הנחיות לפתרון תרגילי הבית

- על הקוד המוגש להיות מתועד היטב ועליו לכלול:
  - מפרט, כפי שהודגם בתרגול.
  - תיעוד של כל מחלקה ומתודה ושל קטעי קוד רלוונטיים.
  - במידת הצורך, יש להוסיף תיעוד חיצוני.
- יש להפעיל את הכלי Javadoc כדי ליצור קבצי תיעוד בפורמט HTML ולצרף אותם לפתרון הממוחשב המוגש. כדי לגרום לקובצי ה-HTML להכיל את פסקאות המפרט שבהן אנו משתמשים, יש לציין זאת במפורש. ב-Eclipse, ניתן לבצע פעולה זו באופן הבא: 1. לבחור Export מתפריט File, לבחור Java->Javadoc וללחוץ על כפתור Next, 2. לבחור עבור Javadoc command את הקובץ javadoc.exe מתוך התיקייה bin הנמצאת בתיקייה שבה מותקן ה-Java SDK, 3. לבחור את הקבצים שלהם מעוניינים ליצור תיעוד וללחוץ פעמיים על כפתור Next, 4. להקיש ב-Extra Javadoc options את השורה הבאה וללחוץ על כפתור Finish:

-tag requires:a:"Requires:" -tag modifies:a:"Modifies:" -tag effects:a:"Effects:"

- התנהגות ברירת המחדל של פעולות assert היא disabled (הבדיקות לא מתבצעות). כדי לאפשר את הידור וביצוע פעולות assert, יש לבצע ב-Eclipse את הפעולות הבאות: 1. מתפריט Run לבחור Debug Configurations, 2. בחלון שנפתח, לעבור ללשונית Arguments, 3. בתיבת הטקסט VM arguments לכתוב ea–, 4. ללחוץ על כפתור Debug.

הנחיות להגשת תרגילי בית

- תרגילי הבית הם חובה.
- ההגשה בזוגות בלבד.
- עם סיום פתירת התרגיל, יש ליצור קובץ דחוס להגשה המכיל את:
  - כל קבצי הקוד והתיעוד.
  - פתרון לשאלות ה"יבשות" בקובץ Word או PDF. על הקובץ להכיל את שמות ומספרי תעודות הזהות **של שני הסטודנטים** המגישים.
- הגשת התרגיל היא **אלקטרונית בלבד**, דרך אתר הקורס **ע"י אחד מבני הזוג בלבד**.

הקובץ המוגש יקרא hw0_<id1>_<id2>.zip כאשר <id1> ו-<id2> הם מספרי הזהות של הסטודנטים המגישים. לדוגמא hw1_12345678_9876541.zip (כמובן יש להשתמש במספרי הזהות שלכם)

- תרגיל שיוגש באיחור וללא אישור מתאים (כגון, אישור מילואים), יורד ממנו ציון באופן אוטומטי לפי חישוב של 5 נקודות לכל יום איחור ועד 2 ימי איחור שלאחריהם לא תתאפשר ההגשה כלל.

- על התוכנית לעבור הידור (קומפילציה). על תכנית שלא עוברת הידור יורדו 30 נקודות.

