המחלקה להנדסת
חשמל ואלקטרוניקה

המכללה האקדמית להנדסה בכרמיאל
בראודה

פרויקט סוף בקורס

OOP **31695** סמסטר א' תשפ"ג

נגן מוסיקה

תאריך הגשה: 1.3.2023

שם: ליאן כהן
ת.ז: 315476457
מייל: lian1996cohen@gmail.com

שם: טוהר וקנין
ת.ז: 314865031
מייל: toharvaknin@gmail.com

ציון:

הסבר כללי על הפרויקט:

בפרוייקט זה ממשו למעשה נגן מוסיקה אשר מבוסס על ממשק DOS ומוסגל לנגן שירים של קבצי mp3.
כעת נפרט על כל אחת מהמחלקות שהגדרנו בפרוייקט זה.

:UserInterface

תיאור כללי:

מחלקה זו היא למעשה "המנהל" של כל התוכנית ובה מוגדרים כל הקונטיינרים המרכזיים של התוכנית וגם עוד אובייקטים של מחלקות אחרות שממשנו שנפרט עליהם בהמשך.

:Data members

std::string m_libraryPafh - זהו משתנה שלתוכו נכניס את הנתיב לקובוץ הסיפריה שלנו שבו יש את כל השמות של השירים, הנתיב של כל שיר וכל המאפיינים של השיר (זמר,אלבום,ז'אנר וכ'ו...).

Search m_searchObj – הגדרת אובייקט למחלקת החיפושים שנרחיב עלייה בהמשך.

Library m_libraryObj – הגדרת אובייקט למחלקת הסיפריה שנרחיב עלייה בהמשך.

PlayListOps m_playListOps – הגדרת אובייקט למחלקת הפעולות על פלייליסט שנרחיב עלייה בהמשך.

std::map<std::string, Song> m_library – קוטניינר שבמפתח שלו (key) יש לנו את שם השיר ובערך שלו (value) יש לנו אובייקט למחלקת שיר (Song) שלמעשה אוגרת לנו את כל המידע והמאפיינים של השיר כמו שם זמר, אלבום, ז'אנר, הנתיב לשיר, מספר הפעמים שהשיר הושמע וכ'ו...
בחרנו בקונטיינר זה כיוון שהוא מונע כפילויות בערכי המפתח שלו כלומר לא יהיה מצב כזה שיהיו לנו שירים עם שמות זהים.
בנוסף קונטיינר זה הוא ממויין לפי ערכי המפתחות שלו (keys) לכן כאשר ננגן או נדפיס את השירים הם ינוגנו\יודפסו לפי סדר האלפבת כנידרש.
דבר אחרון, קונטיינר זה מאוד יעיל בחיפוש,השמה ומחיקה של אלמנטים ((Red-black-tree->log(n)).

std::map<std::string, PlayList*> m_ListOfPlayLists – קונטיינר שבמפתח שלו (key) יש לנו את שם הפלייליסט ובערך שלו (value) יש לנו מצביע למחלקת פלייליסט שהיא למעשה מחלקה אבסטרקטית - הפולימורפיזם שלנו "והילדים" שלה הם כל הפלייליסטים שיש לנו בנגן: פלייליסטים קבועים ופלייליסטים שהוספנו תוך כדי השימוש בנגן.
בחרנו בקונטיינר זה כיוון שהוא מונע כפילויות בערכי המפתח שלו, כלומר לא יהיה מצב כזה שיהיו לנו מספר רשימות השמעה עם שמות זהים.
בנוסף קונטיינר זה הוא ממויין לפי ערכי המפתחות שלו (keys) לכן שנדפיס למסך את רשימות ההשמעה הקיימות בנגן הם יוצגו בסדר אלפבתי כנידרש.
דבר אחרון, קונטיינר זה מאוד יעיל בחיפוש, בהכנסה ובמחיקה של אלמנטים. ((Red-black-tree->log(n)).

std::set<std::string> m_constPlayLists – קונטיינר המכיל את רשימת הפלייליסטים הקבועים שלא ניתן למחוק והם: Recent, DailyMix, MostPlayed

std::set<std::string> m_regularPlayListNames – קונטיינר המכיל את השמות של הפלייליסטים החדשים שייצרנו תוך כדי השימוש בנגן וזאת על מנת שבעת העלת הנתונים לקובץ (בסיום התוכנית) נוכל להכניס אליו את השמות של כל הפלייליסטים שהוספנו במהלך התוכנית (השימוש בנגן) ואז ככה למעשה הם יישמרו שנפעיל את התוכנית בפעם הבאה (נקרא אותם מהקובץ).

int m_numberOfRegularPlayLists – מספר הפלייליסטים החדשים שיצרנו בנגן תוך כדי השימוש בתוכנית. משתנה זה נועד כדי לדעת בכמה שורות בקובץ (שבו שומרים את האובייקטים של התוכנית) יש שמות של פלייליסטים שאנחנו יצרנו תוך כדי השימוש בתוכנית.

std::unordered_map<std::string, char> m_options - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט הראשי.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של התפריט הראשי במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

bool m_bufferFlag – דגל זה נועד על מנת לדלג על פונקצית ה- help (שמבוצעת אוטומטית אחרי כל פעולה) כאשר המשתמש בוחר בפעולת ה-help וזאת כדי לא להדפיס אותה פעמיים במסך ה-DOS.
בנוסף דגל זה מסייע לנו לזהות מתי בבאפר של ה- cin יש מחרוזת ריקה.

Function members:

void Play() – זאתי למעשה המתודה שאנו מתחילים איתה את התוכנית והיא למעשה אחראית להציג למשתמש את כל האפשריות שיש לרשותו בתפריט הראשי.
בנוסף היא אחראית לקרוא למתודות המתאימות בהתאם לבחירה של המשתמש.

void Help() const – מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט הראשי.

char CheckParams(std::string input) – מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות.

bool BufferHandling(std::string userInput) – פונקציה זו נועדה לטפל לנו במצב בו נשארת מחרוזת ריקה ב-cin לאחר שהמשתמש ביצע פעולה מסויימת.
מצב זה בעייתי כיוון שבאינטרציה הבאה למעשה המחרות הריקה שיש ב-cin "תישפך" לקלט של המשתמש ויווצר לנו קלט לא תקין ולא באשמת המשתמש.

PlayList:

תיאור כללי:

זוהי למעשה המחלקה האבסטרקטית שהגרנו בפרוייקט "והילדים" שלה הם למעשה כלל הפלייליסטים שיש לנו בנגן.
כל המתודות במחלקה זו הם pure virtual והם ממומשות אצל הילדים בלבד חוץ ממתודה סטטית אחת שנפרט עליה בהמשך.

Data members:

Player m_player – הגדרת אובייקט ממחלקת שחקן שניתנה לנו.
אובייקט זה למעשה מורש לכל "הילדים" של פלייליסט והוא אחראי על ניגון השירים.

Function members:

virtual void AddSong(std::string songName, Song songObj) = 0 – מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על הוספת שיר לפלייליסט מסויים.

virtual void DeleteSongBySongName(std::string songName) = 0 – מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על מחיקת שיר מרשימת השמעה מסויימת.

virtual void DeleteAllSongs() = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על מחיקת **כל** השירים מרשימת השמעה מסויימת.

virtual void Menu() = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על הצגת תפריט למשתמש בהתאם לרשימת ההשמעה שאליה הוא נכנס.

virtual void Play() = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על ניגון השירים של רשימת השמעה מסויימת זה אחר זה בדרך כלל לפי **סדר אלפבת** של השירים ברשימה.

virtual void PlayRandom() = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על ניגון השירים של רשימת השמעה מסויימת זה אחר זה **באופן רנדומלי**.

virtual void Print() = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על הדפסת רשימת ההשמעה למסך ה- DOS.

virtual void UpdateSongName(std::string songName, std::string UpdateName) = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על עידכון שם של שיר מסויים ברשימת ההשמעה.

virtual bool CheckIfSongExist(std::string songName) = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על בדיקה האם שיר מסויים קיים ברשימת ההשמעה.

virtual std::string GetPlaylistPath() const = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על החזרת הנתיב לרשימת השמעה מסויימת.

virtual Song GetSongObjectBySongName(std::string songName) = 0 - מתודה "וירטואלית טהורה" שבכל פעם תמומש אצל "הילד" המתאים בהתאם למי שהפלייליסט מצביע עליו והיא אחראית על החזרת האובייקט שיר (Song) המתאים לשיר שנשלח למתודה זו.

static char PlayMidMenu() – הגדרנו את המתודה הזו כמתודה סטטית כיוון שאנו צריכים את השימוש בה לא רק במחלקות היושרות מהמחלקה פלייליסט אלא גם במחלקת הסיפריה (Library) ובמחלקת הפעולות של הפלייליסט (PlayListOps).
בנוסף בגלל שהמחלקה פלייליסט היא מחלקה אבסטרקטית ולכן לא ניתן ליצור לה אובייקטים אנו הגדרנו מתודה זו כמתודה סטטית וכך למעשה אנו יכולים לקרוא לה בכל מקום בתוכנית ללא ייצירת אובייקט של פלייליסט בצורה הבאה: **PlayList::PlayMidMenu() .**
מתודה זו למעשה משמשת אותנו בכל פעם שאנו מנגנים רשימה של שירים התפקיד שלה הוא אחרי כל שיר לשאול את המשתמש אם הוא רוצה להמשיך לשיר הבא ברשימה או לעצור את השמעת הרשימה.

