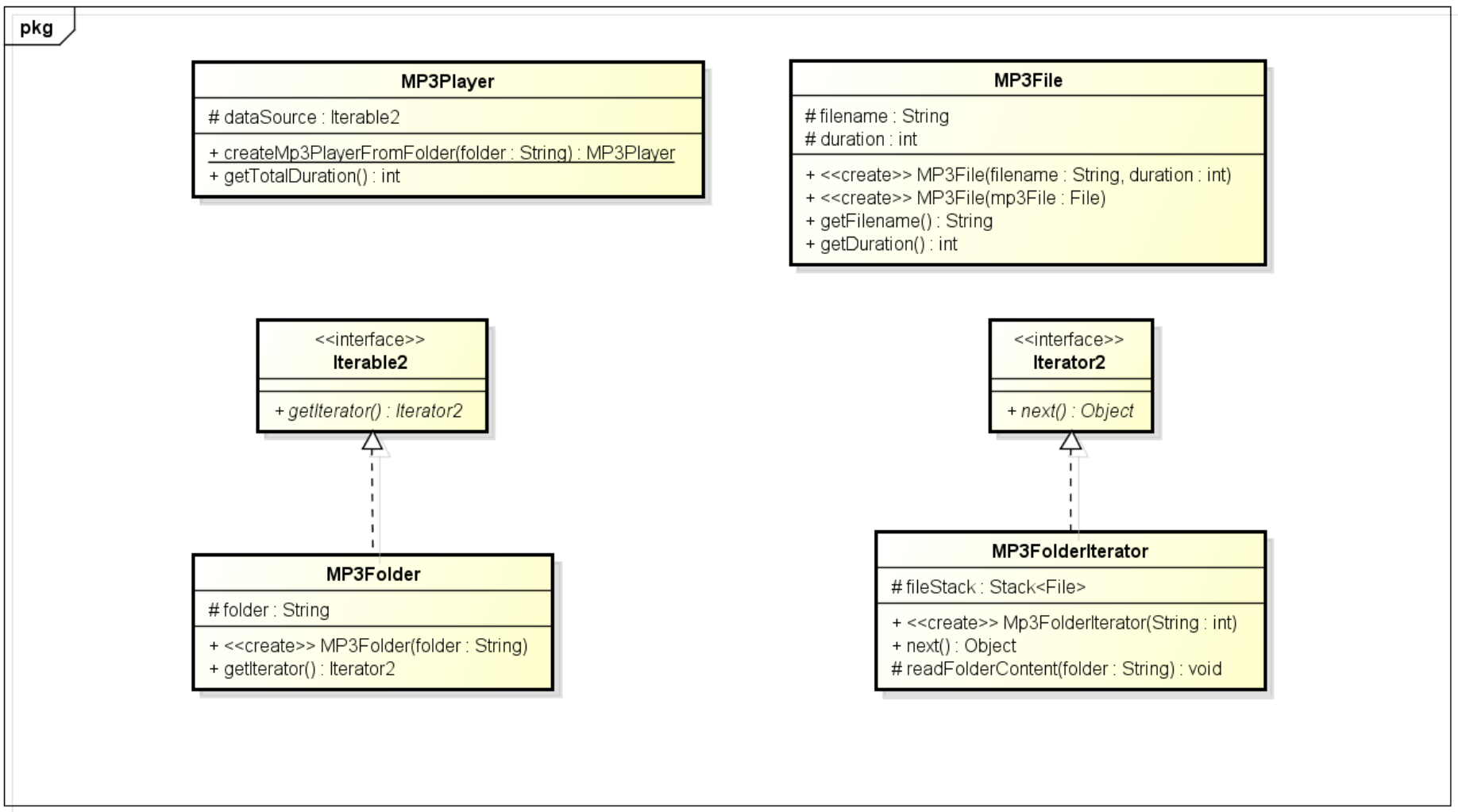
Pada modul kali ini kita akan membuat versi sederhana dari kedua interface diatas dengan nama Iterable2 dan Iterator2

- Iterable2 : interface ini akan memiliki sebuah method dengan nama getIterator yang akan mengembalikan iterator dari kelas ini. Iterator dari kelas ini akan mengimplementasikan interface Iterator2.
- Iterator2 : interface ini akan memiliki sebuah method dengan nama next(). Method ini akan mengembalikan satu buah objek yang disimpan oleh collection yang di-*iterate* oleh objek ini. Bila seluruh objek pada collection tersebut telah dibaca maka method ini akan mengembalikan nilai null.