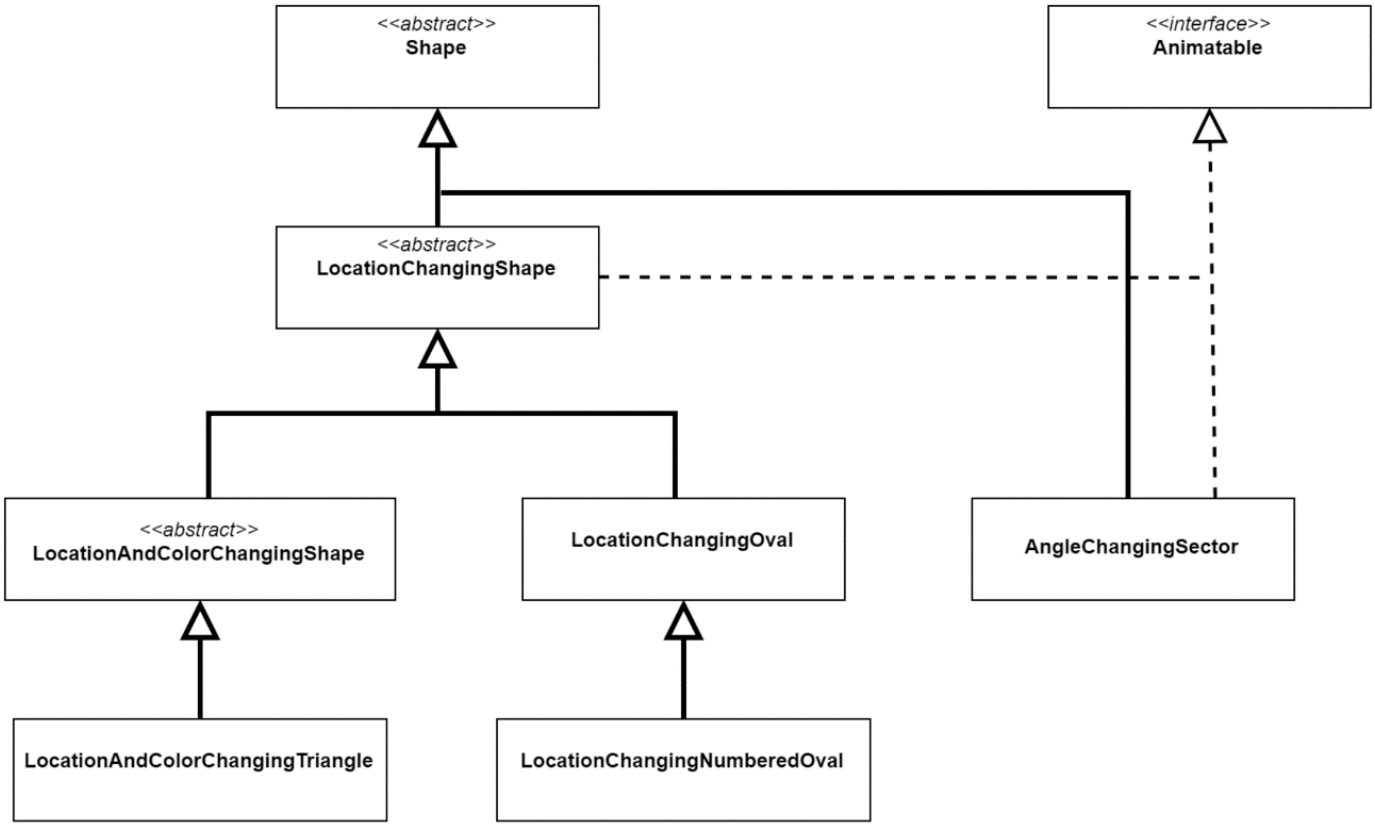
מועד ההגשה:
יום ג', 12/12/17

המטרות של תרגיל בית זה הן להתנסות בתחומים הבאים:

- בניית היררכיית טיפוסים בעזרת הורשה ורב צורתיות.
- שימוש נכון במחלקות מופשטות ובממשקים.
- שימוש במכלים (containers) וב-iterators.
- כתיבת abstraction function ו-representation invariant של טיפוס נתונים מופשט.
- בניית ממשק משתמש גרפי בשפת Java.