Podcasts:

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימת ההסכתים בנגן המוסיקה.

Data members:

std::set<std::string> m_podcasts – זהו למעשה הקונטיינר בו השתמשנו על מנת לאחסן את שמות ההסכתים שלנו.
השתמשנו בקונטיינר מסוג set בגלל שקונטיינר זה הוא ממויין ולכן יישמור לנו את שמות ההסכתים בסדר אלפבתי וכאשר ננגן אותם הם כבר ייתנגנו בסדר אלפביתי כנדרש.
בנוסף קונטיינר זה מאוד יעיל בחיפוש ((log(n).

std::map<std::string, Song>* m_library – מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::string m_podcastsPath – משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את שמות ההסכתים.

std::unordered_map<std::string, char> m_options - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של ההסכתים.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של תפירט ההסכתים במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

std::vector<std::string> m_selections – קונטיינר זה אחראי לאחסן לנו את הקלט של המשתמש לאחר שהופרד ע"י רווחים.
בחרנו בקונטיינר זה מכיוון שאנו צריכים גישה לכל איבר בוקטור ע"י אינדקס ספציפי.

bool m_bufferFlag - דגל זה נועד על מנת לדלג על פונקצית ה- help (שמבוצעת אוטומטית אחרי כל פעולה) כאשר המשתמש בוחר בפעולת ה-help וזאת כדי לא להדפיס אותה פעמיים במסך ה-DOS.
בנוסף דגל זה מסייע לנו לזהות מתי בבאפר של ה- cin יש מחרוזת ריקה.

Function members:

void Menu() – מתודה זו אחראית על הצגת תפריט ההסכתים למשתמש והפנייה למתודה המתאימה בהתאם לבחירה של המשתמש.

void AddSong(std::string podcastName, Song songObj) – מתודה זו אחראית על הוספת פודקאסט חדש לרשימת ההסכתים והיא תבצע את ההוספה רק אם שם הפודקאסט לא קיים כבר ברשימת ההסכתים.

void DeleteSongBySongName(std::string podcastName) – מתודה זו אחראית על מחיקת פודקאסט מרשימת ההסכתים, כמובן שהיא תתבצע רק אם השם של הפודקאסט שנשלח למתודה קיים ברשימת ההסכתים.

void DeleteAllSongs() – מתודה זו אחראית על מחיקת כל הפודקאסטים הקיימים ברשימת ההסכתים.

void UpdateSongName(std::string podcastName, std::string UpdateName) – מתודה זו אחראית על עדכון שם של פודקאסט מסויים, כמובן שהיא תבצע את הפעולה רק אם שם הפודקאסט שנשלח למתודה קיים ברשימת ההסכתים.

bool CheckIfSongExist(std::string podcastName) – מתודה זו אחראית על בדיקה האם פודקאסט קיים ברשימת ההסכתים לפי שם הפוקדאסט שנשלח למתודה.

void PlaySpecificPodcast(std::string podcastName) – מתודה זו אחראית על ניגון פודקאסט ספציפי מרשימת ההסכתים ע"פ שם הפודקאט שנשלח למתודה, כמובן שפעולה זו תתבצע רק בתנאי ששם הפודקאסט קיים ברשמית ההסכתים.

void Play() – מתודה זו אחראית לנגן את כל הפודקאסטים ברשמית ההסכתים לפי סדר האלפבת.
(נזכיר שהקונטיינר שבחרנו לאיחסון שמות הפודקאסטים כבר מסדר אותם לפי סדר האלפבת).

void PlayRandom() – מתודה זו אחראית לנגן את כל הפודקאסטים ברשימת ההסכתים באופן רנדומלי.

void Print() – מתודה זו אחראית על הצגת רשימת ההסכתים במסך ה-DOS.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ ששומר את רשמית ההסכתים.

Song GetSongObjectBySongName(std::string podcastName) – מתודה זו ממומשת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void Help() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט ההסכתים.

char CheckUserParams() - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות.

bool CheckPlayParams() – מתודה זו אחראית לבדוק שמספר הפרמטרים שהמשתמש הכניס תואם למספר הפרמטרים הנידרש לפעולת ה-Play.

bool CheckDeleteParams() – מתודה זו אחראית לבדוק שמספר הפרמטרים שהמשתמש הכניס תואם למספר הפרמטרים הנידרש לפעולת ה- Delete .

bool BufferHandling(std::string userInput) - פונקציה זו נועדה לטפל לנו במצב בו נשארת מחרוזת ריקה ב-cin לאחר שהמשתמש ביצע פעולה מסויימת.
מצב זה בעייתי כיוון שבאינטרציה הבאה למעשה המחרות הריקה שיש ב-cin "תישפך" לקלט של המשתמש ויווצר לנו קלט לא תקין ולא באשמת המשתמש.

Favorite:

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימת המועדפים בנגן המוסיקה.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::set<std::string> m_favorite - זהו למעשה הקונטיינר בו השתמשנו על מנת לאחסן את שמות השירים המועדפים שלנו.
השתמשנו בקונטיינר מסוג set בגלל שקונטיינר זה הוא ממויין ולכן יישמור לנו את שמות השירים בסדר אלפבתי וכאשר ננגן אותם הם כבר ייתנגנו בסדר אלפביתי כנדרש.
בנוסף קונטיינר זה מאוד יעיל בחיפוש ((Red-bleck-tree->log(n)).

std::string m_favoritePath - משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את השירים המועדפים.

Function members:

void AddSong(std::string songName, Song songObj) – מתודה זו אחראית על הוספת שיר חדש לרשימת השירים המועדפים והיא תבצע את ההוספה רק אם שם השיר שנשלח למתודה לא קיים כבר ברשימת המועדפים.

void DeleteSongBySongName(std::string songName) – מתודה זו אחראית על מחיקת שיר מרשימת המועדפים, כמובן שהיא תתבצע רק אם השם של השיר שנשלח למתודה קיים ברשימת המועדפים.

void DeleteAllSongs() – מתודה זו אחראית על מחיקת כל השירים הקיימים ברשימת המועדפים.

void UpdateSongName(std::string songName, std::string UpdateName) – מתודה זו אחראית על עדכון שם של שיר מסויים, כמובן שהיא תבצע את הפעולה רק אם שם השיר שנשלח למתודה קיים ברשימת המועדפים.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית על בדיקה האם שיר קיים ברשימת המועדפים לפי שם השיר שנשלח למתודה.

void Play() – מתודה זו אחראית לנגן את כל השירים ברשמית המועדפים לפי סדר האלפבת.
(נזכיר שהקונטיינר שבחרנו לאיחסון שמות השירים כבר מסדר אותם לפי סדר האלפבת).

void PlayRandom() – מתודה זו אחראית לנגן את כל השירים ברשימת המועדפים באופן רנדומלי.

void Print() – מתודה זו אחראית על הצגת רשימת המועדפים במסך ה-DOS.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ ששומר את רשמית המועדפים.

void Menu() וגם **Song GetSongObjectBySongName(std::string songName)** - אלו מתודות שממומשות במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותן עבור מחלקה זו ומצד שני אנו חייבים לממש אותן כי הן נמצאות אצל "האבא" – פלייליסט.

DailyMix:

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימת המיקס היומית בנגן המוסיקה.
בכל יום נגריל באופן רנדומלי 10 שירים מסיפרית השירים והם ירכיבו לנו את המיקס היומי.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::string m_dailyMixPath - משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את שמות השירים של המיקס היומי.

std::set<std::string> m_dailyMix - זהו למעשה הקונטיינר בו השתמשנו על מנת לאחסן את שמות השירים של המיקס היומי.
השתמשנו בקונטיינר מסוג set בגלל שקונטיינר זה הוא ממויין ולכן יישמור לנו את שמות השירים בסדר אלפבתי וכאשר ננגן אותם הם כבר ייתנגנו בסדר אלפביתי כנדרש.
בנוסף קונטיינר זה מאוד יעיל בחיפוש, בהכנסה ובמחיקה של אלמנטים. ((Red-black-tree->log(n)).

std::unordered_map<std::string, char> m_options - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של הרשימת השמעה היומית.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של תפירט רשימת ההשמעה היומית במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

bool m_bufferFlag - דגל זה נועד על מנת לדלג על פונקצית ה- help (שמבוצעת אוטומטית אחרי כל פעולה) כאשר המשתמש בוחר בפעולת ה-help וזאת כדי לא להדפיס אותה פעמיים במסך ה-DOS.
בנוסף דגל זה מסייע לנו לזהות מתי בבאפר של ה- cin יש מחרוזת ריקה.

int m_day , int m_month, int m_year – משתנים אלו שומרים לנו את התאריך האחרון בו הפעלנו את הנגן וכך נוכל לדעת מתי התחלף יום.
אם התחלף יום ניצור מיקס יומי חדש המכיל 10 שירים שנלקחים באופן רנדומלי מהסיפריית השירים.