MP3 Player

1. Menggunakan Sebuah Iterator

Untuk menunjukkan keuntungan dari penggunaan pattern Iterator, kita akan menggunakan kasus MP3 Player. Sebuah MP3 player memiliki daftar lagu-lagu yang akan dia mainkan. Pada bagian ini, MP3 Player milik kita akan memainkan seluruh lagu yang ada di dalam sebuah folder (dan subfolder-subfolder di bawahnya). Berikut ini adalah desain kelas dari bagian ini :



Method yang ditulis dengan garis bawah adalah method static!

Pada bagian ini anda akan diberikan 4 buah kelas dan 2 buah interface. **Kelas yang perlu anda lengkapi pada bagian ini hanyalah kelas MP3Player!**

1. Iterable2
   Iterable2 adalah sebuah interface yang memastikan bahwa kelas yang mengimplementasikan interface ini dapat menghasilkan sebuah objek iterator. Objek iterator yang dihasilkan akan bertipe Iterator2. Iterable2 adalah versi sederhana dari interface java.util.Iterable milik Java.
2. MP3Folder
   MP3Folder adalah sebuah kelas yang merepresentasikan sebuah folder di komputer anda yang berisi file-file MP3 yang ingin dimainkan. Sesuai dengan prinsip OO (dan pattern Iterator), anda **tidak perlu mengetahui cara kerja kelas MP3Folder dan iteratornya**. Yang perlu anda ketahui hanyalah bahwa MP3Folder adalah sebuah collection dan kita dapat mendapatkan iterator dari kelas MP3Folder dengan cara memanggil method getIterator(). **Penjelasan mengenai cara kerja kelas MP3Folder (dan iteratornya) dapat dibaca pada bagian akhir modul**.
3. Iterator2

Iterator2 adalah sebuah interface yang memastikan bahwa kelas yang mengimplementasikan interface ini dapat berfungsi sebagai iterator dari sebuah collection. Iterator2 adalah versi sederhana dari interface java.util.Iterator milik Java.

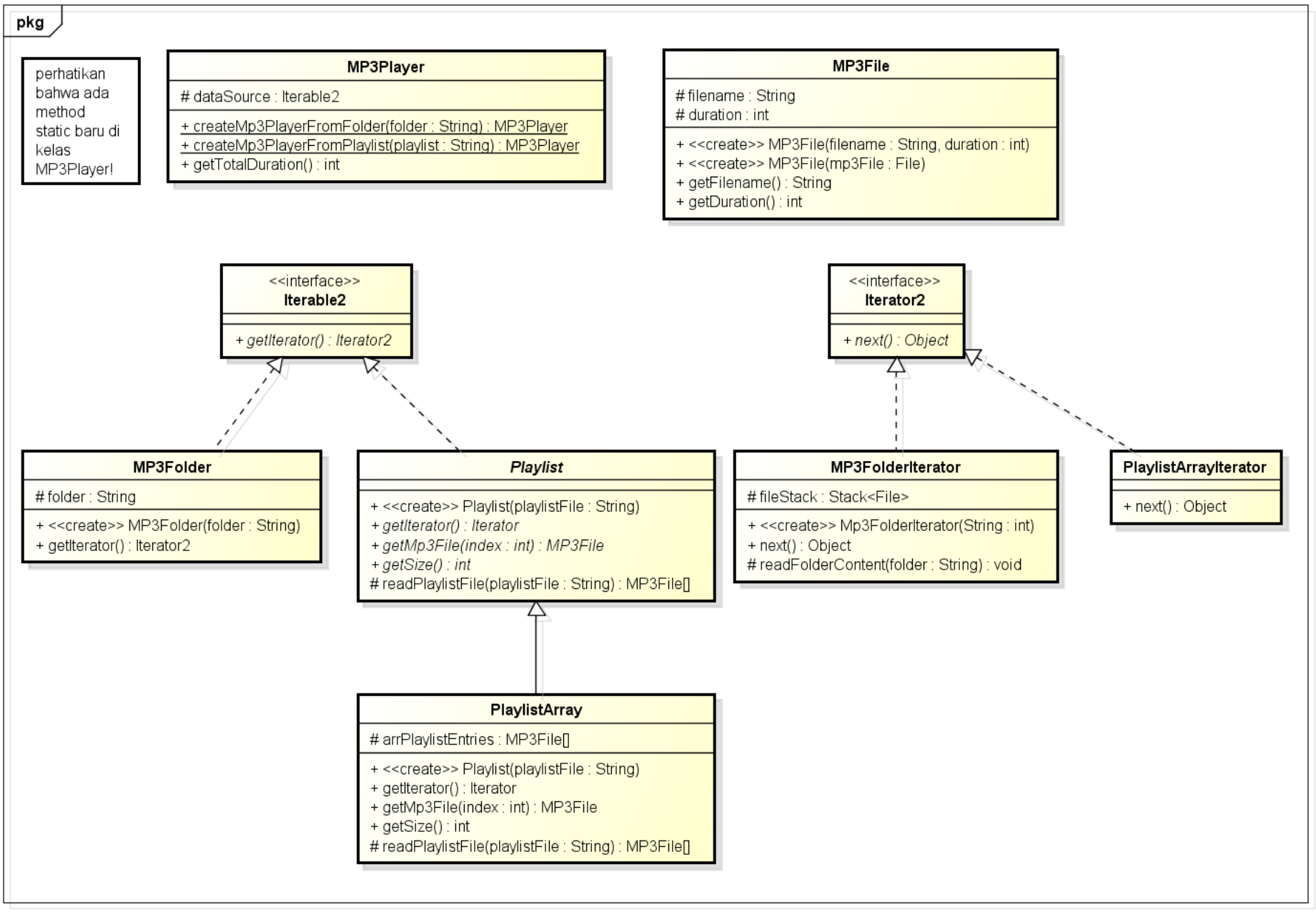
4. MP3FolderIterator
   Kelas ini adalah iterator dari kelas MP3Folder. Sama seperti pada kelas MP3Folder, sebagai pengguna dari iterator ini, anda tidak/belum perlu mengetahui detil implementasi dari kelas ini. **Method next dari kelas ini akan mengembalikan sebuah objek dari kelas MP3File**. Method ini akan mengembalikan nilai null bila semua objek milik *collection* ini sudah dibaca.
5. MP3File
   Kelas ini merepresentasikan sebuah file MP3. Kelas ini tidak dapat membaca sebuah file MP3 sesungguhnya. Kelas ini mengasumsikan bahwa isi file yang dibaca adalah sebuah bilangan bulat yang merupakan durasinya. Untuk lebih jelasnya bukalah salah satu file lagu pada folder contoh.
6. MP3Player
   Kelas ini merepresentasikan sebuah player Mp3. Alasan mengapa kelas ini tidak memiliki sebuah konstruktor dan dibuat dengan memanggil sebuah method static akan terlihat pada bagian berikutnya.