הצגת הבעיה

בתרגיל בית זה תכתבו תכנית בעלת ממשק משתמש גרפי שתאפשר להציג ולבצע אנימציה של צורות גיאומטריות שונות. היררכיית הטיפוסים שתמומש מוצגת בתרשים הבא:



המחלקה המופשטת Shape מייצגת צורה גיאומטרית הניתנת לציור בתוך חלון.
תכונותיה של צורה גיאומטרית הן מיקום, גודל, צבע וטיפוס הצורה (האם היא מרובע, משולש, עיגול וכד'). הממשק Animatable מייצג אובייקט שמסוגל לבצע אנימציה של עצמו. המחלקה המופשטת LocationChangingShape מרחיבה את Shape ע"י הוספת תכונה נוספת – מהירות. היא מממשת את הממשק Animatable ומאפשרת לבצע אנימציה של שינוי מיקום בהתאם למהירות. המחלקה LocationChangingShape היא האב הקדמון של מחלקה מופשטת ColorAndLocationChangingShape (אשר מרחיבה את LocationChangingShape ע"י הוספת תכונה נוספת – שינוי צבע) ושל אליפסה ואליפסה ממוספרת.
המחלקה ColorAndLocationChangingShape היא האב הקדמון של מחלקה המייצגת משולש.
המחלקה המוחשית AngleChangingSector מרחיבה ישירות את המחלקה Shape עבור גזרה של אליפסה. היא מממשת את הממשק Animatable ומאפשרת לבצע אנימציה ע"י שינוי זווית הפתיחה.
היחסים בין כל המחלקות המתוארות לעיל מופיעות בתרשים.

הערה 1 : לכל צורה עשוי להיות גודל שמוגדר באופן אחר. למשל, עבור מלבן זה עשוי להיות אורך ורוחב, עבור ריבוע אורך הצלע עבור עיגול רדיוס. לכן, לא ציינו את גודל הצורה כחלק מהתכונות המשותפות לכל הצורות הגיאומטריות.

הערה 2 : הצורות שנגדיר בתרגיל ישמשו בממשק משתמש גרפי שיוצג בהמשך אך הן צריכות להיות גנריות לחלוטין כך שניתן יהיה להשתמש בהן ביישומים שונים. לכן, למשל, צורה לא יכולה להכיל מידע על ממדי החלון של היישום.

בשפת Java לא ניתן לצייר ישירות לחלון אלא יש להשתמש לשם כך באובייקט מטיפוס Graphics המסופק ע"י המערכת החלונאית. מסופק אובייקט מטיפוס Graphics2D היורש מ-Graphics, אך כדי לשמור על תאימות אחורה עדיין מועברים אובייקטים מטיפוס Graphics. לכן, כדי לצייר לחלון יש לבצע המרה מטיפוס Graphics שהתקבל כפרמטר לטיפוס Graphics2D ולהפעיל את המתודות המתאימות של אובייקט זה.

במהלך התרגיל יעשה שימוש במספר מחלקות מתוך החבילה java.awt : Graphics, Graphics2D, Point, Rectangle, Dimension, Color. את המפרט של מחלקות אלה ניתן למצוא ב-Java API Specification באתר של חברת Oracle. כמו כן, ניתן למצוא דוגמאות לשימוש במחלקות אלה ב-Java Tutorial.

שאלה 1 (20 נקודות)

נתון מפרט ומימוש חלקי עבור המחלקה Shape.

א. קראו בעיון את המפרט הנתון עבור המחלקה Shape והשלימו את המימוש של המתודות getLocation() ו-clone(). שימו לב כי המתודה clone() כאן אינה זורקת חריגה מטיפוס CloneNotSupportedException.

ב. הסבירו מדוע במפרט של המתודה clone() במחלקה Shape לא נזרקת אף חריגה.

ג. המתודה setSize() של המחלקה Shape עשוי לזרוק חריגה מטיפוס ImpossibleSizeException. כתבו קובץ בשם ImpossibleSizeException.java שיכיל מפרט ומימוש עבור חריגה זו. שימו לב כי בתוך מימוש החריגה יש לחשב גודל חלופי לגודל הלא חוקי שגרם לזריקת החריגה. הגודל החילופי ישמר בתוך אובייקט החריגה ומי שקרא ל-setSize() יוכל להשתמש בו.

