

תוכנה 1 – אביב תשע"ד

תרגיל מספר 10

GUI

הנחיות כלליות:

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

- הגשת התרגיל תעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש aviv יקרא הקובץ aviv_hw10.zip). קובץ ה-zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
 - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש, כולל תיקיות החבילה.
 - ג. קובץ PDF בשם answers.pdf המכיל את התשובות לשאלות.

חלק א' - MusicTunes (20 נק')

בחלק זה נכתוב מערכת לניהול רשימות אישיות של אלבומי מוסיקה, MusicTunes. באתר הקורס נתונים לכם שלדים של המחלקות הבאות בחבילה i1.ac.tau.cs.sw1.musictones.data. עליכם להשלים את מימוש המחלקות בהתאם לתיאור של כל אחת מהמתודות.

1. Album - מחלקה המתארת אלבום מוסיקה. לכל אלבום ניתן להוסיף רצועות מוסיקה (tracks) ואמנים שהשתתפו באלבום. אותו אמן לא יכול להופיע באלבום פעמיים, אבל רצועות מוסיקה עשויות לחזור על עצמן. המזהה הייחודי של אלבום הוא צירוף של שמו והשנה בו פורסם.
2. Artist - מחלקה המתארת אמן מוסיקלי. כל אמן מזהה ע"י שמו (יכול להיות שם של אדם, של להקה וכו').
3. Track - מחלקה המתארת רצועת שמע, שיש לה שם (מזהה ייחודי) וכן אורך בשניות.
4. MusicRepository - מחלקה המייצגת מאגר מוסיקלי שניתן להוסיף אליו אלבומים.

הערות

- ניתן להוסיף מחלקות, שדות, מתודות וכו' לפי הצורך.
- ניתן להוסיף ירושה ממחלקות ומימוש מנשקים לפי הצורך.
- אין לשנות את חתימות המתודות הנתונות לכם (אבל אפשר להוסיף העמסה לפי הצורך).
- **שימו לב**, אינכם יכולים לשנות את תנאי הקדם עבור המתודות הנתונות, ולכן עליכם לדאוג לטפל במקרי קצה בעת הצורך.
- בחלק מן המחלקות אולי יהיה צורך בדריסה של מתודות equals ו-hashCode, או במימוש המנשק Comparable או Comparator. כדאי להקדיש לכך מחשבה.
- המקומות שבהם נדרשת השלמת קוד מסומנים ב-`TODO`. תוכלו להיעזר ברשימת ה-tasks ב-Eclipse על מנת לראות את כל המקומות בקוד בהם מופיע הסימון הנ"ל. לשם כך, לחצו על Window > Show view > Tasks ב-Eclipse. לחיצה כפולה על אחת השורות ב-view שנפתח תוביל אתכם לנקודה הרלוונטית בקוד.
- היעזרו במחלקה MusicTunesDataSmallTest כדי לבדוק את עצמכם. מומלץ להוסיף בדיקות משלכם.

חלק ב' - GUI (50 נק')

כעת, נכתוב ממשק גרפי ל-MusicTunes אשר יתבסס על המחלקות שכתבתם בחלק א' של התרגיל. רוב הקוד הדרוש ליצירת רכיבי ה-GUI כבר נתון לכם, אך עליכם להשלים אותו כדי לגרום לתכנית לפעול בצורה הרצויה.