Tugas anda pada bagian ini adalah melengkapi isi method getTotalDuration.

Buatlah sebuah kelas Tester untuk menguji bagian ini. **Kumpulkanlah bagian ini sebagai R0901xxyyy.jar!**

2. Membuat Sebuah Objek Yang Iterable dan Iterator-nya

Pada bagian ini, kita akan menambahkan kemampuan dari MP3 player kita agar dia dapat membaca daftar lagu yang diinginkan dari sebuah file playlist. Desain kelas kita pada bagian ini adalah sebagai berikut :



Perbedaan-perbedaan pada bagian ini adalah :

1. MP3Player
   Kelas ini akan memiliki 1 method baru (createMp3PlayerFromPlaylist) yang berfungsi untuk mengembalikan sebuah objek MP3Player yang datasource-nya adalah sebuah playlist.
2. Playlist
   Kelas abstract ini merepresentasikan sebuah playlist. Kelas abstract ini tidak mendefinisikan struktur data yang digunakan untuk menyimpan playlist tersebut. Kelas ini memiliki sebuah method konkrit bernama readPlaylistFile yang berfungsi untuk membaca isi dari sebuah file playlist.
3. PlaylistArray
   Kelas ini adalah turunan konkrit dari kelas Playlist yang menyimpan daftar filenya dalam sebuah array. Iterator dari kelas ini adalah kelas PlaylistArrayIterator.
4. PlaylistIterator
   Kelas ini adalah iterator dari kelas PlaylistArray. Method next dari kelas ini akan mengembalikan sebuah objek dari kelas Mp3File. Method next akan mengembalikan nilai null bila semua objek dalam collection ini sudah dibaca.

Tugas anda pada bagian ini adalah :

- Buatlah kelas PlaylistArrayIterator. Anda boleh menambahkan method atau atribut lain yang anda butuhkan.
- Lengkapi method getIterator dari kelas PlaylistArray
- Tambahkanlah method
  `public int getJumlahFile()`
  pada kelas MP3Player. Method ini akan mengembalikan jumlah file pada datasource.