ד. התבוננו על המפרט של המחלקה java.awt.Color. האם מחלקה זו היא mutable או immutable? הסבירו. כיצד עובדה זו משפיעה על המימוש של המחלקה Shape?

ה. כתבו abstraction function ו-representation invariant (כהערה בתוך הקובץ הנתון) עבור המחלקה Shape. כתבו גם מתודת checkRep() וקראו לה במקומות המתאימים בקוד.

להגשה ממוחשבת: המחלקות Shape ו-ImpossibleSizeException.
להגשה "יבשה": תשובות לסעיפים ב', ד'.

שאלה 2 (50 נקודות)

נתון מפרט עבור המחלקה LocationChangingShape, ColorAndLocationChangingShape ונתון הממשק Animatable.

בסעיפים בהמשך, בהם נדרש לתכנן ולממש מחלקות, אתם נדרשים גם:

- לכתוב מפרט עם משפטי @requires, @modifies ו-@effects.
- לכתוב abstraction function ו-representation invariant (כהערות בתוך הקבצים).
- לכתוב מתודת checkRep() ולקרוא לה במקומות המתאימים בקוד.
- בהתאם לצורך, לדרוס את המתודה clone() כדי לאפשר יצירת deep copy של האובייקט.

א. קראו בעיון את המפרט הנתון עבור המחלקה LocationChangingShape ו-ColorAndLocationChangingShape וממשו אותם.
הנחיה: ניתן להשתמש במחלקה java.util.Random.
הנחיה: ניתן להוסיף (אבל לא לשנות) את המפרט הנתון, למשל ע"י הוספת מתודות. שימו לב שעדיין עליכם לשמור על תכן נכון כפי שנלמד בקורס.

ב. צרו מחלקה בשם LocationChangingOval.java שיכיל מפרט ומימוש עבור מחלקה של אליפסה המסוגלת לזוז. מחלקה זו תירש מהמחלקה LocationChangingShape ותממש את כל המתודות המופשטות שלה. בנוסף, היא עשויות, בהתאם לשיקול דעתכם, לדרוס חלק מהמתודות הנורשות מ-LocationChangingShape.
הנחיה: ניתן להשתמש במתודות fillOval() ו-setColor() של המחלקה Graphics2D.

ג. צרו קובץ בשם LocationChangingNumberedOval.java שיכיל מפרט ומימוש עבור מחלקה של אליפסה ממוספרת המסוגלת לזוז. מחלקה אלה תירש מהמחלקה שיצרתם בסעיף ב', כפי שמתואר בתרשים בתחילת התרגיל.
אליפסה ממוספרת היא אליפסה שבמרכזה מופיע מספרה הסידורי מבין האליפסות הממוספרות שנוצרו עד כה כאשר מספרה הסידורי של האליפסה הראשונה הוא 1.
הנחיה: ניתן להשתמש במתודה drawString() של המחלקה Graphics2D.

ד. צרו קובץ בשם LocationAndColorChangingTriangle.java המייצג משולש ישר זווית המשנה את צבעו ואת מיקומו.
נקודת הזווית הישרה של המשולש תיקבע על-פי תכונת ה-location של Shape ואורך הניצבים ייקבע לפי תכונת ה-size (המלבן החוסם) של Shape.
מחלקה זו תירש מהמחלקה ColorAndLocationChangingShape ותממש את כל המתודות המופשטות שלה. בנוסף, היא עשויות, בהתאם לשיקול דעתכם, לדרוס חלק מהמתודות הנורשות מ-ColorAndLocationChangingShape.
הנחיה: ניתן להשתמש במתודות fillPolygon () ו-setColor() של המחלקה Graphics2D.