Function members:

void AddSong(std::string songName, Song songObj) – מתודה זו ממשומת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו כי לא ניתן להוסיף שירים למיקס היומי (אחת מרשימות ההשמעה הקבועות) ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void DeleteSongBySongName(std::string songName) – מתודה זו אחראית על מחיקת שיר מהמיקס היומי, כמובן שהיא תתבצע רק אם השם של השיר שנשלח למתודה קיים במיקס היומי.
(מתודה זו תתבצע רק במקרה ומחקנו שיר מהסיפריה אז הוא יימחק גם במיקס היומי).

void DeleteAllSongs() - מתודה זו ממשומת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו כי לא ניתן למחוק את כל השירים מהמיקס היומי (אחת מרשימות ההשמעה הקבועות) ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void UpdateSongName(std::string songName, std::string UpdateName) – מתודה זו אחראית על עדכון שם של שיר מסויים, כמובן שהיא תבצע את הפעולה רק אם שם השיר שנשלח למתודה קיים במיקס היומי.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית על בדיקה האם שיר קיים במיקס היומי לפי שם השיר שנשלח למתודה.

void Menu() - מתודה זו אחראית על הצגת התפריט של המיקס היומי למשתמש והפנייה למתודה המתאימה בהתאם לבחירה של המשתמש.

void Play() – מתודה זו אחראית לנגן את כל השירים במיקס היומי לפי סדר האלפבת.
(נזכיר שהקונטיינר שבחרנו לאיחסון שמות השירים כבר מסדר אותם לפי סדר האלפבת).

void PlayRandom() – מתודה זו אחראית לנגן את כל השירים במיקס היומי באופן רנדומלי.

void Print() – מתודה זו אחראית על הצגת המיקס היומי במסך ה-DOS.

bool CheckIfDayChanged() – מתודה זו בודקת אם היום התחלף מההפעלה האחרונה של הנגן.

void SelectRandomSongs() – מתודה זו מגרילה לנו 10 שירים באופן רנדומלי מסיפריית השירים וכך יוצרת מיקס יומי חדש כאשר מתחלף יום.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ השומר את המיקס היומי.

Song GetSongObjectBySongName(std::string songName) – מתודה זו ממומשת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void Help() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט של רשימת ההשמעה היומית.

char CheckUserParams(std::string input) - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות.

bool BufferHandling(std::string userInput) - פונקציה זו נועדה לטפל לנו במצב בו נשארת מחרוזת ריקה ב-cin לאחר שהמשתמש ביצע פעולה מסויימת.
מצב זה בעייתי כיוון שבאינטרציה הבאה למעשה המחרות הריקה שיש ב-cin "תישפך" לקלט של המשתמש ויווצר לנו קלט לא תקין ולא באשמת המשתמש.

MostPlayed:

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימת השירים שנשמעו הכי הרבה פעמים.
רשימה זו תככלול את 10 השירים שנוגנו הכי הרבה פעמים.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::multimap<int,std::string, std::greater<int>> m_mostPlayed – זהו למעשה הקונטיינר בו השתמשנו על מנת לאחסן את השירים שהושמעו הכי הרבה פעמים.
השתמשנו ב-multimap כיוון שהוא מאפשר שיהיו מספר מפתחות (keys) זהים ובנוסף הוא ממויין לפי הערכים במפתחות.
במפתחות (keys) שמנו את מספר הפעמים שהשיר הושמע ובערך (value) שמנו את שם השיר מכיוון שאנחנו רוצים שהשירים יהיו ממויינים לפי מספר הפעמים שהם הושמעו.
בנוסף על מנת שהשירים יהיו מסודרים מהשיר שהושמע הכי הרבה פעמים לשיר שהושמע הכי פחות פעמים (מהגדול לקטן) הוספנו **std::greater<int>** שאחראי על מיון זה ובכך כאשר נכניס את כל השירים בסיפריה לקונטיינר זה הם יסודרו מהשירים שהושמעו הכי הרבה ועד לשירים שהושמעו הכי מעט ואז נוכל לקחת את 10 השירים הראשונים ברשמיה זו וכך למעשה ליצור את רשימת השירים שהושמעו הכי הרבה פעמים.
בנוסף קונטיינר זה מאוד יעיל בחיפוש ((Red-black-tree->log(n)).

std::string m_mostPlayedPath - משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את שמות השירים של השירים שהושמעו הכי הרבה פעמים.

Function members:

void AddSong(std::string songName, Song songObj) – מתודה זו ממשומת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו כי לא ניתן להוסיף שירים לרשימת ההשמעה של השירים שהושמעו הכי הרבה פעמים (אחת מרשימות ההשמעה הקבועות) ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void DeleteAllSongs() - מתודה זו ממשומת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו כי לא ניתן למחוק את כל השירים מרשימת ההשמעה של השירים שהושמעו הכי הרבה פעמים (אחת מרשימות ההשמעה הקבועות) ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void DeleteSongBySongName(std::string songName) – מתודה זו אחראית על מחיקת שיר מרשימת ההשמעה של השירים שהושמעו הכי הרבה פעמים, כמובן שהיא תתבצע רק אם השם של השיר שנשלח למתודה קיים ברשימה זו.
(מתודה זו תתבצע רק במקרה ומחקנו שיר מהסיפריה אז הוא יימחק גם מרשימה זו).

void UpdateSongName(std::string songName, std::string UpdateName) – מתודה זו אחראית על עדכון שם של שיר מסויים, כמובן שהיא תבצע את הפעולה רק אם שם השיר שנשלח למתודה קיים ברשימת השירים שהושמעו הכי הרבה פעמים.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית על בדיקה האם שיר קיים ברשמית השירים של השירים שהושמעו הכי הרבה פעמים לפי שם השיר שנשלח למתודה.

void UpdateMostPlayed() – מתודה זו אחראית על עידכון רשימת השירים שהושמעו הכי הרבה.
בכל פעם שניכנס למתודה זו היא תסרוק שוב את כל סיפריית השירים ותיצור לנו רשימת שירים חדשה ומעודכנת כיוון שבכל פעם שמנוגן שיר המשתנה ששומר את מספר הפעמים שאותו שיר הושמע עולה ב-1 ומשתנה זה נמצא באובייקט Song (המכיל את כלל המאפיינים של השיר) שרק לסיפריית השירים יש גישה אליו והוא למעשה מוכל בקונטיינר שלה.

void Play() – מתודה זו אחראית לנגן את כל השירים ברשימת ההשמעה של השירים שהושמעו הכי הרבה פעמים מהשיר שנוגן הכי הרבה פעמים ועד השירים שנוגן הכי פחות פעמים.

void PlayRandom() – מתודה זו אחראית לנגן את כל השירים ברשימה באופן רנדומלי.

void Print() – מתודה זו אחראית על הצגת רשימת ההשמעה של השירים שנוגנו הכי הרבה במסך ה-DOS – השירים יוצגו מהשיר שנוגן הכי הרבה עד השיר שנוגן הכי פחות מתוך ה-10 שירים הקיימים ברשימה.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ השומר את רשימת השירים שנוגנו הכי הרבה פעמים.

void Menu() וגם **Song GetSongObjectBySongName(std::string songName)** - אלו מתודות שממומשות במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותן עבור מחלקה זו ומצד שני אנו חייבים לממש אותן כי הן נמצאות אצל "האבא" – פלייליסט.

Recent:

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימת השירים האחרונים שנוגנו.
רשימה זו תכלול את 10 השירים האחרונים שנוגנו.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::list<std::string> m_recentList - זהו למעשה הקונטיינר בו השתמשנו על מנת לאחסן את 10 השירים שנוגנו אחרונים.
בחרנו בקונטיינר זה כיוון שניתן לבצע בו push_front ובכך השיר שנוגן אחרון יהיה ראשון ברשימה.
בנוסף היינו צריכים גם את פעולת ה- push_back כיוון שהקובץ ששומר לנו את השירים שנוגנו לאחרונה שומר אותם מהשיר שנוגן אחרון וכך הלאה וכאשר אנו קוראים את הקובץ אנו צריכים לבצע push_back כדי לשמור על הסדר הזה (שהשיר שנוגן אחרון יהיה ראשון ברשימה) לכן השתמשנו ב- list ולא ב- stack או queue.

std::string m_recentPath - משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את שמות 10 השירים שנוגנו אחרונים.

Function members:

void AddSong(std::string songName, Song songObj) – מתודה זו אחראית על הוספת שיר חדש לרשימת השירים האחרונים שנוגנו והיא תבצע את ההוספה רק אם שם השיר שנשלח למתודה לא קיים כבר ברשימה זו.

void DeleteSongBySongName(std::string songName) – מתודה זו אחראית על מחיקת שיר מרשימת השירים האחרונים שנוגנו, כמובן שהיא תתבצע רק אם השם של השיר שנשלח למתודה קיים ברשימה זו.

void DeleteAllSongs() - מתודה זו ממשומת במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותה עבור מחלקה זו כי לא ניתן למחוק את כל השירים מרשימת השירים שנוגנו לאחרונה (אחת מרשימות ההשמעה הקבועות) ומצד שני אנו חייבים לממש אותה כי היא נמצאת אצל "האבא" – פלייליסט.

void UpdateSongName(std::string songName, std::string UpdateName) – מתודה זו אחראית על עדכון שם של שיר מסויים, כמובן שהיא תבצע את הפעולה רק אם שם השיר שנשלח למתודה קיים ברשימת השירים שנוגנו לאחרונה.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית על בדיקה האם שיר קיים ברשמית השירים שנוגנו לאחרונה לפי שם השיר שנשלח למתודה.

void Play() – מתודה זו אחראית לנגן את כל השירים ברשימת השירים שנוגנו לאחרונה מהשיר שנוגן אחרון והלאה – 10 שירים.

void PlayRandom() – מתודה זו אחראית לנגן את כל השירים ברשימה באופן רנדומלי.

void Print() – מתודה זו אחראית על הצגת רשימת ההשמעה של השירים שנוגנו לאחרונה במסך ה – DOS – למעשה יוצגו 10 השירים שנוגנו אחרונים.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ השומר את רשימת השירים שנוגנו לאחרונה.

void Menu() וגם **Song GetSongObjectBySongName(std::string songName)** - אלו מתודות שממומשות במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותן עבור מחלקה זו ומצד שני אנו חייבים לממש אותן כי הן נמצאות אצל "האבא" – פלייליסט.