Ujilah pekerjaan anda dengan menggunakan kelas Tester anda! Kumpulkan bagian ini sebagai R0902xxyyy.jar

3. Review

Beberapa hal yang perlu diperhatikan pada modul ini adalah :

a. Efek iterator pattern terhadap coupling antar kelas
   Dengan mengabstraksikan kedua collection di atas ke kemampuan dasarnya (sebuah collection hanya dapat diambil isinya dengan next()), isi dari method-method seperti getTotalDurasi dan getJumlahFile **tidak perlu diubah** walaupun isi datasourcenya berubah**.** Hal ini disebabkan karena kelas MP3Player dan dataSource-nya yang bertipe Iterable tidak memiliki hubungan yang erat (tidak tightly coupled).

   Walaupun pada saat ini keuntungannya seakan-akan sangat kecil (hanya ada 2 method yang tidak perlu diubah vs ada beberapa interface dan abstract class yang perlu dibuat), keuntungan dari desain kelas yang tidak tightly coupled akan menjadi sangat besar pada saat kita mulai membuat program-program yang dibuat untuk digunakan di dunia nyata.

   Sebagai latihan, cobalah untuk membuat sebuah rancangan kelas yang **buruk** untuk kasus MP3Player ini lalu pikirkan kasus yang akan menyebabkan kelas-kelas yang sudah anda buat harus dimodifikasi secara luas.
b. Objek iterable tidak boleh sama dengan objek iterator-nya!
   Biasanya objek iterator harus merupakan objek terpisah dari collection yang diiterate. Sebagai contoh, pada bagian 2 iterable-nya adalah PlaylistArray dan iteratornya adalah PlaylistArrayIterator.

   Mengapa kita tidak menyatukan iterable dan iteratornya ? (contoh : public class PlaylistArray implements Iterable2,Iterator2) Alasannya adalah agar Iterator kita memenuhi poin ini :

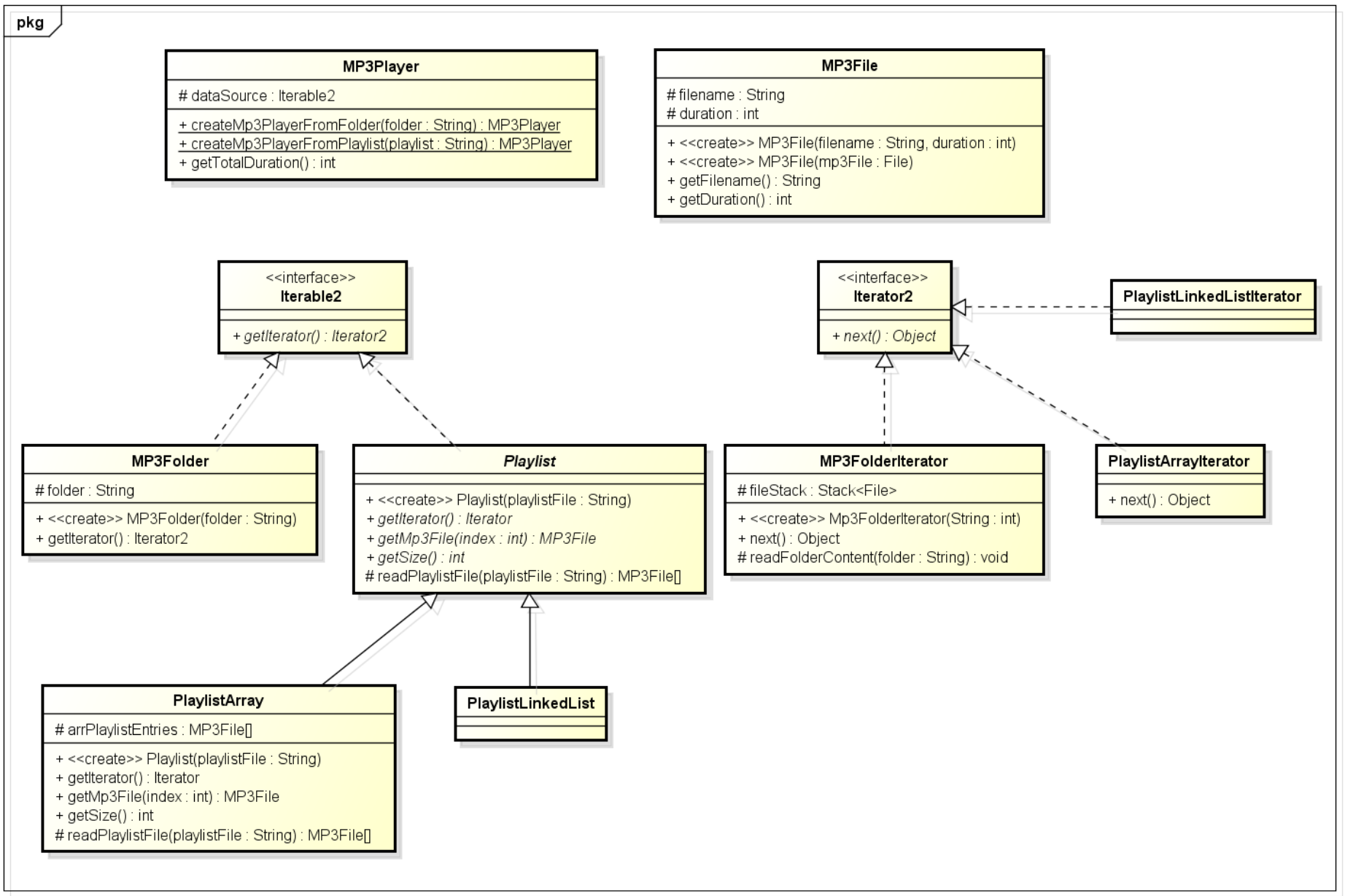
   `Support multiple traversal of a collection`