ה. צרו קובץ בשם AngleChangingSector.java שיכיל מפרט ומימוש עבור מחלקה של גזרה באליפסה המבצעת אנימציה ע"י שינוי זווית הגזרה שלה. זווית ההתחלה של הגזרה וזווית הגזרה במעלות יקבעו ע"י מי שיוצר את הגזרה.
האנימציה תבוצע ע"י קידום זוית הגזרה במעלה אחת עד ל-359 מעלות, ולאחר מכן הורדה של מעלה אחת עד ל-0 מעלות, וחוזר חלילה. מחלקה זו תירש מהמחלקה Shape ותממש את הממשק Animatable.
הנחיה: ניתן להשתמש במתודה fillArc() של המחלקה Graphics2D.

ו. סטודנט מעוניין ליצור מחלקה חדשה בשם LocationChangingCircle עבור עיגול שמסוגל לזוז. עיגול הוא אליפסה שאורכה ורוחבה חייבים להיות שווים.

הסטודנט הציע תכן שבו המחלקה LocationChangingCircle תירש מהמחלקה LocationChangingOval. האם אתם מסכימים עם הצעתו של הסטודנט? הסבירו.

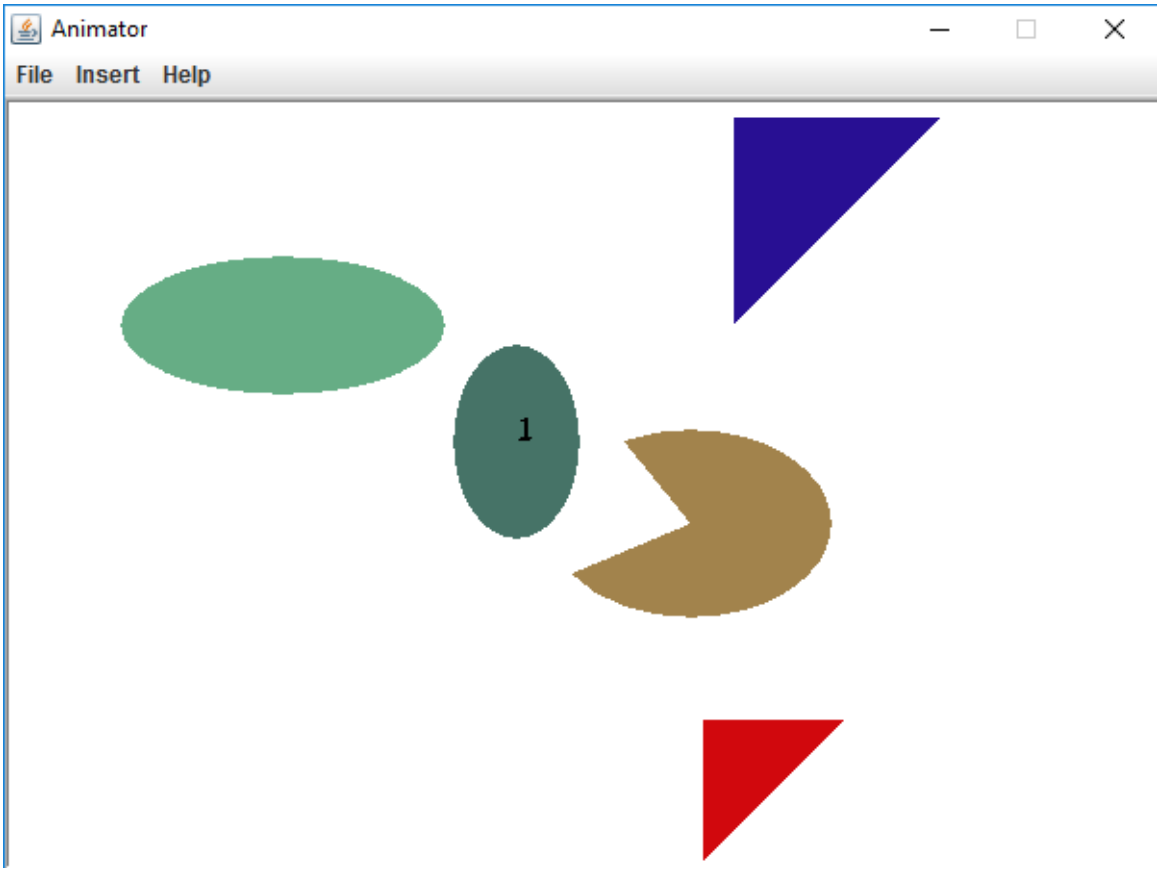
ז. אותו סטודנט טוען שמכיוון שגזרה של אליפסה היא חלק מאליפסה (an oval has sectors), יש לשנות את היררכיית הטיפוסים בתוכנית כך שהמחלקות LocationChangingOval ו-AngleChangingSector יקיימו קשר של הרכבה. כלומר, כל מופע של המחלקה LocationChangingOval יכיל שדות שהם מופעים של המחלקה AngleChangingSector. האם אתם מסכימים עם הצעתו של הסטודנט? הסבירו.