Deleted:

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימת השירים שנחמקו.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, Song> m_deleted - זהו למעשה הקונטיינר בו השתמשנו לרשימת השירים שנמחקו מהסיפריה.
בחרנו בקונטיינר זה כיוון שהוספנו לנגן פעולה המבצעת שיחזור של שירים שנמחקו מהסיפריה וכדי לשחזר שיר שנמחק לא מספיק לשמור רק את השם של השיר אלה צריך לשמור גם את כל המאפיינים של השיר שהם בעצם נשמרים באובייקט Song.
לכן בכל פעם שנמחוק מהסיפריה שיר , נבצע קודם העברה של שם השיר והאובייקט Song (המאפיינים של השיר) השייך לאותו השיר לרשימת ה- Deleted ואז נמחוק את השיר מהסיפריה.
בנוסף קונטיינר זה מאוד יעיל בחיפוש,בהכנסה ובמחיקה של אלמנטים. ((Red-black-tree->log(n)).

std::string m_deletedPath - משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את השמות של השירים שנמחקו וכל המאפיינים שלהם (Song object).

Function members:

void AddSong(std::string songName, Song songObj) – מתודה זו אחראית על הוספת שיר חדש לרשימת השירים שנמחקו והיא תבצע את ההוספה רק אם שם השיר שנשלח למתודה לא קיים כבר ברשימה זו.

void DeleteSongBySongName(std::string songName) – מתודה זו אחראית על מחיקת שיר מרשימת השירים שנמחקו, כמובן שהיא תתבצע רק אם השם של השיר שנשלח למתודה קיים ברשימה זו.
שיר שנמחק מרשימה זו יימחק לצמיתות ולא יהיה ניתן לשחזר אותו.

void DeleteAllSongs() – מתודה זו אחראית על מחיקת כל השירים הקיימים ברשימת השירים שנמחקו.
כל השירים יימחקו לצמיתות ולא יהיה ניתן לשחזר אותם.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית על בדיקה האם שיר קיים ברשמית השירים שנמחקו לפי שם השיר שנשלח למתודה.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ השומר את רשימת השירים שנמחקו.

Song GetSongObjectBySongName(std::string songName) – מתודה זו מחזירה את האובייקט Song לפי שם השיר שנשלח אלייה.
אנו משתמשים במתודה זו כאשר אנו משחזרים שיר מהרשימה של השירים שנמחקו לסיפריה.

void Print() – מתודה זו אחראית על הצגת רשימת השירים שנמחקו במסך ה – DOS.

void UpdateSongName(std::string songName, std::string UpdateName), void Menu(),

void Play(), void PlayRandom() - אלו מתודות שממומשות במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותן עבור מחלקה זו ומצד שני אנו חייבים לממש אותן כי הן נמצאות אצל "האבא" – פלייליסט.

:RegularPlayList

תיאור כללי:

מחלקה זו היא אחת היורשות של המחלקה האבסטרקטית פלייליסט והיא למעשה מנהלת לנו את רשימות השירים שנוצרות ע"י המשתמש במהלך השימוש בנגן המוסיקה.

:Data members

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::set<std::string> m_regularPlayList - זהו למעשה הקונטיינר בו השתמשנו לשמור את שמות השירים שאנו מכניסים לרשימת השירים שהמשתמש יוצר.
בחרנו בקונטיינר זה בגלל שהוא גם מונע כפילויות כלומר לא ניתן להכניס את אותו השיר פעמיים וגם כי הוא יעיל בחיפוש,בהכנסה ובמחיקה של אלמנטים. ((Red-black-tree->log(n)).
בנוסף קונטיינר זה הוא ממויין וכאשר ננגן את רשימת ההשמעה היא תתנגן לפי סדר האלפבתי של שמות השירים.

std::string m_regularPlayListPath - משתנה ששומר לנו את הנתיב המלא לקובץ המכיל את השמות של השירים ברשימת השמעה שהמשתמש יצר.

std::string m_regularPlayListName – משתנה זה שומר לנו את שם רשימת ההשמעה שהמשתמש יצר.

:Function members

void AddSong(std::string songName, Song songObj) – מתודה זו אחראית על הוספת שיר חדש לרשימת ההשמעה שהמשתמש יצר והיא תבצע את ההוספה רק אם שם השיר שנשלח למתודה לא קיים כבר ברשימה זו.

void DeleteSongBySongName(std::string songName) – מתודה זו אחראית על מחיקת שיר מרשימת ההשמעה שהמשתמש יצר, כמובן שהיא תתבצע רק אם השם של השיר שנשלח למתודה קיים ברשימה זו.

void DeleteAllSongs() – מתודה זו אחראית על מחיקת כל השירים הקיימים ברשימה זו.

void UpdateSongName(std::string songName, std::string UpdateName) – מתודה זו אחראית על עדכון שם של שיר מסויים, כמובן שהיא תבצע את הפעולה רק אם שם השיר שנשלח למתודה קיים ברשימה זו.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית על בדיקה האם שיר קיים ברשימת ההשמעה שהמשתמש יצר לפי שם השיר שנשלח למתודה.

void Play() – מתודה זו אחראית לנגן את כל השירים ברשמית ההשמעה שהמשתמש יצר לפי סדר האלפבת.
(נזכיר שהקונטיינר שבחרנו לאיחסון שמות השירים כבר מסדר אותם לפי סדר האלפבת).

void PlayRandom() – מתודה זו אחראית לנגן את כל השירים ברשימה זו באופן רנדומלי.

void Print() – מתודה זו אחראית על הצגת רשימת ההשמעה שהמשתמש יצר במסך ה-DOS.

std::string GetPlaylistPath() const – מתודה זו אחראית על החזרת הנתיב המלא לקובץ ששומר את רשימת ההשמעה שהמשתמש יצר.

void Menu() וגם **Song GetSongObjectBySongName(std::string songName)** - אלו מתודות שממומשות במחלקה זו באופן סתמי בגלל שמצד אחד אנו לא צריכים אותן עבור מחלקה זו ומצד שני אנו חייבים לממש אותן כי הן נמצאות אצל "האבא" – פלייליסט.

Song:

תיאור כללי:

מחלקה זו למעשה אחראית על איחסון כל המאפיינים של כל שיר כמו שם השיר, שם הזמר, ז'אנר, אלבום, הנתיב לשיר וכ'ו...

Data members:

std::string m_songName – משתנה זה שומר לנו את שם השיר.

std::string m_singerName – משתנה זה שומר לנו את שם הזמר.

std::string m_album – משתנה זה שומר לנו את שם האלבום.

std::string m_songPath – משתנה זה שומר לנו את הנתיב המלא לשיר.

std::string m_publishingYear – משתנה זה שומר לנו את שנת ההוצאה של השיר.

std::string m_duration – משתנה זה שומר לנו את משך הזמן של השיר.

std::string m_genre – משתנה זה שומר לנו את הז'אנר של השיר.

int m_numberOfPlays – משתנה זה שומר לנו את מספר הפעמים שהשיר נוגן.

int m_songId – משתנה זה שומר לנו את המספר הייחודי של השיר.

static int SerialNumber – משתנה סטטי שבכל פעם שאנו יוצרים אובייקט Song עולה ב-1 ונשמר במשתנה songId וכך למעשה לכל שיר יהיה מספר ייחודי משלו.

static int NumberOfSongs – משתנה זה נועד כדי שנדע בכל רגע נתון כמה שירים יש לנו במערכת.
כאשר אנו יוצרים שיר הוא עולה ב-1 וכאשר אנו מוחקים שיר הוא יורד ב-1.

Function members:

int GetNumberOfSongs() const – מתודה זו אחראית להחזיר את מספר השירים הקיימים באותו הרגע בנגן.

std::string GetSongName() const – מתודה זו אחראית להחזיר את שם השיר.

std::string GetSingerName() const – מתודה זו אחראית להחזיר את שם הזמר.

std::string GetAlbum() const – מתודה זו אחראית להחזיר את שם האלבום לו שייך השיר.

std::string GetPublishingYear() const – מתודה זו אחראית להחזיר את שנת הפרסום של השיר.

std::string GetDuration() const – מתודה זו אחראית להחזיר את משך הזמן של השיר.

std::string GetSongPath() const – מתודה זו אחראית להחזיר את הנתיב המלא לשיר.

std::string GetGenre() const – מתודה זו אחראית להחזיר את הז'אנר לו שייך השיר.

int GetNumberOfPlays() const – מתודה זו אחראית להחזיר את מספר הפעמים שהשיר נוגן.

int GetSongId() const – מתודה זו אחראית להחזיר את המספר הייחודי של השיר.

void SetGenre(std::string genre) – מתודה זו אחראית להגדיר את שם הז'אנר לו שייך השיר.

void SetSongName(std::string songName) – מתודה זו אחראית להגדיר את שם השיר.

void SetSingerName(std::string singerName) – מתודה זו אחראית להגדיר את שם הזמר.

void SetAlbum(std::string album) – מתודה זו אחראית להגדיר את שם האלבום לו שייך השיר.

void SetPublishingYear(std::string publishingYear) – מתודה זו אחראית להגדיר את שנת הפרסום של השיר.

void SetDuration(std::string duration) – מתודה זו אחראית להגדיר את משך הזמן של השיר.

void SetSongPath(std::string songPath) – מתודה זו אחראית להגדיר את הנתיב המלא לשיר.

void SetNumberOfPlays(int numberOfPlays) – מתודה זו אחראית להגדיר את מספר הפעמים שהשיר נוגן.

void SetSongId(int songId) – מתודה זו אחראית להגדיר את המספר הייחודי של השיר.

void IncreaseNumberOfPlays() – מתודה זו אחראית להעלות את מספר הפעמים שנוגן השיר ב-1.

