הנדסת תוכנה – תרגיל בית 3

דגשים להגשת המטלה

1. תאריך הגשה: יום רביעי 21.06.2023 בשעה 23:59.
2. **הגשה בזוגות בלבד!**
3. הקוד חייב להיכתב בהתאם למוסכמות כתיבת הקוד בקורס כולל תיעוד כנדרש. קוד שלא עומד בדרישות יגרור הורדת ניקוד. ניתן למצוא את קובץ מוסכמות הקידוד באתר הקורס תחת הלשונית "קבצי עזר".
4. ההגשה מתבצעת ב-Moodle באזור המיועד על ידי אחד מהשותפים, לאחר יצירת קבוצה.
5. ניתן להגיש את התרגיל לכל היותר עד 48 שעות לאחר מועד ההגשה ללא הורדת ניקוד. לאחר 48 שעות תיבת ההגשה תיסגר ולא יהיה ניתן להגיש את התרגיל כלל.
6. פורמט הגשת התרגיל נמצא בקובץ ההנחיות ב-Moodle. **כל חריגה מפורמט זה תגרור ציון אפס.**

מטרת התרגיל

עבודה עם מבני נתונים, חריגות, העתקת עצמים ואיטרטורים.

הכנות טרם תחילת התרגיל

1. פתיחת פרויקט ג'אווה חדש.
2. הורדת קבצי התרגיל, והעתקת הקבצים Main.java ו-Stack.java **בלבד** אל תוך תיקיית ה-src.

הוראות כלליות

1. מומלץ להריץ את התוכנית עם מספר קלטים שונים ולחשוב על מקרי קצה אפשריים.
2. מומלץ לחזור על התרגולים וההרצאות וכן להיעזר באינטרנט.
3. יש להשתמש בגרסה 9.0.4 של ג'אווה בעת פתרון התרגיל.
4. מומלץ להשתמש ב-Git במהלך כתיבת התרגיל.

הוראות הגשה

1. יש למלא אחר הוראות ההגשה בהתאם למסמך הדרישות "הנחיות כלליות לפתרון והגשת תרגילי הבית" אשר מופיע באתר הקורס.
2. הגשה אלקטרונית **בלבד** דרך אתר הקורס ב-moodle. ההגשה תכלול קובץ ה-zip בלבד בפורמט HW3_<id1>_<id2> כאשר <id1> ו-<id2> הם תעודות הזהות של המגישים. פירוט נוסף נמצא בסוף המסמך.
3. ההגשה מתבצעת על ידי אחד מבני הזוג לאחר שיצר קבוצה ובן הזוג השני הצטרף אליה.
4. תרגיל בית שלא יוגש על פי הוראות ההגשה – **לא ייבדק**.
5. יש להקפיד על יושרת הכנת התרגיל וההגשה.
6. יש לוודא כי הקוד מתקמפל – קוד אשר לא יעבור הידור יקבל ציון אפס.
7. אין צורך להגיש את קובץ הפלט אשר ניתן כחלק מתרגיל זה.

חלק א – מבני נתונים

בהרצאות ובתרגולים דנו במספר מבני נתונים. ראינו כיצד מבני הנתונים השונים ממומשים בג'אווה, ואף מימשנו את חלקם בעצמנו.

בחלק זה של התרגיל תממשו מבנה נתונים אותו פגשנו – מחסנית.

מחסנית הינה מבנה נתונים אשר עובד בשיטת LIFO: Last In First Out. כלומר – האיבר האחרון שנכנס למחסנית יהיה הראשון לצאת ממנה. כאשר איבר מוכנס למחסנית, הוא נכנס אל ראש המחסנית (head), וכאשר איבר מוצא מן המחסנית הוא יוצא מראשה. האיבר האחרון במחסנית (זה שהוכנס ראשון אליה) נמצא בזנב המחסנית.

אחד מן הקבצים שניתנו כחלק מהתרגיל הוא קובץ בשם Stack.java. בקובץ זה מוגדר ממשק המייצג מחסנית כללית. בממשק מוגדרות מספר פעולות שכל מחסנית צריכה לממש:

- `push`: פעולה אשר מקבלת איבר ומוסיפה אותו למחסנית.
- `pop`: פעולה אשר מוציאה איבר מראש המחסנית ומחזירה אותו.
- `peek`: פעולה אשר מחזירה את האיבר שנמצא בראש המחסנית, מבלי להוציאו.
- `size`: פעולה אשר מחזירה את מספר האיברים במחסנית.
- `isEmpty`: פעולה אשר בודקת האם יש איברים במחסנית ומחזירה את התוצאה.
- `clone`: פעולה אשר מבצעת העתקה עמוקה של המחסנית.