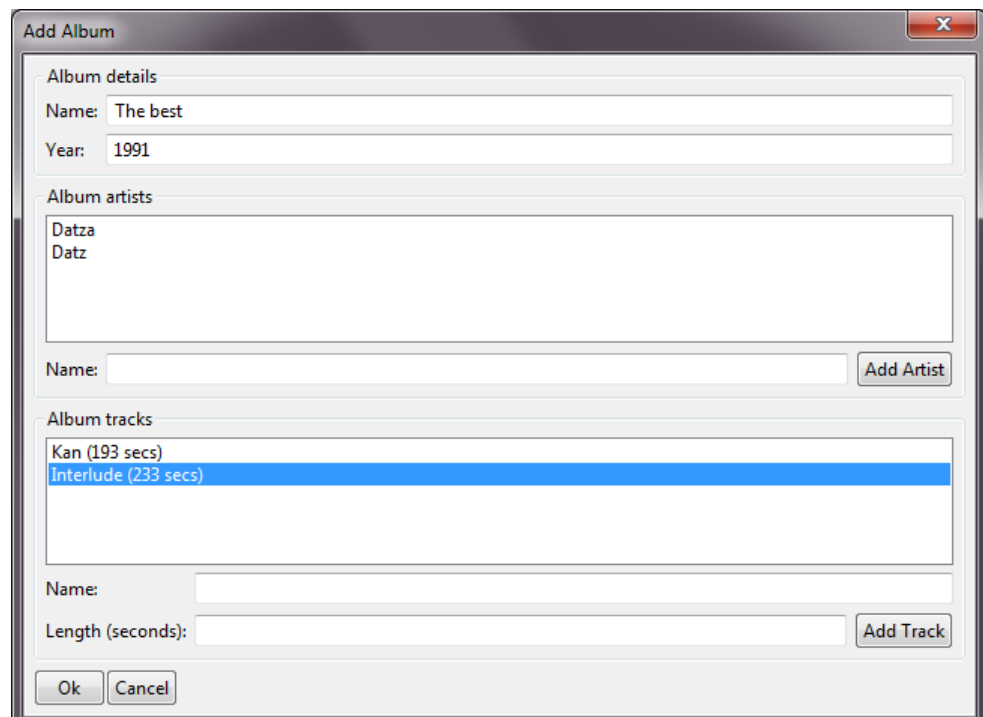
תיאור השימוש בממשק הגרפי

כדי לפתוח את הממשק הגרפי יש להריץ את התכנית GraphicalMusicTunesViewer. כעת, ייפתח החלון הראשי של התכנית.



בחלון זה מופיעים שני tab-ים: Albums - המציג את רשימת האלבומים של המשתמש הנוכחי, ותחתיהם את האמנים המשתתפים באלבום ואת רצועות המוסיקה. האלבומים ממוינים כברירת מחדל לפי שמם ואז לפי השנה שבה פורסמו. האמנים בכל אלבום ממוינים לפי שמם, ורצועות המוסיקה מוצגות לפי סדר הופעתן באלבום.

ניתן להוסיף אלבומים לרשימה ע"י לחיצה על הכפתור Add Album. ייפתח החלון הבא, המאפשר הכנסת פרטי אלבום (CreateAlbumDialog).



דרך לחיצה על Add Artist ניתן להוסיף אמנים לאלבום, ודרך לחיצה על Add Track ניתן להוסיף רצועות מוסיקה לאלבום. לחיצה על Cancel תסגור את החלון הנ"ל, והשינויים שבוצעו לא יישמרו. לחיצה על Ok תסגור את החלון הנ"ל, תוסיף את האלבום למאגר ותעדכן את הנתונים על אלבומים באמנים בחלון הראשי. נתונים אלה יוצגו בצורת עץ כך שתחת כל אלבום ב-tab האלבומים יוצגו האמנים המשתתפים בו ורצועות השמע, ותחת כל אמן ב-tab האמנים יוצגו האלבומים בהם השתתף.



ב- tab האלבומים: האלבומים ממוינים לפי שם ואז לפי שנה; האמנים בכל אלבום ממוינים לפי שם, ורצועות השמע לפי סדר הוספתן לאלבום. לחיצה על Sort by year תגרום למיון האלבומים קודם לפי שנה ואז לפי שם, וכן לשינוי הטקסט על הכפתור ל-Sort by name. לחיצה על Sort by name תחזיר את המיון והכפתור למצב הקודם.



ב- tab האמנים: האמנים ממוינים לפי שם, והאלבומים תחת כל אמן ממוינים לפי שם ואז לפי שנה.

לבסוף, הכפתור About בתפריט החלון הראשי יפתח הודעה למשתמש עם פרטי התכנית - מי כתב אותה (שמכם) ומתי (נניח, התאריך שבו סיימתם את התרגיל).