Library:

תיאור כללי:

מחלקה זו למעשה אחראית על ניהול סיפריית השירים בפרוייקט שלנו.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::set<std::string>* m_regularPlaylistNams מצביע לקונטיינר המיכל את שמות רשימות ההשמעה שהמשתמש הוסיף במהלך השימוש בנגן.

std::unordered_map<std::string, char> m_options - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של הסיפריה.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של תפריט הסיפריה במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

std::vector<std::string> m_selections - קונטיינר זה אחראי לאחסן לנו את הקלט של המשתמש לאחר שהופרד ע"י רווחים.
בחרנו בקונטיינר זה מכיוון שאנו צריכים גישה לכל איבר בוקטור ע"י אינדקס ספציפי.

int* m_numberOfRegularPlaylists – מצביע למשתנה המכיל את מספר רשימות ההשמעה שהמשתמש יצר.

int m_count – משתנה זה למעשה ייקבע לנו כמה שירים להציג על מסך ה-DOS כאשר אנו מדפיסים את השירים הקיימים בסיפרית השירים.

bool m_bufferFlag - דגל זה נועד על מנת לדלג על פונקצית ה- help (שמבוצעת אוטומטית אחרי כל פעולה) כאשר המשתמש בוחר בפעולת ה-help וזאת כדי לא להדפיס אותה פעמיים במסך ה-DOS.
בנוסף דגל זה מסייע לנו לזהות מתי בבאפר של ה- cin יש מחרוזת ריקה.

std::string m_songName, m_album, m_duration, m_genre, m_singer – משתנים השומרים לנו את העידכונים של המשתמש כאשר מעדכנים או מוסיפים שיר מוסיים.

Player m_player – הגדרת אובייקט ממחלקת שחקן (Player) שיהיה אחראי על ניגון השירים.

Function members:

std::map<std::string, Song>* GetLibrary() const – מתודה זו מחזירה לנו מצביע לקונטיינר של הסיפריה שמכיל את שם השיר במפתח (key) והמאפיינים של השיר בערך (value) שהם למעשה אובייקט מסוג Song.

int GetCount() const – מתודה זו מחזירה את מספר השירים מהסיפריה שיש להציג על מסך ה-DOS.

void SetCount(int count) – מתודה זו מגדירה את מספר השירים מהסיפריה שיש להציג על מסך ה-DOS.

void Menu() – מתודה זו אחראית על הצגת התפריט של הסיפריה למשתמש והפנייה למתודה המתאימה בהתאם לבחירה של המשתמש.

void PrintSongsByGivenCount(int count) – מתודה זו אחראית על הדפסת השירים הקיימים בסיפריה למסך ה- DOS.
מספר השירים שהמתודה תדפיס למסך תלוי בפרמטר count שנשלח למתודה זו, אם פרמטר זה גדול ממספר השירים שיש בסיפריה אז כל השירים בסיפריה יודפסו למסך ה-DOS.

bool ExtensionValidate(std::string path) – מתודה זו בודקת אם הנתיב לשיר מסתיים ב-mp3. וכך ניתן לדעת אם הנתיב של השיר חוקי (כלומר ניתן לנגן את השיר) או לא.

std::string FindSongNameBySongId(int songId) – מתודה זו אחראית על חיפוש שם של שיר לפי המספר הייחודי שלו.
אם היא מוצאת התאמה כזו היא תחזיר את שם השיר ואם לא היא תחזיר מחזורת ריקה.

void DeleteSong(std::string songName,int songId) – מתודה זו אחראית למחוק שיר מהסיפריה או לפי המספר הייחודי של השיר או לפי שם השיר – תלוי בפרמטרים שאנו שולחים.

void AddSong(std::string songName, std::string songPath) – מתודה זו אחראית על הוספת שיר לסיפריית השירים – המשתמש יהיה חייב להכניס את שם השיר והנתיב המלא לשיר.

void PrintSongDetails(std::string songName, int songId) – מתודה זו אחראית על הדפסת כל המאפיינים של השיר המוכלים באובייקט Song (זמר, ז'אנר, אלבום וכ'ו..) ובנוסף באיזה רשימות השמעה השיר קיים.
ניתן להדפיס את הפרטים של שיר מסויים לפי שם השיר או לפי המספר הייחודי שלו – תלוי בפרמטרים שהמשתמש שולח למתודה זו.

void AddSongToPlaylist(int songId, std::string playlistName) – מתודה זו אחראית על העברת (העתקת) שיר מסיפרית השירים לרשימת השמעה מסויימת.
המתודה מקבלת את המספר הייחודי של השיר שנרצה להעביר ואת שם רשימת ההשמעה שאלייה יעבור השיר.

void RemoveSongFromPlaylist(std::string songName , std::string playlistName) – מתודה זו אחראית על הסרת שיר מרשימת השמעה מסויימת.
מתודה זו מקבלת את שם השיר שנרצה להסיר ואת שם רשימת ההשמעה שממנה נסיר את השיר.

void PrintAllPlaylist() const – מתודה זו אחראית להדפיס למסך ה- DOS את כל רשימות ההשמעה שקיימות בנגן המוסיקה.

void PlaySong(std::string songName, int songId) – מתודה זו אחראית על ניגון שיר ספציפי מסיפרית השירם.
ניתן לנגן שיר או לפי המספר הייחודי שלו או לפי שם השיר – תלוי בפרמטרים שנשלח למתודה זו.

void PlayAllSongs() – מתודה זו אחראית לנגן את כל השירים בסיפריה לפי סדר האלפבת.
נזכיר שהקונטיינר בו השתמשנו לאיחוסון השירים הוא map שבמפתח שלו נמצא שם השיר ובערך שלו נמצא אובייקט מסוג Song לכן השירים בקונטיינר זה כבר ממויינים באופן אוטומטי לפי סדר האלפבת.

void PlayAllSongsRandom() – מתודה זו אחראית לנגן את כל השירים בסיפריה באופן רנדומלי.

void Undelete(std::string songName) – מתודה זו אחראית על שיחזור שיר מרשימת השירים שנמחקו בחזרה לסיפרית השירים.

bool CheckIfSongExist(std::string songName) – מתודה זו אחראית לבדוק אם שיר מסויים קיים בסיפריה.

void Help() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט הסיפריה.

char CheckUserParams() - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות.

void ClearSongParams() – מתודה זו אחראית "לנקות" את המשתנים השומרים את הפרמטרים שהמשתמש הכניס בזמן עדכון או הוספה של שיר מסויים.
לאחר מתודה זו כל המשתנים המאפיינים שיר יהיו מאותחלים ל- "Unknown" .

bool BufferHandling(std::string userInput) - פונקציה זו נועדה לטפל לנו במצב בו נשארת מחרוזת ריקה ב-cin לאחר שהמשתמש ביצע פעולה מסויימת.
מצב זה בעייתי כיוון שבאינטרציה הבאה למעשה המחרות הריקה שיש ב-cin "תישפך" לקלט של המשתמש ויווצר לנו קלט לא תקין ולא באשמת המשתמש.

bool CheckAddParams(), bool CheckUpdateParams(), bool checkPrintSongParams(), bool CheckDeleteParams(), bool CheckAdd2PlParams(), bool CheckRemoveFromPlParams(), bool CheckPlayParams(), bool CheckUndeleteParams() - כל המתודות הללו אחראיות לבדוק שמספר הפרמטרים שהמשתמש הכניס תואם למספר הפרמטרים הנידרש לפעולת מסויימת.

PlayListOps:

תיאור כללי:

מחלקה זו אחראית למעשה לבצע את כל הפעולות שהמשתמש יכול לבצע ברמת רשימות ההשמעה, לדוגמא להוסיף רשימת השמעה, למחוק רשימת השמעה, לנגן רשימת השמעה וכ'ו...

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::set<std::string>* m_regularPlayListNames – מצביע לקונטיינר המכיל את כל השמות של רשימות ההשמעה שהמשתמש יצר במהלך השימוש בנגן המוסיקה.

std::set<std::string>* m_constPlayLists – מצביע לקונטיינר המכיל את כל רשימות ההשמעה הקבועות שלא ניתן למחוק אותן (Recent, DailyMix, MostPlayed).

std::unordered_map<std::string, char> m_options - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של הפעולות על רשימות ההשמעה.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של התפריט של הפעולות על רשימות ההשמעה במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

std::vector<std::string> m_selections - קונטיינר זה אחראי לאחסן לנו את הקלט של המשתמש לאחר שהופרד ע"י רווחים.
בחרנו בקונטיינר זה מכיוון שאנו צריכים גישה לכל איבר בוקטור ע"י אינדקס ספציפי.

bool m_bufferFlag - דגל זה נועד על מנת לדלג על פונקצית ה- help (שמבוצעת אוטומטית אחרי כל פעולה) כאשר המשתמש בוחר בפעולת ה-help וזאת כדי לא להדפיס אותה פעמיים במסך ה-DOS.
בנוסף דגל זה מסייע לנו לזהות מתי בבאפר של ה- cin יש מחרוזת ריקה.

int* m_numberOfRegularPlaylists – מצביע למשתנה המכיל את מספר רשימות ההשמעה שהמשתמש יצר במהלך השימוש בנגן המוסיקה.

