

Known Issues

ראשית לשם הבנה כללית של התכנית אצרף את [מטלת הפרוייקט מקורס 10118](#) (עשיתי את הקורס בסמסטר א' הקודם אצל ויקטור טאובקין). מדובר בהעמסת קונטיינרים על גבי אונייה באמצעות כפתור move. ניתן להעמיס מס' מכולות אחת על גבי השנייה. כפתור restart יאתחל את האונייה שבנמל מחדש.

הערות כלליות לגבי השינויים שנעשו בפרוייקט המקורי לטובת המטלה:

- בpackage "application" במחלקה "Controller" שיניתי את setOnAction כך שיתפקד על ידי lambda expression במקום בדרך הארוכה. כמו כן במחלקה זו מופיעים האתחולים עבור כל אחד מהכפתורים מהחלונית עם הGUI. היצירה עצמה של הכפתורים נמצאת בpackage "View" במחלקה "MainView" כנהוג במודל MVC.
- בpackage "Model" במחלקה "Port" בשורה 136 - במקור, ההגרלה הרנדומלית עבור הייצוג העשרוני של כל גוון צבע למכולה (מודל RGB) היה בטווח הערכים המקסימאלי (0 עד 256), אך לטובת סעיף 13 בו נדרשת כפילות בין שתי תכונות במספר מופעים של האובייקט, ביצעתי שינוי בערכי redn והgreen של האובייקט כך שיוגרלו בין 0 ל 2, מה שגרם למכולות להיצבע כעת בגווי כחול במקום בצבעוני. (אם היה נשאר כמקור כך שעבור כל ערך יוגרל טווח הצבעים המקסימאלי - היה סיכוי קלוש ששני גוונים ייצאו זהים לשני אובייקטים בשתי תכונות כמבוקש, ובשאר התכונות שאינן צבע לא ייתכן שיוויון בין שני אובייקטים).
- בpackage "Model" במחלקה "Container" בשורות 147 ו156 הוספתי מתודות hashCode ו-equals עבור סעיף 13. השיטות הללו לא היו קיימות ולא היה בהן צורך בפרוייקט המקורי.
- כל שאר המתודות שנוספו עבור המטלה נמצאות בpackage "Model" במחלקה "Model" החל משורה 140.

כעת אתייחס לכל אחד מהסעיפים במטלה הנוכחית:

12. עבור ביצוע הסעיף הוספתי בחלונית של JavaFX כפתור בשם "add to collection 1". בלחיצה על הכפתור, מופעלת המתודה copyContainersToColl1() (נמצאת בpackage "Model" במחלקה "Model" בשורה 145) והקונטיינרים שעל מחסנית הנמל מועתקים לתוך TreeSet. בנוסף מודפסים לconsole כל פרטי הקונטיינרים שנמצאים כעת על האונייה בסדר יורד לפי תכונת left שלהם (מעין ערך הx של הפינה השמאלית העליונה של המכולה). נשים לב שאם הועברו כבר חלק מהמכולות אל המשאית ולוחצים שוב על הכפתור "add to"

collection 1" אז מודפסים לconsole רק הקונטיינרים שכעת על האונייה, פחות אלו שכבר על המשאית או לצד הדרך. המיון מבוצע בעזרת comparator.

13. עבור ביצוע הסעיף הוספתי בחלונית של JavaFX כפתור בשם "add to collection 2". בלחיצה על הכפתור, מופעלת המתודה copyContainersToColl2() (נמצאת בpackage "Model" במחלקה "Model" בשורה 152) והקונטיינרים שעל מחסנית הנמל מועתקים לתוך TreeSet וממנו לתוך HashSet. בלחיצה זו נוספים לHashSet ומודפסים לconsole רק פרטי הקונטיינרים כך שאם קיימת מכולה/מכולות שערכי תכונות redn והgreen שלם/שונים לאותם ערכי redn וgreen של מכולה/מכולות אחרים, ייכנס ויודפס רק אחד מהם, כלומר HashSet לא יכניס איברים חדשים שהם כפילות עם מה שכבר נמצא בו. הברירה נעשית בעזרת hashCode ו- equals.

14. כל אחת מהשיטות לעיל מפעילה את המתודה print(Set<?> set) אשר מדפיסה את תוכן collectionnn שהתקבל כפרמטר בעזרת iterator.

15. כפי שציינתי מעלה, בpackage "application" במחלקה "Controller" שיניתי את הsetOnAction כך שיתפקד על ידי lambda expression במקום בדרך הארוכה.

16. מתועד לעיל.

17. הוספתי בpackage "Model" מחלקה בשם "MyArrayList". **הערה – ייתכן שבהקשר הMVC זה אינו המקום המתאים ליצור את המחלקה הזו, מכיוון שהקורס הנוכחי אינו עוסק בMVC אז לא שמתי דגש על כך.

18. עבור ביצוע הסעיף הוספתי בחלונית של JavaFX כפתור בשם "My Array List". בלחיצה על הכפתור, מופעלת המתודה copyContainersToMyArrayList() (נמצאת בpackage "Model" במחלקה "Model" בשורה 211) והקונטיינרים שנמצאים בHashSet מועתקים לתוך MyArrayList. בלחיצה זו נוספים לMyArrayList ומודפסים לconsole רק פרטי הקונטיינרים כך שאם קיימת מכולה/מכולות שערכי תכונות redn והgreen שלם/שונים לאותם ערכי redn וgreen של מכולה/מכולות אחרים, ייכנס ויודפס רק אחד מהם, כלומר MyArrayList מעתיק מהHashSet אחד לאחד לפני הסדר את המכולות ללא הכפילויות.