תכנית לדוגמא

נמצאת באתר הקורס. יש להוריד את הקובץ MusicTunes.jar ולשמור אותו בתיקיה כלשהי במחשב. באותה תיקיה, שמרו עותק של swt.jar המתאים למערכת ההפעלה שלכם. ניתן להוריד את ה-jar מ-<http://www.eclipse.org/swt>. מומלץ להוריד ע"י לחיצה על Releases > Stable > more... , ולחיצה על http בשורה המתאימה למע' ההפעלה שלכם. קובץ הזיפ שתורידו יכיל בתוכו את swt.jar, ויש להוציא אותו מן הזיפ ולשמור באותה תיקיה יחד עם MusicTunes.jar. כעת, כדי להריץ את התכנית לדוגמא מ-command line היכנסו לאותה תיקיה (ב-Windows למשתמשי Windows, terminal למשתמשי Linux וכו'), והקישו את הפקודה

```
java -jar MusicTunes.jar
```

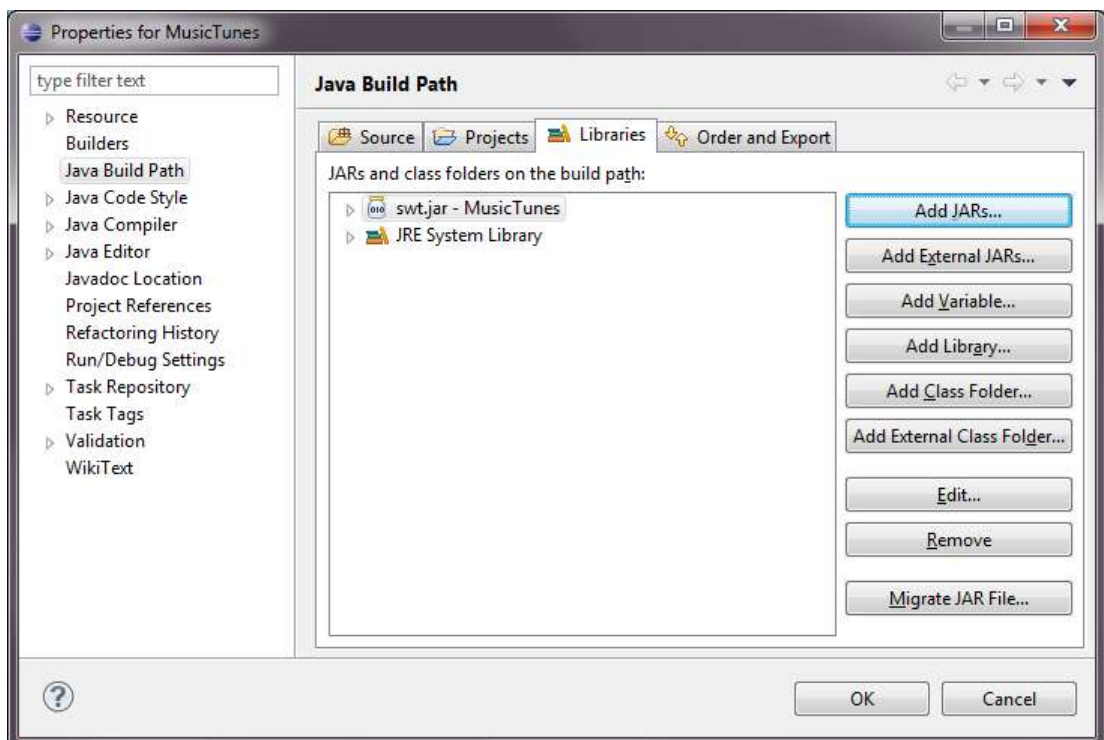
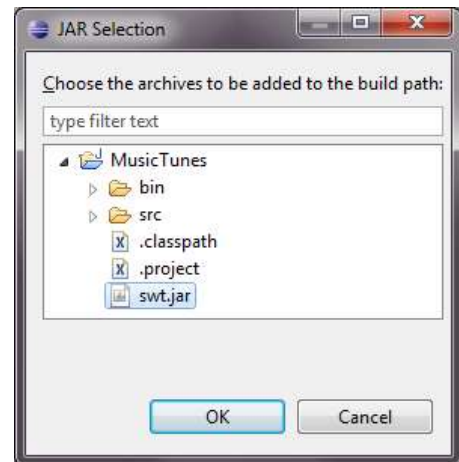
אם פעלתם לפי ההנחיות, ו-7 java מותקן על המחשב שלכם, פקודה זו תפתח את ממשק המשתמש של MusicTunes ותוכלו לשחק בו כדי לבחון את אופן פעולתו.