להגשה ממוחשבת: כל המחלקות המפורטות בסעיפים א'-ה'.
להגשה "יבשה": תשובות לסעיפים ו', ז'.

שאלה 3 (30 נקודות)

נתון מפרט ומימוש חלקי עבור המחלקה Animator. מחלקה זו היא המחלקה הראשית של התוכנה, שהפעלה שלה תיצור ממשק גרפי דומה לזה המופיע באיור בעמוד הבא.

תפריט Insert מאפשר הוספת כל אחת מהצורות שמימשתם. בתפריט File קיימות שלוש פעולות: New – מחיקת כל הצורות, Animate – תיבת סימון הקובעת האם מתבצעת כרגע אנימציה, Exit – יציאה מהתוכנית.



א. קראו בעיון את המפרט הנתון עבור המחלקה Animator והוסיפו ואתחלו מיכל בשם shapes שיכיל את הצורות שהוספו ע"י המשתמש.

ב. השלימו את מימוש המתודה paint() כך שתצייר את כל הצורות ב-shapes למסך. מתודה זו נקראת אוטומטית ע"י Swing בכל פעם שיש צורך לצייר מחדש את החלון.
הנחיה 1 : יש להשתמש ב-iterator.
הנחיה 2 : ניתן לקבל אובייקט מטיפוס Graphics שמכיל את אזור הציור בעזרת getContentPane().getGraphics().

ג. השלימו את המתודה actionPerformed() של האובייקט timer כך שתקדם את כל הצורות ב-this צעד אנימציה אחד קדימה.
הנחיה : יש להשתמש ב-iterator.

ד. בסעיפים ב' וג' ביצעתם איטרציה על אותם אובייקטים (הצורות של this), אך עשיתם זאת תוך שימוש בטיפוסים שונים. הסבירו את השוני בין הפעולות שהצריך אתכם להשתמש בטיפוסים שונים. הסבירו גם את המנגנון בשפת Java המאפשר לבצע את הפעולות באופן זה.

ה. השלימו את המתודה actionPerformed() של המחלקה Animator כך שתיצור צורה חדשה שנבחרה מהתפריט. על הצורה להיות במיקום ובגודל אקראיים, כפי שמוגדר בהערה המתאימה בקוד.

ו. המחלקה Animator יורשת מהמחלקה javax.swing.JFrame וממשת את הממשק java.awt.event.ActionListener. הסבירו מדוע היא יורשת מ-JFrame ובמה זה מתבטא בקוד המחלקה. בנוסף, הסבירו מדוע היא מממשת את ActionListener ובמה מימוש זה מתבטא בקוד המחלקה.

ז. סטודנט בחר לממש את shapes בסעיף א' כ-ArrayList. לאחר סיום המימוש של Animator החליט הסטודנט להחליף את מבנה הנתונים של shapes לרשימה מקושרת (LinkedList). הסבירו במילים אילו שינויים על הסטודנט לבצע בקוד כדי לעבור למבנה הנתונים החדש.

להגשה ממוחשבת: המחלקה Animator.
להגשה "יבשה": תשובות לסעיפים ד', ו', ז'.