Mehran Sahami
CS 106A

Handout #42
November 17, 2017

FlyTunes Program (Data Structures Example)

File: Song.java

```
/*
 * File: Song.java
 * ---------------
 * Keeps track of the information for one song
 * in the music shop, including its name, the band
 * that it is by, and its price.
 */

public class Song {

   /** Constructor
    *  Note that the song name and band name are immutable
    *  once the song is created.
    */
   public Song(String songName, String bandName, double songPrice) {
      title = songName;
      band = bandName;
      price = songPrice;
   }

   public String getSongName() {
      return title;
   }

   public String getBandName() {
      return band;
   }

   public void setPrice(double songPrice) {
      price = songPrice;
   }

   public double getPrice() {
      return price;
   }

   /** Returns a string representation of a song, listing
    *  the song name, the band name, and its price.
    */
   public String toString() {
      return ("\"" + title + "\" by " + band
            + " costs $" + price);
   }

   /* private instance variables */
   private String title;
   private String band;
   private double price;
}
```

File: Album.java

```
/*
 * File: Album.java
 * ----------------
 * Keeps track of all the information for one album
 * in the music shop, including the list of songs
 * it contains.
 */

import java.util.*;

public class Album {

   /** Constructor
    *  Note that the album name and year are immutable
    *  once the album is created.
    */
   public Album(String albumName, int year) {
      title = albumName;
      releaseYear = year;
   }

   public String getAlbumName() {
      return title;
   }

   public int getReleaseYear() {
      return releaseYear;
   }

   /** Adds a song to this album.  There is no duplicate
    *  checking for songs that are added.
    */
   public void addSong(Song song) {
      songs.add(song);
   }

   /** Returns an iterator over all the songs that are
    *  on this album.
    */
   public Iterator<Song> getSongs() {
      return songs.iterator();
   }

   /** Returns a string representation of an album.
    */
   public String toString() {
      return ("Album: [" + title + "] released in "
            + releaseYear);
   }

   /* private instance variables */
   private String title;
   private int releaseYear;
   private ArrayList<Song> songs = new ArrayList<Song>();
}
```

File: FlyTunesStore.java

```
/* File: FlyTunesStore.java
 * ------------------------
 * This program handles the data management for an on-line music store
 * where we manage an inventory of albums as well as individual songs.
 */

import acm.program.*;
import java.util.*;

public class FlyTunesStore extends ConsoleProgram {

    public void run() {
      while (true) {
         int selection = getSelection();
         if (selection == QUIT) break;
         switch (selection) {
            case LIST_SONGS:
               listSongs();
               break;
            case LIST_ALBUMS:
               listAlbums();
               break;
            case ADD_SONG:
               addSong();
               break;
            case ADD_ALBUM:
               addAlbum();
               break;
            case LIST_SONGS_ON_ALBUM:
               listSongsOnAlbum();
               break;
            case UPDATE_SONG_PRICE:
               updateSongPrice();
               break;
            default:
               println("Invalid selection");
               break;
         }
      }
    }

    /** Prompts the user to pick a selection from a menu
     *  of options.  Returns the users selection.  Note that
     *  there is no bounds checking done on the users selection.
     */
    private int getSelection() {
       println();
       println("Please make a selection (0 to quit):");
       println("1. List all songs");
       println("2. List all albums");
       println("3. Add a song");
       println("4. Add an album");
       println("5. List songs on an album");
       println("6. Update song price");
       int choice = readInt("Selection: ");
       return choice;
    }
```

```
 /** Lists all the songs carried by the store */
private void listSongs() {
   println("All songs carried by the store:");
   for(int i = 0; i < songs.size(); i++) {
      println(songs.get(i).toString());
   }
}

/** Lists all the albums carried by the store */
private void listAlbums() {
   println("All albums carried by the store:");
   Iterator<String> albumIt = albums.keySet().iterator();
   while (albumIt.hasNext()) {
      println(albums.get(albumIt.next()).toString());
   }
}

/** Checks to see if the song (defined by its name and
 *  the band that performs it) is already in the store.  It
 *  returns the index of the song in the store's song list
 *  if it already exists and -1 otherwise.
 */
private int findSong(String name, String band) {
   for(int i = 0; i < songs.size(); i++) {
      if (songs.get(i).getSongName().equals(name)
            && songs.get(i).getBandName().equals(band)) {
         return i;
      }
   }
   return -1;
}

/** Adds a new song to the store's inventory and returns that
 *  song to the caller.  If the song already exists in the
 *  store, it returns the existing song from the inventory.
 *  Otherwise it returns the new song that was just added to
 *  the inventory.  The method may return null if the user
 *  decides not to enter a song (i.e., user just presses
 *  Enter when asked for the song name).
 */
private Song addSong() {
   String name = readLine("Song name (Enter to quit): ");
   if (name.equals("")) return null;

   String band = readLine("Band name: ");
   int songIndex = findSong(name, band);
   if (songIndex != -1) {
      println("That song is already in the store.");
      return songs.get(songIndex);
   } else {
      double price = readDouble("Price: ");
      Song song = new Song(name, band, price);
      songs.add(song);
      println("New song added to the store.");
      return song;
   }
}
```

```
/** Adds a new album to the store's inventory.  If the album
 *  already exists in the store, then the inventory is
 *  unchanged.  Otherwise a new album and any new songs it
 *  contains are added to the store's inventory.
 */
private void addAlbum() {
   String name = readLine("Album name: ");

   if (albums.containsKey(name)) {
      println("That album is already in the store.");
   } else {
      int year = readInt("Release year: ");
      Album album = new Album(name, year);
      albums.put(name, album);

      while (true) {
         Song song = addSong();
         if (song == null) break;
         album.addSong(song);
      }
      println("New album added to the store.");
   }
}

/** Lists all the songs on a single album in the inventory. */
private void listSongsOnAlbum() {
   String name = readLine("Album name: ");
   if (albums.containsKey(name)) {
      Iterator<Song> it = albums.get(name).getSongs();
      println(name + " contains the following songs:");
      while (it.hasNext()) {
         Song song = it.next();
         println(song.toString());
      }
   } else {
      println("No album by that name in the store.");
   }
}

/** Updates the price of a song in the store's inventory.
 *  Note that this price update will also affect all albums
 *  that contain this song.
 */
private void updateSongPrice() {
   String name = readLine("Song name: ");
   String band = readLine("Band name: ");

   int songIndex = findSong(name, band);
   if (songIndex == -1) {
      println("That song is not in the store.");
   } else {
      double price = readDouble("New price: ");
      songs.get(songIndex).setPrice(price);
      println("Price for " + name + " updated.");
   }
}
```

```
   /* Constants */

   private static final int QUIT = 0;
   private static final int LIST_SONGS = 1;
   private static final int LIST_ALBUMS = 2;
   private static final int ADD_SONG = 3;
   private static final int ADD_ALBUM = 4;
   private static final int LIST_SONGS_ON_ALBUM = 5;
   private static final int UPDATE_SONG_PRICE = 6;


   /* private instance variables */

   // Inventory all the albums carried by the store
   private Map<String,Album> albums = new HashMap<String,Album>();

   // Inventory of all the songs carried by the store
   private ArrayList<Song> songs = new ArrayList<Song>();
}
```