פרטים נוספים לגבי הממשק:

- הממשק הינו גנרי, והטיפוס הגנרי מוגבל מלמעלה על ידי הממשק `Cloneable`.
- ממשק זה מרחיב את הממשק `Iterable` על מנת לאפשר מעבר על איברי המחסנית.
- ממשק זה מרחיב את הממשק `Cloneable` על מנת לאפשר העתקה עמוקה של המחסנית.

עליכם ליצור מחלקה גנרית בשם `ArrayStack` אשר תממש את הממשק `Stack` שמוגדר בקובץ שקיבלתם.

המחסנית אותה תממשו תיעזר במערך לצורך שמירת הנתונים. בנוסף, למחסנית יש קיבולת מקסימלית המתקבלת כפרמטר בבנאי המחלקה. במידה והקיבולת שהתקבלה שלילית, תיזרק חריגה בלתי מסומנת בשם `NegativeCapacityException`. במידה ומנסים להוסיף למחסנית איבר כאשר המחסנית הגיעה לקיבולתה המקסימלית, תיזרק חריגה בלתי מסומנת בשם `StackOverflowException`. בנוסף, במידה והמחסנית ריקה ומנסים להוציא ממנה איבר או להציץ באיבר שנמצא בראשה, תיזרק חריגה בלתי מסומנת בשם `EmptyStackException`. עליכם להגדיר את שלוש החריגות ביחד עם מחלקת חריגה נוספת בשם `StackException` אשר שלוש המחלקות ירשו ממנה. שימו לב: בג'אווה קיימת מחלקה בשם `EmptyStackException`, אך אין להיעזר בה ויש ליצור מחלקה חדשה בשם זה.

נוסף על כך, על מנת לשמור על יעילותה של המחסנית, עליכם לממש את פעולות ההכנסה וההוצאה מן המחסנית באופן שלא ידרוש הזזה של איברים הנמצאים במערך. בפרט, בפעולות ההכנסה וההוצאה מן המחסנית אין להשתמש בלולאות וכן אין להשתמש בפעולות המשתמשות בלולאות.

העתקת המחסנית

על המחסנית לתמוך בפעולה שמבצעת העתקה עמוקה של המחסנית. כדי להבטיח כי ניתן יהיה להעתיק את האיברים שהוכנסו למחסנית, הטיפוס הגנרי של המחסנית מוגבל מלמעלה על ידי הממשק `Cloneable`.

עליכם לממש במחלקה `ArrayStack` את הפעולה `clone` תוך שימוש ב-Covariant Return Type. יש להשתמש בבלוק `try-catch` בעת כתיבת פעולת ה-`clone`. במידה ונזרקת חריגה בעת ההעתקה יש להחזיר את ערך ברירת המחדל `null`. אין לזרוק אף חריגה בפעולת ה-`clone`.

הכוונה: תוכלו להיעזר בפעולות `getMethod` ו-`invoke` לצורך העתקת איברי המחסנית.

מעבר על איברי המחסנית

על המחסנית אותה תממשו לתמוך במעבר על איבריה באמצעות לולאת foreach. לשם כך, עליכם להגדיר במחלקת המחסנית מחלקה פנימית בשם `StackIterator` שמממשת את הממשק הגנרי `Iterator` ומייצגת איטרטור שישמש לצורך מעבר על איברי המחסנית. המעבר על האיברים יתבצע לפי סדר הופעתם במחסנית, החל מהאיבר הנמצא בראש המחסנית ועד לאיבר שנמצא בזנב. בעת מעבר על איברי המחסנית אין לשנות את המחסנית כלל, גם לא באופן זמני.

הנחיות לחלק א

- פרט למחלקות אותן התבקשתם לכתוב אין להיעזר במחלקות נוספות, גם אם תגדירו אותן בעצמכם. בפרט, אין להשתמש באף מבנה נתונים דינמי המוגדר בג'אווה.
- ניתן לייבא את מחלקת החריגה `InvocationTargetException` וכן את הממשקים `Iterator` ו-`Iterable` וכן ניתן לתפוס (בשמן) את החריגות שעלולות להיזרק מפעולת ה-`getMethod` וה-`invoke`.
- אין לשנות את **כלל** הממשק `Stack` אותו קיבלתם ואין לשנות את הפעולות המוגדרות בו.
- שימו לב כי לא ניתן להניח דבר על הטיפוסים שנשמרים במחסנית, פרט לכך שהם ניתנים להעתקה. כחלק מהבדיקה של המחלקה אותה הגדרתם, מוגדרת בקובץ Main.java מחלקה בשם `MyCloneable`. מחלקה זו נוצרה אך ורק על מנת שתוכלו לבדוק את הקוד שלכם, וייתכן מאוד כי במהלך בדיקת התרגיל מחלקה זו תוחלף במחלקה אחרת. על כן, בקוד אותו אתם כותבים **אין להשתמש כלל** במחלקה `MyCloneable`.

