Song not cached, fetching from server...

Name: Paint It Blue Album: Cowboy Jams

Artist: Charley Crockett

SongID: 1

Duration (s): 120

Name: Paint It Blue

Album: Cowboy Jams

Artist: Charley Crockett

SongID: 1

Duration (s): 120

Song not cached, fetching from server...

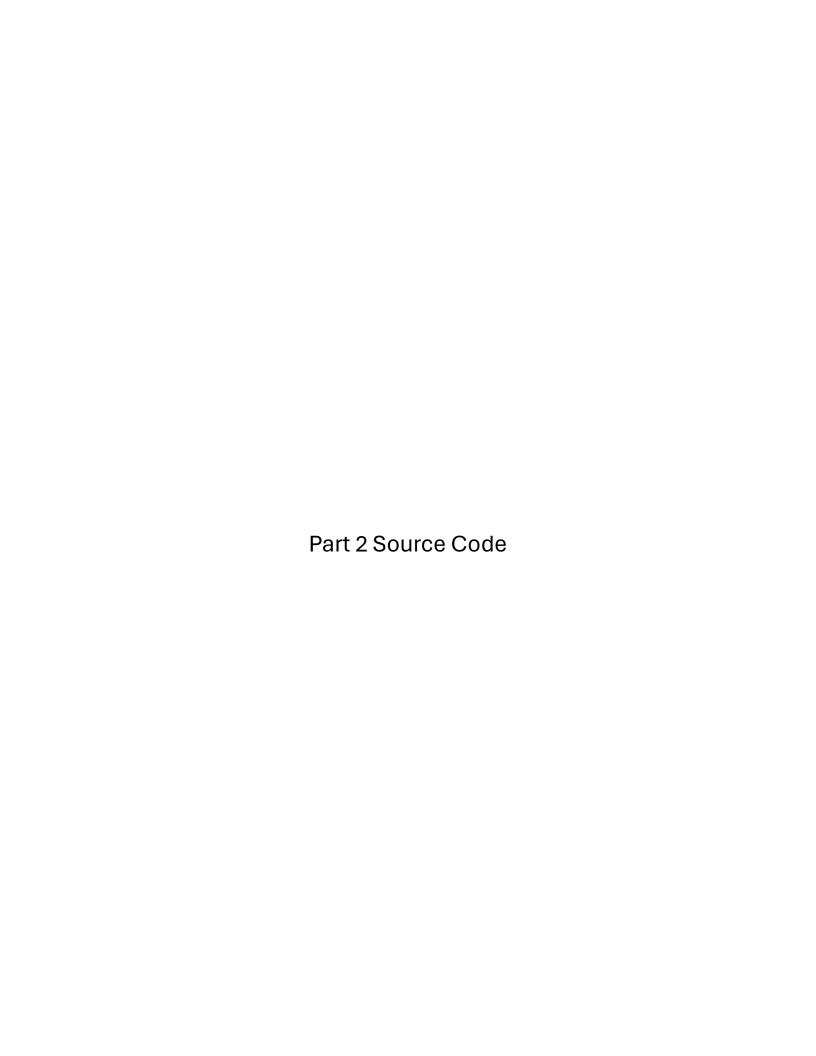
Name: Free Bird

Album: Patriotic Jams Artist: Lynrd Skynyrd

SongID: 6

Duration (s): 600

Process finished with exit code 0



```
import org.junit.Test;
1
2
3
    import java.util.LinkedList;
    import java.util.List;
4
 5
    import static org.junit.Assert.*;
 6
 7
    public class ServerTest {
8
9
10
        @Test
         public void addSong() {
11
             Server server = Server.getInstance();
12
             server.addSong("Title1", "Artist1", "Album1", 1, 180);
13
14
             Song searchedSong = server.searchById(1);
15
             assertNotNull(searchedSong);
16
             assertEquals("Title1", searchedSong.getTitle());
17
             assertEquals("Artist1", searchedSong.getArtist());
18
             assertEquals("Album1", searchedSong.getAlbum());
19
             assertEquals(180, searchedSong.getDuration());
20
21
             List<Song> songsByTitle = server.searchByTitle("Title1");
22
             assertNotNull(songsByTitle);
23
             assertEquals(1, songsByTitle.size());
24
             assertEquals("Title1", songsByTitle.get(0).getTitle());
25
26
             List<Song> songsByAlbum = server.searchByAlbum("Album1");
27
             assertNotNull(songsByAlbum);
28
             assertEquals(1, songsByAlbum.size());
29
             assertEquals("Album1", songsByAlbum.get(0).getAlbum());
30
         }
31
32
        @Test
33
         public void searchById() {
34
             Server server = Server.getInstance();
35
36
             Song testSong = new Song("Test Song", "Test Artist", "Test Album", -1, 120);
37
             server.addSong("Test Song", "Test Artist", "Test Album", -1, 120);
38
39
             Song result = server.searchById(-1);
40
41
             assertEquals(testSong.getTitle(), result.getTitle());
42
         }
43
44
        @Test
45
         public void searchByTitle() {
46
             Server server = Server.getInstance();
47
48
             List<Song> testSong = new LinkedList<Song>();
49
             testSong.addFirst(new Song("Test Song", "Test Artist", "Test Album", 1, 120));
50
             server.addSong("Test Song", "Test Artist", "Test Album", 1, 120);
51
52
             List<Song> result = server.searchByTitle("Test Song");
53
```

```
54
55
             assertEquals(testSong.getFirst().getTitle(), result.getFirst().getTitle());
56
        }
57
58
        @Test
59
        public void searchByAlbum() {
60
            Server server = Server.getInstance();
61
62
            List<Song> testSong = new LinkedList<Song>();
63
             testSong.addFirst(new Song("Test Song", "Test Artist", "Test Album", 1, 120));
64
             server.addSong("Test Song", "Test Artist", "Test Album", 1, 120);
65
66
             List<Song> result = server.searchByAlbum("Test Album");
67
68
             assertEquals(testSong.getFirst().getTitle(), result.getFirst().getTitle());
        }
70
71
        @Test
72
        public void getInstance() {
73
            Server instance1 = Server.getInstance();
74
            Server instance2 = Server.getInstance();
75
76
             assertSame(instance1, instance2);
77
78
             assertNotNull(instance1);
79
             assertNotNull(instance2);
80
        }
81
```

```
import java.util.List;

public interface SongService {
   Song searchById(Integer songID);
   List<Song> searchByTitle(String title);
   List<Song> searchByAlbum(String album);
}
```

```
public class Song {
1
         private final String title;
2
         private final String artist;
3
        private final String album;
4
         private final int id;
 5
        private final int duration;
6
7
        public Song(String givenTitle, String givenArtist, String givenAlbum, int givenID, int givenDuration
8
             this.title = givenTitle;
9
             this.artist = givenArtist;
10
             this.album = givenAlbum;
11
             this.id = givenID;
12
             this.duration = givenDuration;
13
14
         }
15
         public String getTitle() {
16
17
             return title;
         }
18
19
         public String getArtist() {
20
             return artist;
21
         }
22
23
         public String getAlbum() {
24
             return album;
25
26
27
28
         public int getId() {
             return id;
29
         }
30
31
         public int getDuration() {
32
             return duration;
33
        }
34
35
    }
36
```

```
import java.util.ArrayList;
1