Function members:

void AddPlayList(std::string playListName) – מתודה זו אחראית על הוספת רשימת השמעה לנגן המוסיקה, כמובן שהיא קודם בודקת אם רשימת ההשמעה כבר קיימת בנגן המוסיקה.

void DeletePlayList(std::string playListName) – מתודה זו אחראית על מחיקת רשימת השמעה (לא כולל הרשימות השמעה הקבועות) מנגן המוסיקה.

void PlayPlaylist(std::string playListName) – מתודה זו אחראית על ניגון כל השירים ברשימת השמעה מסויימת.

void PlayPlaylistRandom(std::string playListName) – מתודה זו אחראית על ניגון כל השירים ברשימת השמעה מסויימת באופן רנדומלי.

void PrintPlaylist(std::string playListName) – מתודה זו אחראית להדפיס את כל השירים ברשימת השמעה מסויימת למסך ה- DOS.

void PrintAllExistPlayLists() const – מתודה זו אחראית להדפיס את כל רשימות ההשמעה שקיימות בנגן המוסיקה למסך ה- DOS.

void PlayListMenu() - מתודה זו אחראית על הצגת התפריט של הפעולות שהמשתמש יכול לבצע על רשימת השמעה והפנייה למתודה המתאימה בהתאם לבחירה של המשתמש.

void Help() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט רשימות ההשמעה.

bool BufferHandling(std::string userInput) - פונקציה זו נועדה לטפל לנו במצב בו נשארת מחרוזת ריקה ב-cin לאחר שהמשתמש ביצע פעולה מסויימת.
מצב זה בעייתי כיוון שבאינטרציה הבאה למעשה המחרות הריקה שיש ב-cin "תישפך" לקלט של המשתמש ויווצר לנו קלט לא תקין ולא באשמת המשתמש.

char CheckUserParams() - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות.

bool CheckPlaylistParams() - מתודה זו אחראית לבדוק שמספר הפרמטרים שהמשתמש הכניס תואם למספר הפרמטרים הנידרש לפעולות ה-playlist.

Search:

תיאור כללי:

מתודה זו אחראית על יצירת ממשק חיפוש נוח למשתמש לפי כל מיני קטגוריות שייבחר כמו: שם הזמר, שם הז'אנר, שם האלבום וכ'ו...
בנסונף מתודה זו תשמור לנו את היסטוריית החיפושים של המשתמש.

Data members:

std::map<std::string, Song>* m_library - מצביע לקונטיינר של הסיפריה המכילה את כל השמות של השירים והמאפיינים שלהם.

std::map<std::string, PlayList*>* m_listOfPalyLists – מצביע לקונטיינר המכיל את כל שמות רשימות ההשמעה במפתח (key) ובערך (value) מכיל מצביע למחלקה האבסטרקטית של פלייליסט.

std::set<std::string> m_search – בקונטיינר זה למעשה נשמור בכל פעם את כל השירים השייכים לקטגוריה מסומיימת שהמשתמש חיפש, לדומא אם המשתמש חיפש את כל השירים של זמר מסויים אזי כל השירים של אותו הזמר יישמרו באופן זמני בקונטיינר זה ואז המשתמש יוכל לבחור איזה פעולות לבצע על שירים אלו כמו לדוגמא לנגן את כל השירים או לנגן שיר ספציפי מהרשימה הזו.
לאחר סיום הפעולות של המשתמש על רשימה זו היא תימחק ותמתין לחיפוש הבא של המשתמש.
בחרנו בקונטיינר זה מכיוון שזהו קונטיינר שמונע כפילויות וכך לא יכול להיווצר לנו מצב שבו שם של שיר נמצא פעמיים ברשימה.
בנוסך זהו קונטיינר ממויין ואז ברגע שנגן רשימת שירים מסויימת היא תנוגן בסדר אלפבתי.
בנוסף קונטיינר זה מאוד יעיל בחיפוש, בהכנסה ובמחיקה של אלמנטים. (Red-black-tree->log(n)).

std::list<std::string> m_history – קונטיינר זה למעשה שומר לנו את היסטורית החיפושים של המשתמש מהחיפוש האחרון שעשה והאלה...
בחרנו בקונטיינר זה כיוון שניתן לבצע בו push_front ובכך החיפוש האחרון של המשתמש יהיה הראשון ברשימת ההיסטוריה של החיפושים.
בנוסף היינו צריכים גם את פעולת ה- push_back כיוון שהקובץ ששומר לנו את החיפושים האחרונים של המשתמש שומר אותם מהחיפוש האחרון שהמשתמש ביצע וכך הלאה וכאשר אנו קוראים את הקובץ אנו צריכים לבצע push_back כדי לשמור על הסדר הזה (שהחיפוש האחרון יהיה הראשון ברשימה וכך הלאה...)
לכן השתמשנו ב- list ולא ב- stack או queue.

std::string m_searchPath – משתנה זה שומר לנו את הנתיב המלא לקובץ ששומר לנו את היסטורית החיפושים של המשתמש.

Player m_player - הגדרת אובייקט ממחלקת שחקן (Player) שיהיה אחראי על ניגון השירים.

int m_count – משתנה שקובע את מספר החיפושים מהיסטורית החיפושים שיופיעו על מסך ה-DOS כאשר נבקש להדפיס את היסטוריית החיפושים.
אם המספר במשתנה זה יהיה גדול יותר ממספר החיפושים שיש אז כל החיפושים שקיימים יופיעו על מסך ה-DOS.

std::unordered_map<std::string, char> m_searchOptions – קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של החיפושים.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של תפריט החיפוש במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

std::unordered_map<std::string, char> m_deleteOptions - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של מחיקת היסטורית החיפושים.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של תפריט מחיקת היסטורית החיפושים במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

std::unordered_map<std::string, char> m_playOptions - קונטיינר זה אחראי לאחסן לנו את כל הפעולות שהמשתמש יכול לבצע בתפריט של ניגון החיפושים.
במפתחות שלו (keys) הוא מכיל את הפעולות המותרות ובערכים שלו (values) הוא מכיל את התו המתאים לפעולה לפי מה שהגדרנו ב-enum של תפריט ניגון החיפושים במחלקה זו.
בחרנו בקונטיינר זה מכיוון שהוא ממומש בשימוש ב- hash table ולכן החיפוש, ההכנסת איברים והמחיקת אלמנטים בו מתבצעים ב-o(1).

std::vector<std::string> m_selections - קונטיינר זה אחראי לאחסן לנו את הקלט של המשתמש לאחר שהופרד ע"י רווחים.
בחרנו בקונטיינר זה מכיוון שאנו צריכים גישה לכל איבר בוקטור ע"י אינדקס ספציפי.

bool m_bufferFlag - דגל זה נועד על מנת לדלג על פונקצית ה- help (שמבוצעת אוטומטית אחרי כל פעולה) כאשר המשתמש בוחר בפעולת ה-help וזאת כדי לא להדפיס אותה פעמיים במסך ה-DOS.
בנוסף דגל זה מסייע לנו לזהות מתי בבאפר של ה- cin יש מחרוזת ריקה.

Function members:

void SearchMenu() - מתודה זו אחראית על הצגת תפריט החיפוש למשתמש והפנייה למתודה המתאימה בהתאם לבחירה של המשתמש.

void PlayMenu() – מתודה זו אחראית על הצגת תפריט הניגון למשתמש והפנייה למתודה המתאימה בהתאם לבחירה של המשתמש.

void PlayBySongName(std::string songName) – מתודה זה אחראית לנגן שיר ספציפי מרשימת החיפושים בהתאם לחיפוש שהמשתמש ביצע.
כמובן שמתודה זו קודם תבדוק אם השיר קיים ברשימת החיפושים.

void PlayAll() – מתודה זו אחראית על ניגון כל השירים ברשימת החיפושים בהתאם לחיפוש שהמשתמש ביצע.

void SearchBySongName(std::string songName) – מתודה זו אחראית על חיפוש שיר ספציפי לפי שם השיר.

void SearchBySingerName(std::string singerName) – מתודה זו אחראית על חיפוש כל השירים של זמר מסויים.

void SearchByAlbum(std::string albumName) – מתודה זו אחראית על חיפוש כל השירים השייכים לאלבום מסויים.

void SearchByGenre(std::string genreName) – מתודה זו אחראית על חיפוש כל השירים השייכים לז'אנר מסויים.

void InsertToHistorySearch(std::string search) – מתודה זו אחראית על הכנסת חיפוש שהמשתמש ביצע לרשימת היסטורית החיפושים.

void PrintHistoryByGivenCount(int count) – מתודה זו אחראית על הדפסת היסטורית החיפושים של המשתמש.
המתודה תדפיס למסך ה – count DOS חיפושים ואם count יהיה גדול יותר ממספר החיפושים שהמשתמש ביצע אזי כל היסטורית החיפושים תוצג במסך ה- DOS.