מה עליכם לעשות

באתר הקורס נתונים לכם שלדים של מחלקות ה-GUI בחבילה `il.ac.tau.cs.sw1.musictones.ui`. השלימו את המחלקות במקומות המסומנים ב-`TODO`, הוסיפו ושנו קוד לפי הצורך. יש להוסיף ל-`build path` של הפרויקט את `swt.jar` המתאים למערכת ההפעלה, כך:

העתיקו את `swt.jar` תחת הפרויקט שיצרתם עבור MusicTunes ב-Eclipse.

קליק ימני על הפרויקט < properties < Java build path < Libraries < Add JARs < והוסיפו את `swt.jar`



לבסוף, לחצו על OK לאישור שינויי ההגדרות.

הערות

- לצורך פתרון התרגיל, חשובה יכולת קריאה והבנה של הקוד הנתון. הקוד מתועד ברובו, אך ייתכן שתצטרכו להיעזר גם בתיעוד של SWT, בתיעוד של java ובאינטרנט כדי להבין חלקי קוד מסוימים.
- בחלק זה של התרגיל אתם רשאים לשנות את המחלקות והמתודות הנתונות באופן חופשי, מלבד:
 - פונקציה ה-main הראשית, שצריכה להימצא ב-GraphicalMusicTunesViewer,
 - עליכם להשתמש במחלקות מחלק א' של התרגיל.
- לפני תחילת פתרון התרגיל, מומלץ לעבור על כל הקוד, ובפרט על המחלקה GUIUtils. ייתכן שתמצאו שירותים שיעזרו לכם בפתרון התרגיל (מומלץ אבל אין חובה להשתמש בהם).
- התכנית שלכם לא חייבת להיות זהה לגמרי לתכנית לדוגמא, אך חשוב לשמור על הנק' הבאות:
 - **מראה ה-UI** (החלונות, הכפתורים וכו') -- אין לשנותו
 - **עדכון ה-UI** - ה-UI חייב להיות מעודכן אחרי כל פעולה, כמו בתכנית לדוגמא. למשל, הוספת אלבום צריכה לעדכן את רשימת האלבומים ואת רשימת האמנים בהתאם.
 - **טיפול בשגיאות:** בשום מקרה, אסור לתכנית "לעוף" או "להיתקע" כשהיא נתקלת בשגיאה שניתן להתאושש ממנה. במקרה של שגיאה יש להציג חלון עם הודעה למשתמש (אינפורמטיבית ככל האפשר), ולהחזיר את התכנית למצב תקין שממנו ניתן להמשיך לעבוד. לשם כך, יש להשתמש ב-try-catch כדי לתפוס שגיאות שעלולות להיזרק עקב פעולות שונות.
 - **טיפול בערכים לא תקינים:** יש לבדוק שהערכים שהוכנסו בשדות הטקסט השונים בתכנית תקינים. למשל, יש למנוע הכנסת שם ריק עבור אלבום, או מספר שניות שלילי עבור רצועת מוסיקה. לחילופין ניתן להציג הודעת שגיאה למשתמש.
 - **סדר:** יש להקפיד על הסדר שבו יוצגו האלבומים, האמנים ורצועות השמע בהתאם לתכנית לדוגמא ולהנחיות. יש לשים לב מתי מותרות כפילויות (duplicates) ומתי לא.

דוגמת שימוש ב-org.eclipse.swt.widgets.Tree

```
// create a tree
Tree tree = new Tree(parent, SWT.SINGLE | SWT.BORDER);

// add an item
TreeItem item = new TreeItem(tree, SWT.NULL);
item.setText("item");

// add an item under the previous item
TreeItem nestedItem = new TreeItem(item, SWT.NULL);
nestedItem.setText("item");
```

דוגמת שימוש ב-org.eclipse.swt.widgets.List

```
// create a list
List list = new List(parent, SWT.BORDER);

// add an item
list.add("list item");
```

חלק ג' - חידות ג'אוה - תיקון מחלקות (15 נק')

בחלק זה נתרגל, בעזרת תיקון של קוד קיים, נושאים מתקדמים הכוללים הבנה של אופן פעולתו של קוד נתון, פתרון שגיאות קומפילציה וריצה, binding דינאמי וסטטי, ירושה ועוד.