    import java.util.HashMap;
2
    import java.util.LinkedList;
3
    import java.util.List;
4
 5
 6
    public class Server implements SongService {
         private static Server instance;
 7
         private static HashMap<String, List<Song>> titleHashmap;
8
         private static HashMap<String, List<Song>> albumHashmap;
9
         private static HashMap<String, Song> idHashmap;
10
11
         private final int waitTime = 3000;
12
13
         private Server() {
14
             titleHashmap = new HashMap<String, List<Song>>();
15
             albumHashmap = new HashMap<String, List<Song>>();
16
             idHashmap = new HashMap<String, Song>();
17
         }
18
19
         public void addSong(String title, String artist, String album, int id, int Duration) {
20
             if (titleHashmap == null) {
21
                 titleHashmap = new HashMap<String, List<Song>>();
22
             }
23
             if (albumHashmap == null) {
24
                 albumHashmap = new HashMap<String, List<Song>>();
25
26
             if (idHashmap == null) {
27
                 idHashmap = new HashMap<String, Song>();
28
             }
29
30
             Song songToAdd = new Song(title, artist, album, id, Duration);
31
32
             if (titleHashmap.get(title) == null) {
33
                 titleHashmap.put(title, new LinkedList<Song>());
34
35
             titleHashmap.get(title).addFirst(songToAdd);
36
37
             if (albumHashmap.get(album) == null) {
38
                 albumHashmap.put(album, new LinkedList<Song>());
39
             }
40
             albumHashmap.get(album).addFirst(songToAdd);
41
42
             if (idHashmap.get(Integer.toString(id)) == null) {
43
                 idHashmap.put(Integer.toString(id), songToAdd);
44
             }
45
         }
46
47
         public Song searchById(Integer songID) {
48
             try {
49
                 Thread.sleep(waitTime);
50
             } catch (Exception e) {
51
                 e.printStackTrace();
52
             }
53
```

```
54
55
             return idHashmap.get(Integer.toString(songID));
56
         }
57
58
        public List<Song> searchByTitle(String title) {
59
60
                 Thread.sleep(waitTime);
61
             } catch (Exception e) {
62
                 e.printStackTrace();
63
64
65
             return titleHashmap.get(title);
66
         }
67
68
         public List<Song> searchByAlbum(String album) {
            try {
70
                 Thread.sleep(waitTime);
71
             } catch (Exception e) {
72
                 e.printStackTrace();
73
             }
74
75
             return albumHashmap.get(album);
76
        }
77
78
        public static Server getInstance() {
79
             if (instance == null) {
                 instance = new Server();
81
82
83
             return instance;
84
         }
85
```

```
import java.util.HashMap;
1
    import java.util.LinkedList;
2
    import java.util.List;
3
4
    public class ProxyServer implements SongService {
5
        private static ProxyServer proxyServer;
6
        private static HashMap<String, List<Song>> titleHashmap;
7
        private static HashMap<String, List<Song>> albumHashmap;
8
        private static HashMap<String, Song> idHashmap;
9
        private Song songResult;
10
        private List<Song> listSongResult;
11
12
        private ProxyServer() {
13
             titleHashmap = new HashMap<String, List<Song>>();
14
             albumHashmap = new HashMap<String, List<Song>>();
15
             idHashmap = new HashMap<String, Song>();
16
        }
17
18
        public Song searchById(Integer songID) {
19
             this.songResult = idHashmap.get(Integer.toString(songID));
20
21
             if (songResult == null) {
22
                 System.out.println("Song not cached, fetching from server...");
23
                 Song retrievedSong = Server.getInstance().searchById(songID);
24
                 idHashmap.put(Integer.toString(songID), retrievedSong);
25
             }
26
27
             return idHashmap.get(Integer.toString(songID));
28
        }
29
30
        public List<Song> searchByTitle(String title) {
31
             this.listSongResult = titleHashmap.get(title);
32
33
             if (listSongResult == null) {
34
                 System.out.println("Song not cached, fetching from server...");
35
                 List<Song> retrievedSong = Server.getInstance().searchByTitle(title);
36
                 titleHashmap.put(title, retrievedSong);
37
             }
38
39
             return titleHashmap.get(title);
40
        }
41
42
        public List<Song> searchByAlbum(String album) {
43
             this.listSongResult = albumHashmap.get(album);
44
45
             if (listSongResult == null) {
46