   Coba pikirkan alasan mengapa bila Iterable dan Iteratornya merupakan satu kelas yang sama maka poin di atas **dapat** terlanggar.
c. Iterator dan iterablenya biasanya bersifat tightly coupled
   Hal ini disebabkan karena biasanya iteratornya harus mengetahui detil implementasi objek yang dia iterate agar dia dapat melakukan tugasnya dengan efisien.
d. Penggunaan pattern iterator pada kasus dunia nyata
   Pattern iterator biasanya digunakan pada library-library yang digunakan untuk mengakses sebuah sumber data. Contohnya adalah library-library untuk mengakses sebuah DBMS seperti MySql,Oracle, dan lain-lain.

4. Latihan Tambahan

Bagian ini bersifat optional. Tugas anda pada bagian ini adalah menambahkan sebuah kelas yang berfungsi untuk menyimpan playlist dalam sebuah linked list dan membuat iteratornya. Usahakanlah agar running time dari operasi next() milik iteratornya adalah O(1). Untuk dapat mencapai hal ini kelas PlaylistLinkedList dan PlaylistLinkedListIterator hampir pasti akan bersifat tightly coupled. Agar lebih mudah, asumsikanlah bahwa linked list anda tidak memiliki operasi remove.

Diagram kelas dari bagian ini adalah sebagai berikut :



5. Keterangan Tambahan

Bagian ini sengaja diletakkan di akhir modul untuk mencegah terjadinya kasus dimana seorang mahasiswa membaca bagian ini lalu putus asa karena mengira bahwa Stack (dan rekursi) adalah hal yang bisa diignore setelah mereka melewati ASD. Mahasiswa diharapkan untuk ~~baru putus asa **setelah** kuliah berakhir~~ membaca kembali mengenai rekursi dan stack bila merasa perlu.

Kelas MP3Folder memiliki perbedaan yang mencolok dengan PlaylistArray dan PlaylistLinkedList. Walaupun ketiga kelas tersebut merupakan sebuah *collection*, namun kelas MP3Folder tidak menyimpan seluruh objeknya. MP3Folder baru akan membuat objek MP3File pada saat objek tersebut dibutuhkan. Hal ini berbeda dengan kedua kelas lainnya yang selalu menyimpan seluruh objek yang ada.

MP3FolderIterator merupakan contoh proses penelusuran pohon folder(definisi dari “pohon” ada pada kuliah Struktur Diskret). Sebuah pohon biasanya ditelusuri dengan menggunakan method yang bersifat rekursif. Namun pada MP3FolderIterator proses rekursif ini dilakukan dengan menggunakan Stack yang kita buat sendiri (method rekursi biasa akan menggunakan stack yang di-manage oleh JVM). Hal ini bertujuan agar kita dapat memenuhi kontrak dari interface Iterator2 yaitu “pemanggilan next() akan mengembalikan satu buah objek pada collection”.

Dengan memahami kelas MP3FolderIterator anda dapat menambahkan beberapa fitur menarik pada tugas besar anda seperti :

- Fitur untuk membaca level dari file terpisah
- Fitur dimana anda hanya perlu menambahkan file level ke suatu folder dan program anda akan secara otomatis memasukkan level itu ke dalam permainan. Dengan cara ini anda tidak perlu memodifikasi kode maupun meminta interaksi dari user pada saat anda membuat level baru.