בקובץ ProgramCorrection.zip, הנמצא באתר הקורס, נתונות 3 מחלקות: SuperClass, OtherClass ו-SubClass. בחלק מן המחלקות, כפי שהן כתובות כעת, מופיעות שגיאות קומפילציה. השגיאות נובעות מבעיות בהגדרות של נראות, ירושה, חריגים, טיפוסים ועוד. עבור כל שגיאה עליכם לתקן את הקוד כך שבסופו של דבר:

1. הקוד יתקמפל ללא שגיאות (ייתכנו אזהרות קומפילציה בלבד)
2. פונק' ה-main המוגדרת ב-SubClass תרוץ ללא שגיאות ותדפיס את המספר 491-.

הנחיות

- אין לשנות את תוכן פונק' ה-main ב-SubClass (אפשר רק להוסיף/לשנות/להוריד הצהרת throws).
- עבור כל שגיאת קומפילציה, בצעו תיקון מינימלי כדי לתקן אותה. נסו תחילה לחשוב בעצמכם ממה נגרמת השגיאה ומה התיקון הדרוש, ואז בדקו שהתיקון שלכם אכן פותר את הבעיה. שימו לב, ייתכנו מס' תיקונים מינימליים אפשריים עבור חלק מן השגיאות, ולכן קיים יותר מפתרון אחד.
- **צעדים אפשריים:** הוספה, הורדה ושינוי של
 - Imports
 - ה-modifiers הבאים: (private, protected, public, visibility, static, final, abstract)
 - ארגומנטים למתודות
 - הצהרה על זריקת חריג throws
 - casting של משתנים
 - super.-I this לפני קריאות למתודות \ שדות (למשל (this.myMethod()
- בקובץ התשובות answers.pdf כתבו **במילים שלכם ובקצרה ככל האפשר**, בשורה נפרדת עבור כל שגיאה:
 - באיזו שגיאה מדובר ובאיזה קובץ (לדוגמא, שגיאה בהשמה לשדה X במחלקה Y במתודה Z)
 - מה גרם לשגיאה (לדוגמא, X מוגדר כ-final ולכן לא ניתן לשנות את ערכו)
 - כיצד תיקנתם את השגיאה (לדוגמא, הסרתי את ה-final modifier מ-X)
- תארו במילים שלכם ובקצרה ככל האפשר מה התכנית עושה לאחר התיקון. (לדוגמא, התכנית קוראת למתודה Z ששמה ב-X את תוצאת החישוב W-2000, ואז מדפיסה את X).
- יש להגיש את שלוש המחלקות המתוקנות, בתוך ספריות החבילה.

חלק ד' - שיפור סגנון קידוד (Java Coding Style) (15 נק')

בשאלה זו נעבוד על שיפור סגנון הקידוד שלכם, בעזרת שיפור פתרון של תרגיל של חברה שלכם לקורס. על כל אחד מכם לבקש מחבר או חברה לקורס את **הפתרון שהם הגישו לבדיקה** עבור תרגיל 6, חלק א' - כלומר, המחלקה EncDec.java. עליכם יהיה לשפר את סגנון הקידוד של המחלקה ולהגיש **שתי מחלקות**:

- את EncDec המקורית שקיבלתם מהחברה
- את המחלקה EncDecCorrected, אשר תכיל את השיפור שלכם. על המחלקה המשופרת להיות בעל סגנון קידוד מושלם.
- בנוסף, הסבירו **במילים שלכם ובקצרה ככל האפשר** מהם 5 התיקונים השונים המשמעותיים ביותר שביצעתם ומדוע הם חשובים (להגיש בקובץ התשובות).