                 System.out.println("Song not cached, fetching from server...");
47
                 List<Song> retrievedSong = Server.getInstance().searchByAlbum(album);
48
                 albumHashmap.put(album, retrievedSong);
49
50
51
             return albumHashmap.get(album);
52
        }
53
```

```
54
55
        public static ProxyServer getInstance() {
56
            if (proxyServer == null) {
57
                 proxyServer = new ProxyServer();
58
59
60
            return proxyServer;
61
        }
62
63
   | }
```

```
import java.sql.Driver;
1
    import java.util.List;
2
3
    public class DriverProgram {
4
        public static void printSongInfo(Song song) {
 5
             System.out.println();
 6
             System.out.printf("\nName: %s", song.getTitle());
 7
             System.out.printf("\nAlbum: %s", song.getAlbum());
8
             System.out.printf("\nArtist: %s", song.getArtist());
9
             System.out.printf("\nSongID: %d", song.getId());
10
             System.out.printf("\nDuration (s): %d\n\n", song.getDuration());
11
        }
12
13
        public static void printSongListInfo(List<Song> listOfSongs) {
14
             for (Song song : listOfSongs) {
15
                 printSongInfo(song);
16
17
        }
18
19
        public static void main(String[] args) {
20
             Server server = Server.getInstance();
21
             ProxyServer proxyServer = ProxyServer.getInstance();
22
23
             server.addSong("Paint It Blue", "Charley Crockett", "Cowboy Jams", 1, 120);
24
             server.addSong("Separate Ways", "Journey", "80s Hits", 2, 240);
25
             server.addSong("Life is a Highway", "Pixar", "Cars Movie OST", 3, 120);
26
             server.addSong("21", "Sam Hunt", "Sad Country", 4, 120);
27
             server.addSong("Paint It Black", "The Animals", "Vietnam War Music", 5, 240);
28
             server.addSong("Free Bird", "Lynrd Skynyrd", "Patriotic Jams", 6, 600);
29
30
             List<Song> returnedSongs = proxyServer.searchByTitle("Paint It Blue");
31
             // Slow Retrieval
32
             DriverProgram.printSongListInfo(returnedSongs);
33
34
             returnedSongs = proxyServer.searchByTitle("Paint It Blue");
35
             // Fast Retrieval (Cached)
36
             DriverProgram.printSongListInfo(returnedSongs);
37
38
39
             Song returnedSong = proxyServer.searchById(6);
             // Fast Retrieval (Cached)
40
             DriverProgram.printSongInfo(returnedSong);
41
42
        }
43
    }
44
```

```
import static org.junit.Assert.*;
1
    import java.util.*;
2
    public class ProxyServerTest {
3
4
        @org.junit.Test
 5
        public void searchById() {
 6
             ProxyServer proxyServer = ProxyServer.getInstance();
 7
             Server server = Server.getInstance();
8
9
             Song testSong = new Song("Test Song", "Test Artist", "Test Album", 1, 120);
10
             server.addSong("Test Song", "Test Artist", "Test Album", 1, 120);
11
12
             Song result = proxyServer.searchById(1);
13
14
             assertEquals(testSong.getTitle(), result.getTitle());
15
        }
16
17
        @org.junit.Test
18
        public void searchByTitle() {
19
             ProxyServer proxyServer = ProxyServer.getInstance();
20
             Server server = Server.getInstance();
21
22
23
             List<Song> testSong = new LinkedList<Song>();
24
             testSong.addFirst(new Song("Test Song", "Test Artist", "Test Album", 1, 120));
25
             server.addSong("Test Song", "Test Artist", "Test Album", 1, 120);
26
27
             List<Song> result = proxyServer.searchByTitle("Test Song");
28
29
             assertEquals(testSong.getFirst().getTitle(), result.getFirst().getTitle());
30
        }
31
32
        @org.junit.Test
33
        public void searchByAlbum() {
34
             ProxyServer proxyServer = ProxyServer.getInstance();
35
             Server server = Server.getInstance();
36
37
             List<Song> testSong = new LinkedList<Song>();
38
             testSong.addFirst(new Song("Test Song", "Test Artist", "Test Album", 1, 120));
39
             server.addSong("Test Song", "Test Artist", "Test Album", 1, 120);
40
41
             List<Song> result = proxyServer.searchByAlbum("Test Album");
42
43
             assertEquals(testSong.getFirst().getTitle(), result.getFirst().getTitle());
44
        }
45
46
        @org.junit.Test
47
        public void getInstance() {
48
             ProxyServer instance1 = ProxyServer.getInstance();
49
             ProxyServer instance2 = ProxyServer.getInstance();
50
51
             assertSame(instance1, instance2);
52
53
```

```
54
55
56
57
}
assertNotNull(instance1);
assertNotNull(instance2);

66
67
}
```