char DeleteMenu() – מתודה זו אחראית על הצגת תפריט מחיקת החיפושים למשתמש והחזרת הבחירה של המשתמש.

void DeleteHistory() – מתודה זו למעשה מבצעת את מחיקת החיפוש מהיסטורית החיפושים לפי הערך שחוזר מהמתודה **char DeleteMenu().**

bool ChekIfExsitInHistory(std::string search) – מתודה זו אחראית לבדוק אם חיפוש מסויים שביצע המשתמש קיים ברשימת היסטורית החיפושים.

bool BufferHandling(std::string userInput) - פונקציה זו נועדה לטפל לנו במצב בו נשארת מחרוזת ריקה ב-cin לאחר שהמשתמש ביצע פעולה מסויימת.
מצב זה בעייתי כיוון שבאינטרציה הבאה למעשה המחרות הריקה שיש ב-cin "תישפך" לקלט של המשתמש ויווצר לנו קלט לא תקין ולא באשמת המשתמש.

void Help() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט החיפושים.

void DeleteHelp() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט של מחיקת היסטורית החיפושים.

void PlayHelp() const - מתודה זו אחראית להדפיס את כלל הפעולות שהמשתמש יכול לבצע בתפריט של ניגון היסטורית החיפושים.

char CheckUserParamsSearchMenue() - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות בתפריט החיפושים.

char CheckUserParamsDeleteMenu() - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות בתפריט מחיקת היסטורית החיפושים.

char CheckUserParamsPlayMenu() - מתודה זו אחראית לבדוק אם הקלט שהמשתמש הכניס תקין ונמצא בקונטיינר המכיל את כל הפעולות המותרות בתפריט ניגון היסטורית החיפושים.

bool CheckSearchParams(), bool CheckDeleteParams() - המתודות הללו אחראיות לבדוק שמספר הפרמטרים שהמשתמש הכניס תואם למספר הפרמטרים הנידרש לפעולת מסויימות.

הממשק למשתמש:

1. בתחילת ההרצה של התוכנית המשתמש יוכל לבחור בין מספר פעולות:

* Daily Mix
* Search
* Podcasts
* Library
* Play List
* Help
* Exit

```
--------------- Main Menu ---------------

1. For Daily Mix write:
   Daily Mix

2. For Search write:
   Search

3. For Podcasts write:
   Podcasts

4. For Library write:
   Library

5. For Play List write:
   Play List

5. For help write:
   Help

7. For Exit write:
   Exit

-----------------------------------------
```

2. כעת המשתמש ייצטרך לבחור באחת מהאופציות ע"י כתיבת האופציה הרצויה במסך ה – DOS.
לדוגמא אם המשתמש רוצה לבצע פעולות על רשימת השמעה מסויימת הוא ייכנס לתפריט של ה- Play List:

* ניתן לראות את הפקודה שהמשתמש רשם (בצהוב) ולאחר מכן רשימה של כלל רשימות ההשמעה הקיימות בתוכנית כולל אלו שהוספנו תוך כדי השימוש בנגן:

```
-----------------------------------------
Play List
All existing playlist names:
1. Favorite
2. MostPlayed
3. Recent
4. DailyMix
5. Lian_playlist
6. Podcasts
7. Tohar_playlist
8. Deleted
```

לאחר מכן יופיעו כל הפעולות שניתן לבצע על רשימת השמעה מסויימת מהרשימה שהוצגה לעיל:

```
--------------- Playlist Menu ---------------

1. For add playlist write:
   Add <playlist_name>
** You must not include a space in the playlist name **

2. For delete playlist write:
   Delete <playlist_name>
** You must not include a space in the playlist name **

3. For play playlist write:
   Play <playlist_name>
** You must not include a space in the playlist name **

4. For play playlist randomly write:
   PlayRandom <playlist_name>
** You must not include a space in the playlist name **

5. For print playlist write:
   Print <playlist_name>
** You must not include a space in the playlist name **

6. For help write:
   Help

7. For back to main menu write:
   Back

-------------------------------------------------------
```

3. כעת אם המשתמש יירצה לבצע פעולה כלשהי בתפריט זה הוא ייצטרך לרשום את הפעולה שהוא רוצה לבצע בדיוק בצורה שכתובה בתפריט.
לדוגמא אם להוסיף רשימת השמעה חדש לרשימות השמעה הקיימות:

```
-------------------------------------------------------
Add <New_playlist>
Playlist named New_playlist added to list of playlist
```

* לאחר שנחזור אחורה לתפריט הראשי ואז שוב ניכנס לתפריט של הפעולות על רשימות ההשמעה נראה את רשימת ההשמעה שהוספנו בתור אחת מרשימות ההשמעה הקיימות בנגן מוסיקה זה.

נדגים זאת למטה...

4. כעת נחזור אחורה לתפריט הראשי ע"י כתיבת הפקודה – Back בתפריט זה:

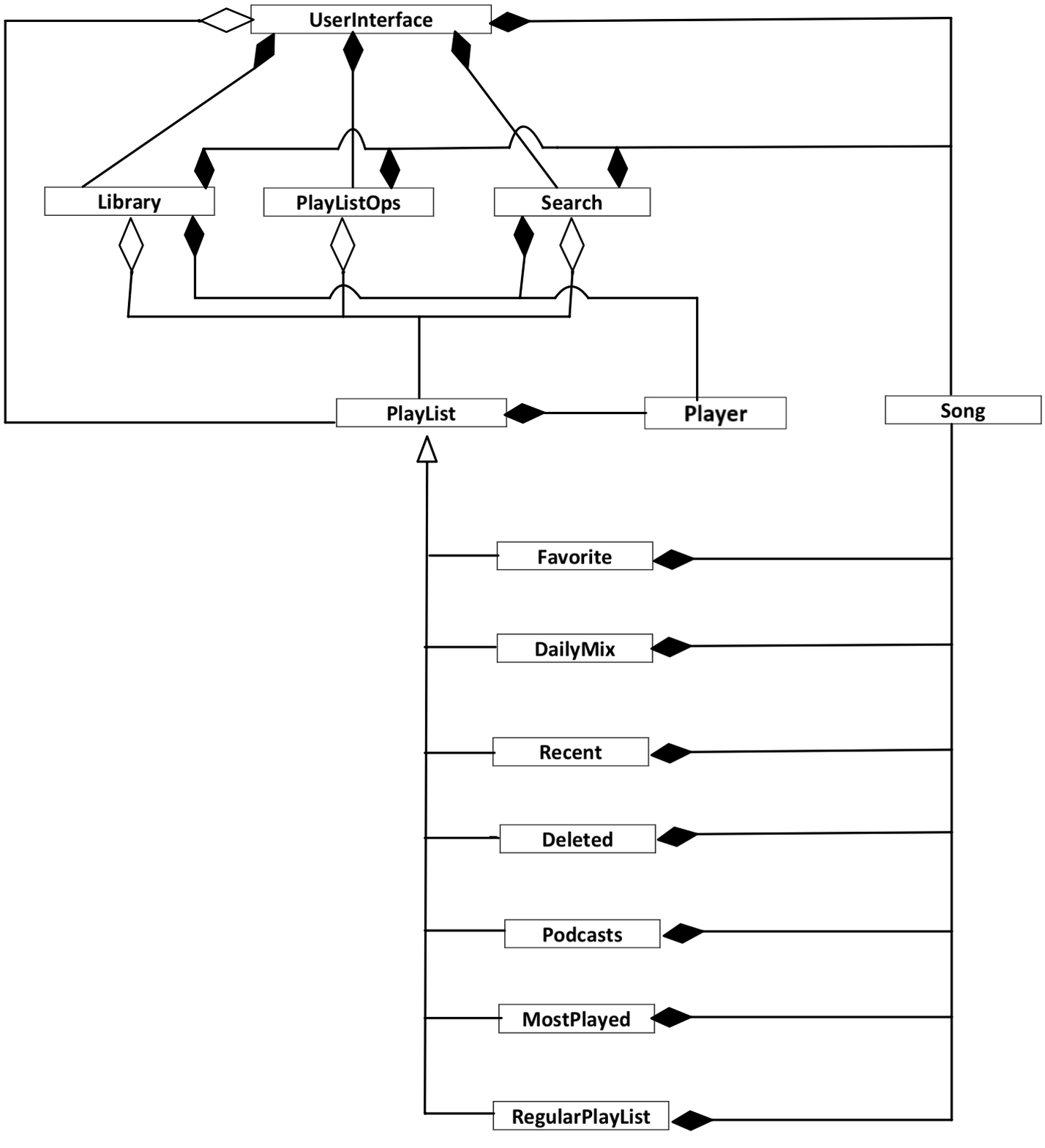
```
----------------------------------------
Back
--------------- Main Menu -------------

1. For Daily Mix write:
   Daily Mix

2. For Search write:
   Search

3. For Podcasts write:
   Podcasts

4. For Library write:
   Library

5. For Play List write:
   Play List

5. For help write:
   Help

7. For Exit write:
   Exit

----------------------------------------
```

5. ניכנס שוב לתפריט ה- Play List כדי לראות את רשימת ההשמעה החדשה שהוספנו:

```
---------------------------------
Play List
All existing playlist names:
1. Favorite
2. MostPlayed
3. Recent
4. DailyMix
5. Lian_playlist
6. New_playlist
7. Podcasts
8. Tohar_playlist
9. Deleted
```

באופן הזה ניתן לבצע את כל הפעולות האפשריות בנגן מוסיקה זה בצורה פשוטה ונוחה למשתמש!