חלק ב – רשימת השמעה

תיאור המשימה

בעקבות שביתת הסגל באוניברסיטאות, לסטודנטים רבים התפנה זמן אותו הם רוצים להקדיש לשמיעת מוזיקה. על מנת שהסטודנטים יוכלו לשמוע את השירים האהובים עליהם בקלות, יושב ראש ועד הסטודנטים ביקש מכם לבנות מערכת לניהול רשימות השמעה.

יושב הראש סיפר לכם שכל רשימת השמעה מורכבת ממספר לא מוגבל של שירים, וכן כי כל שיר מכיל מידע על שם השיר, שם מחבר השיר, הסוגה (ז'אנר) של השיר ומשך השיר בשניות. עוד הוא הוסיף כי לאחר יצירת שיר לא ניתן לשנות את שמו ואת שם המחבר שלו.

לצורך נוחות, יושב הראש ביקש מכם שתהיה באפשרות הסטודנטים היכולת להציג את רשימת ההשמעה שלהם באופן הבא:

[(name, artist, genre, duration), …, (name, artist, genre, duration)]

כאשר name הינו השם של השיר, artist הינו שמו של מחבר השיר, genre הינו הסוגה של השיר ו-duration הינו משך השיר בפורמט mm:ss. במידת הצורך, יש להוסיף 0 מוביל לשניות בלבד.

יתרה מכך, יושב הראש ביקש לאפשר לסטודנטים לסרוק את רשימת ההשמעה שלהם על ידי שימוש במגוון פילטרים:

- מעבר רק על שירים של מחבר מסוים.
- מעבר רק על שירים מסוגה מסוימת.
- מעבר רק על שירים שמשכם לא עולה על משך זמן מסוים.

בכל מעבר על רשימת השמעה נעבור רק על השירים אשר עומדים בכל התנאים, לפי אחד מסוגי הסריקה הבאים:

- מעבר על השירים על פי סדר הוספתם לרשימת ההשמעה.
- מעבר על השירים לפי סדר אלפביתי של שמותיהם. במידה וישנם שירים בעלי אותו שם, המעבר עליהם יתבצע לפי סדר אלפביתי של שמות המחברים.
- מעבר על השירים לפי משכם. במידה וישנם שירים בעלי משך זהה, המעבר עליהם יתבצע לפי סדר אלפביתי של שמם. במידה ויש שירים שגם משכם וגם שמם שווים, המעבר יתבצע לפי שמות המחברים.

בנוסף, כדי שסטודנטים יוכלו לשלוח לחבריהם את רשימת ההשמעה שלהם, יושב הראש ביקש שהסטודנטים יוכלו לבצע העתקה עמוקה של רשימת ההשמעה שלהם, תוך שימוש ב-Covariant Return Type על מנת לחסוך לסטודנטים את הצורך להמיר את רשימת ההשמעה בעת העתקתה.

נוסף על כך, יושב הראש רוצה לאפשר לסטודנטים לבדוק האם רשימת ההשמעה שלהם ושל חבריהם שוות אחת לשנייה. לצורך כך, הוא הגדיר כי שתי רשימות השוואה שוות זו לזו אם ורק אם הן מכילות את אותם השירים, ללא חשיבות לסדר ההופעה שלהם. על פי יושב הראש, שני שירים שווים זה לזה אם ורק אם שמותיהם ושמות המחברים שלהם זהים (בהתאמה).

מימוש המשימה

עליכם להיענות לכל בקשותיו של יושב ראש אגודת הסטודנטים, ולהכין בעבורו את מערכת השירים על מנת שיוכל להפיצה לשימוש בקרב כל הסטודנטים בטכניון כבר במהלך הסמסטר הנוכחי.

לשם כך, התחילו בלהגדיר מחלקה בשם Song המייצגת שיר ומחלקה בשם Playlist המייצגת רשימת השמעה. בכל מחלקה יש לכלול את התכונות הנדרשות לה לפי התיאור של יושב הראש. במחלקה Song יש להגדיר enum פומבי בשם Genre המייצג סוגה. הסוגות האפשריות הן פופ (POP), רוק (ROCK), היפ-הופ (HIP_HOP), קאנטרי (COUNTRY), ג'אז (JAZZ) ודיסקו (DISCO).