 \emph{PDF} document made with CodePrint using $\underline{\textit{Prism}}$

```
import org.junit.Test;
1
2
3
    import static org.junit.Assert.*;
4
    public class SongTest {
 5
6
        @Test
7
         public void getTitle() {
8
             String givenTitle = "Bohemian Rhapsody";
9
             String givenArtist = "Queen";
10
             String givenAlbum = "A Night at the Opera";
11
             int givenID = 1;
12
             int givenDuration = 355;
13
14
             Song song = new Song(givenTitle, givenArtist, givenAlbum, givenID, givenDuration);
15
16
             assertEquals(givenTitle, song.getTitle());
17
         }
18
19
        @Test
20
         public void getArtist() {
21
             String expectedArtist = "Ed Sheeran";
22
             Song song = new Song("Shape of You", expectedArtist, "÷", 1, 234);
23
24
             String actualArtist = song.getArtist();
25
26
             assertEquals(expectedArtist, actualArtist);
27
         }
28
29
        @Test
30
         public void getAlbum() {
31
             Song song = new Song("Title", "Artist", "Album", 1, 180);
32
             assertEquals("Album", song.getAlbum());
33
         }
34
35
        @Test
36
         public void getId() {
37
             String givenTitle = "Song Title";
38
             String givenArtist = "Artist Name";
39
             String givenAlbum = "Album Title";
40
             int givenID = 123;
41
             int givenDuration = 240;
42
43
             Song song = new Song(givenTitle, givenArtist, givenAlbum, givenID, givenDuration);
44
45
             assertEquals(givenID, song.getId());
46
         }
47
48
        @Test
49
         public void getDuration() {
50
             int expectedDuration = 240;
51
52
             Song song = new Song("Title", "Artist", "Album", 1, expectedDuration);
53
```

```
assertEquals(expectedDuration, song.getDuration());

solution

assertEquals(expectedDuration, song.getDuration());

}
```