UML Diagram


UserInterface
Library
PlayListOps
Search
PlayList
Player
Song
Favorite
DailyMix
Recent
Deleted
Podcasts
MostPlayed
RegularPlayList

UML Diagram Details:

UserInteface
-m_libraryPath : string -m_searchObj : Search -m_library : LibraryObj -m_playListOps : PlayListOps -m_library : map<string ,Song> -m_listOfPlaylist : map<string , PlayList*> -m_constPlayLists : set<string> -m_regularPlayListNames : set<string> -m_numberOfRegularPlayLists : int -m_options : unordered_map<string , char> -m_bufferFlag : bool
+Play() : void -Help() const : void -CheckParams(string : input) : char -BufferHandling(string : userInput) : bool

PlayListOps
- m_library : map<string , Song>* - m_listOfPlayLists : map<string , PlayList*>* - m_regularPlayListNames : set<string>* - m_constPlayLists : set<string>* - m_numberOfRegularPlaylists : int* - m_options : unordered_map<string , char> - m_selections : vector<string> - m_bufferFlag : bool
- AddPlayList(playListName : string) : void - DeletePlayList(playlistName : string) :void - PlayPlaylist(playListName : string) : void - PlayPlaylistRandom(playListName : string) - PlayListMenu() : void - Help() const : void - BufferHandling(string : userInput) : bool - CheckUserParams() : char - CheckPlaylistParams() : bool +PrintPlaylist(playListName : string) : void +PrintAllExistPlayLists() : void

AuxLib
+ SplitByDelim(string : str, char : delim) : vector<string> + ExtractWordBetween2Delim(string : word , char : delim1 , char : delim2)

Library
-m_library : map<string , Song>* -m_listOfPlayLists : map<string , PlayList*>* -m_regularPlayListNames : set<string>* -m_numberOfRegularPlayLists : int* -m_count : int -m_player : Player -m_options : unordered_map<string , char> -m_selections : vector<string> -m_singer : string -m_songName : string -m_genre : string -m_album : string -m_duration : string -m_bufferFlag : bool
+GetLibrary() : map<string , Song>* +GetCount() : int +SetCount(int) : void +Menu() : void -PrintSongByGivenCount(count : int) : void - ExtensionValidate(path : string) : bool - FindSongNameBySongId(songId : int) : string - DeleteSong(songName : string , songId : int) : void - AddSong(songName : string , path : string) : void - PrintSongDetails(songName : string , songId : int) : void - AddSongToPlaylist(songId : int , playListName : string) : void - RemoveSongFromPlaylist(songName : string, playListName : string) : void - PrintAllPlaylist() : void - PlaySong(songName : string , songId : int) : void - PlayAllSongs() : void - PlayAllSongsRandom() : void - Undelete(songName : string) : void -UpdateSongParams(string : input, int : songId, vector<string> UpdateVec) : void - CheckIfSongExist(string : songName) : bool - Help() const : void - CheckUserParams() : char - CheckAddParams() : bool - ClearSongParams() : void - CheckUpdateParams() : bool - checkPrintSongParams() : bool - CheckDeleteParams() : bool - CheckAdd2PlParams() : bool - CheckRemoveFromPlParams() : bool - CheckPlayParams() : bool - CheckUndeleteParams() : bool - BufferHandling(string userInput) : bool

Song
- m_songName : string - m_singerName : string - m_album : string - m_songPath : string - m_publishingYear : string - m_duration : string - m_genre : string - m_numberOfPlays : int - m_songId : int + SerialNumber : static int + NumberOfSongs : static int
+ GetNumberOfSongs() : int + GetSongName() : string + GetSingerName() : string + GetAlbum() : string + GetPublishingYear() : string + GetDuration() : string + GetSongPath() : string + GetGenre() : string + GetNumberOfPlays() : int + GetSongId() : int + SetGenre(genre : string) : void + SetSongName(songName : string) : void + SetSingerName(singerName : string) : void + SetAlbum(album : string) : void + SetPublishingYear(publishingYear : string) : void + SetDuration(duration : string) : void + SetSongPath(songPath : string) : void + SetNumberOfPlays(numberOfPlays : int) : void + SetSongId(songId : int) : void + IncreaseNumberOfPlays() : void

PlayList
m_player : Player
+ AddSong(songName : string , songObj : Song) : virtual void = 0 + DeleteSongBySongName(songName : string) : virtual void = 0 + DeleteAllSongs() : virtual void = 0 + Menu() : virtual void = 0 + Play() : virtual void = 0 + PlayRandom() : virtual void = 0 + Print() : virtual void = 0 + UpdateSongName(songName : string , updateName : string) : virtual void = 0 + CheckIfSongExist(songName : string) : virtual bool = 0 + GetPlaylistPath() : virtual string = 0 + GetSongObjectBySongName() : virtual Song = 0 + PlayMidMenu() : static char

Podcasts
- m_podcasts : set<string> - m_library : map <string , Song>* - m_listOfPlayLists : map<string , PlayList*>* - m_podcastsPath : string - m_options : unordered_map<string , char> - m_selections : vector<string> - m_bufferFlag : bool
+ Menu() : void + AddSong(podcastName : string , songObj : Song) : void + DeleteSongBySongName(podcastName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(podcastName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(podcastName : string) : Song - PlaySpecificPodcast(podcastName : string) : void - BufferHandling(string : userInput) : bool - Help() const : void - CheckUserParams() : char - CheckPlayParams() : bool - CheckDeleteParams() : bool

Favorite
- m_favorite : set<string> - m_library : map <string , Song>* - m_listOfPlayLists : map<string , PlayList*>* - m_favoritePath : string
+ Menu() : void + AddSong(songName : string , songObj : Song) : void + DeleteSongBySongName(songName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(songName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(songName : string) : Song

DailyMix
-m_dailyMix : set<string> - m_library : map <string , Song>* -m_listOfPlayLists : map<string , PlayList*>* - m_options : unordered_map<string , char> - m_dailyMixPath : string -m_day : int -m_month : int -m_year : int - m_bufferFlag : bool
+ Menu() : void + AddSong(songName : string , songObj : Song) : void + DeleteSongBySongName(songName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(songName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(songName : string) : Song - CheckIfDayChanged() : bool - SelectRandomSongs() : void - Help() const : void - CheckUserParams(string : input) : char - BufferHandling(string : userInput) : bool

MostPlayed
- m_mostPlayed : multimap<int , string , greater<int>> - m_library : map <string , Song>* - m_listOfPlayLists : map<string , PlayList*>* - m_mostPlayedPath : string
+ Menu() : void + AddSong(songName : string , songObj : Song) : void + DeleteSongBySongName(songName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(songName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(songName : string) : Song -UpdateMostPlayed() : void

Recent
- m_recent : list<string> - m_library : map <string , Song>* - m_listOfPlayLists : map<string , PlayList*>* - m_recentPath : string
+ Menu() : void + AddSong(songName : string , songObj : Song) : void + DeleteSongBySongName(songName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(songName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(songName : string) : Song

RegularPlayList
- m_regularPlayList : set<string> --m_library : map <string , Song>* -m_listOfPlayLists : map<string , PlayList*>* - m_regularPlayListPath : string -m_regularPlayListName : string
+ Menu() : void + AddSong(songName : string , songObj : Song) : void + DeleteSongBySongName(songName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(songName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(songName : string) : Song

Deleted
- m_deleted : map<string , Song> - m_library : map <string , Song>* - m_deletedPath : string
+ Menu() : void + AddSong(songName : string , songObj : Song) : void + DeleteSongBySongName(songName : string) : void + DeleteAllSongs() : void + UpdateSongName(songName : string , updateName : string) : void + CheckIfSongExist(songName : string) : bool +Play() : void +PlayRandom() : void +Print() : void + GetPlaylistPath() : string + GetSongObjectBySongName(songName : string) : Song

Search
- m_library : map<string , Song>* - m_listOfPlayLists : map<string , PlayList*>* - m_search : set<string> - m_history : list<string> - m_searchPath : string - m_player : Player - m_count : int - m_bufferFlag : bool - m_searchOptions : unordered_map<string , char> - m_deleteOptions : unordered_map<string , char> - m_playOptions : unordered_map<string , char> - m_selections : vector<string>
- SearchMenu() : void - PlaySearches() : void - PlayBySongName(songName : string) : void - PlayAll() : void - SearchBySongName(songName : string) : void - SearchBySingerName(songName : string) : void - SearchByAlbum(songName : string) : void - SearchByGenre(songName : string) : void - InsertToHistorySearch(search : string) : void - PrintHistoryByGivenCount(count : int) : void - DeleteMenu() : char - DeleteHistory() : void - ChekIfExsitInHistory(search : string) : bool - Help() const : void - DeleteHelp() const : void - PlayHelp() const : void - CheckUserParamsSearchMenue() : char - CheckUserParamsDeleteMenu() : char - CheckUserParamsPlayMenu() : char - CheckSearchParams() : void - CheckDeleteParams() : void - BufferHandling(string userInput) : bool

מקורות:

Stack overflow	Stack Overflow - Where Developers Learn, Share, & Build Careers
Cppreference	cppreference.com
GreeksforGeeks	GeeksforGeeks | A computer science portal for geeks