DSA GROUP PROJECT

222098236 - CHRISTIAN NATHINGO

[COMPANY NAME] [Company address]

DSA Group Project

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STUDENT  NAME | SURNAME | STUDENT  NUMBER | STUDY  MODE |  |
| Martin | Kapolo | 222071060 | FULLTIME |  |
| Ndinomwaami HT | Shihepo | 222129026 | FULLTIME |  |
| Breznev K | Leonard | 222085509 | PARTTIME |  |
| Christian P | Nathingo | 222098236 | FULLTIME |  |
| Simeon H | Haindobo | 222052058 | FULLTIME |  |
| Erustrus | Bathromeus | 221136304 | FULLTIME |  |

DESCRIPTION

The goal of this project is to create a

music player algorithm, Play music player, search, repeat, add track, remove track, and stop the music are the components of this musical player application. We used the java programming language, the NetBeans platform, MYSQL for our database, and Visio for our flowcharts. The Project has 7 members.

Modules

|  |  |
| --- | --- |
| Module/menu | Description |
| Playlist | The list is to focus on Keywords that Spotify listeners might be searching for like genres, mood, artist names ,song  names etc. |
| Track list | A list of tracks appearing on an album EP,LP etc. |
| Repeat Function | Repeat a string a specified number of times. The string can be specified as a string literal a function returning a string or a  column reference |
| Add or remove Function | When a system delimiter is encountered the value of the extracted element is returned and the creation of a new function through the addition of two other  functions |

Pseudo music player initialize playlist initialize tracklist initialize repeat function initialize search function

initialize add/remove function

play music player

play tracks from playlist repeat tracks as necessary search for specific tracks

add/remove tracks as necessary exit

end

* [**Explanation for step 1**](https://www.chegg.com/homework-help/questions-and-answers/section-algorithms-representation-different-modules-functions-pseudocode-flowchart-75-mark-q103241396#answer-explanation-tabs-for-step-1_tabpanel_0)

please refer to solution in the step. algorithm will provided.

ALGORITHM BELOW:

function create\_playlist() tracks := 100

linked\_list := create\_new\_linked\_list() for i := 0 to tracks-1 do add\_track(linked\_list, i)

end for

return linked\_list end function

function play\_track(linked\_list, track\_number) current\_track := linked\_list.head

for i := 0 to track\_number-1 do current\_track := current\_track.next end for play\_audio(current\_track.data) end function

function add\_track(linked\_list, track\_number) new\_track := create\_new\_track(track\_number) if linked\_list.head = null then

linked\_list.head := new\_track else

current\_track := linked\_list.head while current\_track.next != null do current\_track := current\_track.next end while

current\_track.next := new\_track end if

end function

function remove\_track(linked\_list, track\_number) current\_track := linked\_list.head

previous\_track := null

for i := 0 to track\_number-1 do

previous\_track := current\_track current\_track := current\_track.next end for

if previous\_track = null then linked\_list.head := current\_track.next else

previous\_track.next := current\_track.next end if

end function

function search\_playlist(linked\_list, track\_number) current\_track := linked\_list.head

while current\_track != null do

if current\_track.data = track\_number then return true

end if

current\_track := current\_track.next end while

return false end function





add/remove tracks as necessary

search for specific track



start

initialize playlist

initialize track\_list

initialize repeat function

initialize search function

initialize add/remove function



play music player

play tracks player

repeat track as necessary

public class musicplayer {

public static void main(String[] args) {

}

public final class DoubleClass<Music> {

private Music data;

private DoubleClass<Music> next; private DoubleClass<Music> prev;

public DoubleClass(final DoubleClass<Music> prev, final Music data, final DoubleClass<Music> next)

{

this.data = data; this.next = next; this.prev = prev;

}

public DoubleClass(final Music data) { this(null, data, null);

}

public Music getData() { return data;

}

public DoubleClass<Music> getNext() {

return next;

}

public void setTheNextTrack(final DoubleClass<Music> next) { this.next = next;

}

public void setThePreviousTrack(final DoubleClass<Music> prev) { this.prev = prev;

}

public DoubleClass<Music> getPrev() { return prev;

}

public void addTracks(final Music data) { this.data = data;

}

}

}