שימו לב, אין חובה לתקן את אופן פעולת המחלקה, אלא רק לשפר את הקידוד כך שיהיה:

- (א) כתוב לפי המוסכמות לקידוד בג'אווה
- (ב) קריא וברור
- (ג) מודולרי, כלומר, מחולק למתודות ומתודות עזר, שלכל אחת מהם תפקיד ברור, ומאפשרות שימוש חוזר.

היעזרו בקישור הבא, המתאר את מוסכמות הקידוד בג'אווה:

<http://www.oracle.com/technetwork/java/codeconv-138413.html>

שימו לב במיוחד לנקודות הבאות (שעלו מניתוח מדגמי של תרגילים שהגישו סטודנטים בקורס):

- **שמות משמעותיים** למחלקות, משתנים ומתודות
 - שם המשתנה אמור לציין את תפקידו ואת התוכן שלו. לא ראשי תיבות, לא מילים בעברית, לא אותיות בודדות (אלא אם כן זה משתנה אינדקס למשל בלולאת For ואז מקובל להשתמש ב-i,j,k)
 - שמות של משתנים בוליאניים יהיו מהצורה isVisible, isValid
 - שמות מחלקות יתחילו באות גדולה, שמות משתנים ומתודות באות קטנה, ולאחר מכן תופיע אות גדולה בתחילת כל מילה (למשל studentAverage).
 - שמות קבועים יהיו מהצורה COURSE_ID.
- שם המתודה יהיה לרוב פועל, ויציין את הפונקציה אותה היא מבצעת.
- **מודולריות ומניעת שכפול קוד** – ארגון הקוד במתודות קצרות יחסית כאשר כל מתודה מבצעת פונקציה מוגדרת. ניתן להוסיף מתודות ומחלקות עזר לפי הצורך.
- **תיעוד** (באנגלית) – יש לכתוב מעל כל מתודה ושדה מה תפקידם, ובעת הצורך להוסיף הערות אשר מסבירות את פעולתם של קטעי קוד (למשל, "הלולאה הבאה עוברת על כל הבתים ברשימה ומפעילה עליהם את פונק' ההצפנה"). עם זאת, אל תכתבו את המובן מאליו ("השורה הבאה סוכמת את x ו-y").
- **שימוש בקבועים** לאחסון ערכים, במיוחד כאשר יש בהם שימוש חוזר במס' מקומות בקוד.
- **טווח ההכרה של המשתנים** אמור להיות מינימלי – משתנים אמורים להיות כמה שיותר מקומיים.
- במקרים רבים, שיפור הקוד גורם לכך שהקוד קריא בפני עצמו, ואז אין צורך בהוספת הערה.
- **הרשאות נראות** (private, public) של שדות צריכות להיות מינימליות, וגישה פומבית לשדות (במידה והיא נדרשת) פרטיים צריכה להיעשות ע"י Setters ו-Getters.
- בכל שורה תופיע פקודה אחת. בכל שורה יופיע רק סוגר מסולסל יחיד.
- **אינדנטציה** (הזחה).

הערות:

- הקוד המוגש חייב להתקמפל ללא שגיאות. גם אם אינכם מתקנים שגיאות קיימות באופן פעולת הקוד, הימנעו מהוספת שגיאות חדשות.
- אין לשנות את כותרות המתודות שהיו נתונות בתרגיל 6.
- אם אתם מתקשים למצוא מישהו לבקש ממנו את הקוד, מותר גם לשפר את הקוד של עצמכם (אם כי האפשרות הזאת מומלצת פחות).
- ברוב המקרים, גם בקוד של מתכנתים קפדנים ניתן למצוא דרכים לשיפור. אם בכל זאת אינכם מוצאים שום שיפור אפשרי בקוד שקיבלתם, בקשו קוד ממישהו אחר או תארו והדגימו (בקצרה) 5 מאפיינים מרכזיים של הקוד המקורי שהופכים אותו, לדעתכם, למושלם. בכל מקרה יש להגיש את EncDecCorrected.
- למען הסר ספק, לא ירדו נק' לסטודנטים על תרגיל 6 בדיעבד, אם מצאתם בעיות בקוד שלהם.
- מומלץ לתת משוב לחברה על הקוד שהם כתבו, מה היה טוב ומה טעון שיפור.

בהצלחה!