במחלקה Playlist יש להוסיף פעולה בשם addSong המקבלת שיר ומוסיפה אותו לרשימת ההשמעה במידה ולא קיים בה שיר נוסף הזהה לשיר שהתקבל. במידה וכן קיים שיר זהה יש לזרוק חריגה בשם SongAlreadyExistsException. חריגה זו הינה חריגה בלתי מסומנת אותה עליכם להגדיר.

בנוסף, יש להגדיר במחלקה פעולה בוליאנית בשם removeSong המקבלת שיר ומסירה אותו (או שיר הזהה לו) מרשימת ההשמעה. במידה וההסרה הצליחה הפעולה תחזיר אמת, ואחרת תחזיר שקר.

הצגת רשימת השמעה

לצורך הצגת רשימת השמעה, עליכם לדרוס את הפעולה toString במחלקות Song ו-Playlist כך שתתקבל הצגה של כלל השירים ברשימה בפורמט אותו יושב הראש הגדיר. שימו לב שהשירים מוצגים לפי סדר ההוספה שלהם.

העתקת רשימת השמעה

על מנת לאפשר העתקה עמוקה של רשימת השמעה, עליכם לממש במחלקות Song ו-Playlist את הממשק Cloneable ולדרוס בהן את הפעולה clone, תוך שימוש ב-Covariant Return Type בשתי המחלקות. יש להשתמש בבלוק try-catch בעת כתיבת פעולות ה-clone. במידה ונזרקת חריגה בעת ההעתקה יש להחזיר את ערך ברירת המחדל null. אין לזרוק חריגה בפעולות ה-clone.

הערה: שימו לב בכל מחלקה לתכונות הניתנות לשינוי (mutable).

שוויון רשימות השמעה

כדי לאפשר השוואה בין שתי רשימות השמעה יש לדרוס את הפעולה equals במחלקות Song ו-Playlist כך שיתקיים יחס השוויון אותו יושב הראש הגדיר. בנוסף, יש להקפיד על שאר הדרישות אשר קיימות עבור יחס שוויון ופעולת ה-equals.

נוסף על כך, עליכם לדרוס בשתי המחלקות את פעולת ה-hashCode בהתאם לדרישות המופיעות במפרט הפעולה. בעת דריסת הפעולה אין להחזיר ערך קבוע: ערך הגיבוב של כל עצם צריך להיות תלוי בערכי התכונות של העצם.

מעבר על רשימת השמעה

יושב הראש ביקש שסטודנטים יוכלו לעבור על רשימת ההשמעה שלהם בצורות סריקה שונות וכן להשתמש במגוון פילטרים לצורך סינון שירים. לשם כך נרצה להגדיר איטרטור שבאמצעותו נבצע את הסריקה. עליכם ליצור במחלקה Playlist מחלקה **פנימית** בשם PlaylistIterator אשר מממשת את הממשק Iterator<Song>. יש לממש במחלקה זו את הפעולות hasNext ו-next.

לאחר מכן, עליכם ליצור ממשק בשם FilteredSongIterable אשר מרחיב את הממשק Iterable<Song>. בממשק זה יש להגדיר את הפעולות הבאות:

- פעולה בשם filterArtist המקבלת שם של מחבר. בעת מעבר על רשימת השמעה, המעבר יתבצע רק על שירים שחוברו על ידי המחבר שהתקבל. במידה ומועבר לפעולה הערך null, המעבר יתבצע על כל השירים (שעומדים בשאר תנאי הסריקה), ללא תלות בשם המחבר שלהם.
- פעולה בשם filterGenre המקבלת סוגה. בעת מעבר על רשימת השמעה, המעבר יתבצע רק על שירים שהסוגה שלהם שווה לסוגה שהתקבלה. במידה ומועבר לפעולה הערך null, המעבר יתבצע על כל השירים (שעומדים בשאר תנאי הסריקה), ללא תלות בסוגה שלהם.
- פעולה בשם filterDuration המקבלת את המשך המקסימלי של השירים. בעת מעבר על רשימת השמעה, המעבר יתבצע רק על שירים שמשכם קטן או שווה למשך שהתקבל.

אחר כך, עליכם ליצור enum בשם ScanningOrder המייצג את סדר המעבר על רשימת ההשמעה. בהתאם לבקשתו של יושב הראש, ניתן לעבור על השירים לפי סדר הוספתם לרשימת ההשמעה, לפי שמם ולפי משכם, ולכן הערכים של ה-enum הינם ADDING, NAME ו-DURATION. לאחר מכן, הגדירו ממשק בשם OrderedSongIterable אשר גם הוא מרחיב את הממשק Iterable<Song>. בממשק זה יש להגדיר פעולה בשם setScanningOrder המקבלת את סדר המעבר על רשימת ההשמעה.

לבסוף, ממשו במחלקה Playist את שני הממשקים אותם הגדרתם על מנת לאפשר סריקה נוחה של רשימת השמעה על ידי לולאת foreach. שימו לב כי מימוש הממשקים כולל מימוש של הפעולות המוגדרות בהם.

<u>הכוונות:</u>

- על מנת למיין רשימה לפי פעולה מותאמת אישית ניתן להיעזר בפעולה Comparator.comparing.
- על מנת לבצע מיון משני של רשימה לפי פעולה מותאמת אישית ניתן להיעזר בפעולה thenComparing.

<u>הנחיות לפתרון</u>

- בעת פתרון התרגיל ניתן להגדיר מחלקות נוספות.
- בעת פתרון התרגיל ניתן ואף מומלץ להגדיר קבועים ולא להשתמש במספרי קסם.
- במהלך פתרון התרגיל, יש להקפיד על הרשאות הגישה השונות בהתאם לנלמד בקורס.
- יש להקפיד על מתן שמות משמעותיים למשתנים, לפעולות, למחלקות ולממשקים.
- בכל מחלקה יש לכלול את התכונות המתאימות לה, ולספק בעבורן פעולות get ו-set **במידת הצורך בלבד**.
- בכל מחלקת חריגה אותה אתם יוצרים יש להגדיר את **שלושת** הבנאים הסטנדרטיים עליהם דיברנו בתרגולים.
- במהלך כתיבת הקוד, יש לשמור על עקרונות תכנות נכונים, כפי שנלמד בקורס.
- יש ליצור כל מחלקה, ממשק ו-enum בקובץ נפרד, פרט למחלקות הפנימיות ול-enum הפנימי. בנוסף, יש להגדיר את הרשאת הגישה של כלל המחלקות, הממשקים וה-enum-ים כפומבית.
- כל עוד לא הוגדר אחרת, המעבר על רשימת השמעה תתבצע על כל השירים, לפי סדר הוספתם.
- **בכל** דריסה ומימוש של פעולה יש להשתמש באנוטציה @Override על מנת לוודא שאכן מתבצעת דריסה.
- **יש להשתמש ב-Covariant Return Type במקומות המתאימים.**
- יש לתעד את כל הפעולות והמחלקות אותן אתם מגדירים בעזרת שימוש ב-JavaDoc בהתאם לקובץ מוסכמות התיעוד אשר מופיע באתר הקורס. בנוסף, יש לתעד שורות קוד אשר עשויות להיות קשות להבנה. עבור פעולות שזורקות חריגה כלשהי יש לציין זאת **בתיעוד** על ידי פסקת @throws ביחד עם הסבר על מתי החריגה תיזרק (שימו לב כי פסקת @throws שונה מציון החריגה בכותרת הפעולה).

<u>הגשת התרגיל</u>

לפני הגשת התרגיל, עליכם ליצור קבצי html מתוך תיעוד ה-JavaDoc שכתבתם בקוד. ניתן ליצור את הקבצים בעזרת ה-IntelliJ על ידי לחיצה על כרטיסיית ה-Tools ובחירת האפשרות Generate JavaDoc. בעת יצירת המסמכים, יש לבחור את הרשאת הגישה private על מנת שכל המחלקות והפעולות (משני חלקי התרגיל) יופיעו במסמכים. את כל הקבצים שנוצרים יש לשמור בתיקייה בשם JavaDoc.

כפי שהוזכר, קובץ ההגשה הסופי הינו קובץ zip בודד. שם הקובץ מפורט בתחילת המסמך ובמסמך ההוראות הכללי. קובץ ה-zip יכלול שתי תיקיות - תיקייה בשם src בה יימצאו כל קבצי הקוד אותם כתבתם עבור שני חלקי התרגיל ותיקייה נוספת בשם JavaDoc בה יימצאו כל קבצי התיעוד.

הרצת התוכנית וביצוע בדיקות

בקובץ Main.java קיימות מספר מחלקות אשר משמשות לבדיקת הקוד שכתבתם.

להזכירכם, חלק מן הבדיקה נעשה באופן אוטומטי, ולכן אין לשנות את התוכן של הקובץ, ובפרט אין לשנות את פעולות ההדפסה המתבצעות בו.

מצורף לתרגיל זה קובץ הפלט HW3_output.txt, על מנת שתוכלו לבצע את ההשוואה באופן ידני (או על ידי שימוש ב-DiffMerge).

בהצלחה