19. הוספתי למחלקה "MyArrayList" את המתודה iterator() אשר מחזירה מימוש איטרטור פרטי (המימוש במחלקה פרטית "MyIt" אשר גם כך נמצאת ב"MyArrayList"). כמו כן הדפסת התוכן נעשית דרך הכפתור "My Array List". הפלט אכן זהה לפט שהודפס בעזרת HashSet.

20. עבור ביצוע הסעיף הוספתי בחלונית של JavaFX כפתור בשם "remove 1". בלחיצה על הכפתור, מופעלת המתודה removeContainers1() (נמצאת בpackage "Model" במחלקה "Model" בשורה 222) והיא מוחקת מהרשימה את כל המכולות אשר ערך התכונה red שלהם (רמת הצבע האדום במודל RGB) שווה ל-0. לאחר מכן גם מבוצעת הדפסה של התוכן החדש של MyArrayList.

21. עבור ביצוע הסעיף הוספתי בחלונית של JavaFX כפתור בשם "Java Array List". בלחיצה על הכפתור, מופעלת המתודה `copyContainersToJavaArrayList()` (נמצאת בpackage "Model" במחלקה "Model" בשורה 217) והקונטיינרים שנמצאים בHashSet מועתקים לתוך ArrayList של ג'אבה. התוצאה המתקבלת זהה לזו שבסעיף 18. כמו כן אותו כנ"ל לגבי `remove`: הוספתי בחלונית של JavaFX כפתור בשם "remove 2". בלחיצה על הכפתור, מופעלת המתודה `removeContainers2()` (נמצאת בpackage "Model" במחלקה "Model" בשורה 228) והיא מוחקת מהרשימה את כל המכולות אשר ערך התכונה `red` שלהם (רמת הצבע האדום במודל RGB) שווה ל-0. לאחר מכן גם מבוצעת הדפסה של התוכן החדש של הArrayList. ***הערה מסעיף 17 ועד כאן – ניתן לשחק ולהעמיס מכולות מהאונייה למשאית ואז ללחוץ שוב על הכפתורים, ולראות שהתוכן של הסטים שמודפס משתנה בהתאמה.

22. מתועד לעיל.

23. **הערה לגבי הסעיף הנ"ל – לא הייתה התייחסות בשיעורים להיכן צריך להוסיף את המחלקות החדשות שנדרשות בהתאם למודל MVC, לכן הוספתי לפי שיקול דעתי. יצרתי בpackage "Model" את הממשק "Observer" ואת המחלקות "MyButton" ו-"MyLabel", המשמשות כListeners עבור הSubject שהוא המחלקה "MyArrayList". כאשר נלחץ על הכפתור "My Array List" המאזינים יגיבו לכך ויפיעו על החלונית הכפתור והתווית המבוקשים (נעשה בעזרת השיטה `click()` הנמצאת בpackage "Model" במחלקה "MyArrayList" בשורה 62). כמו כן הלחיצה על הכפתור שהופיע, יודפס למסך תוכן האיטרטור כפי שנוצרו בו קודם לכן בלחיצה על "My Array List". *הערה נוספת – אם נוציא מכולות מהאונייה למשאית, ולאחר מכן נלחץ שוב על הכפתור "iterator" אז התוכן לא ישתנה בהתאם למצב הנוכחי, אלא ישאר התוכן הקודם (כי לא משוגר מידע חזרה לSubject). כמו כן כפתור `restart` מאפס את התהליך (נעשה בעזרת השיטה `clickRestart()` הנמצאת בpackage "Model" במחלקה "MyArrayList" בשורה 71). אציין כי הוספתי כפתורים הטיפוסי המחלקות החדשים בpackage "View" במחלקה "MainView", וכן בController הוספתי רשימה מסוג `MyArrayList` בשורה 51. התפקוד של הצופה-מאזין נמצא בכפתורים הרלוונטיים ב`setOnAction`.

24. לא משוגר בחזרה לSubject מידע כלשהו.

25. מתועד לעיל.

26. עבור ביצוע הסעיף יצרתי ממשק `Command` ו-3 מחלקות שמממשות אותו: `ConcreteCommand1`, `ConcreteCommand2`, `ConcreteCommand3`. כל מחלקה מממשת את המתודה `execute()` בדרכה:

- `ConcreteCommand1` בעלת תכונה `scene` מטיפוס `Scene`, וכאשר מופעל ה-`execute()` צבע `scene` משתנה באופן רנדומלי. השיטה מופעלת בלחיצה על הכפתור "change background" בחלונית JavaFX. כמו כן בController יצרתי אובייקט מסוג `ConcreteCommand1` בשורה 55, ובשורה 118 ניתן לראות את ה-

setOnAction שקורה בעת הלחיצה על הכפתור. הגדרת הכפתור נמצאת במערכת mainView בשורה 93 (command).

- ConcreteCommand2 בעלת תכונה model מטיפוס Model, וכאשר מופעל ה- execute() מתקיים כל התפעול של כפתור ה-restart שנוצר עבור הפרויקט מהקודם הקודם ותפקידו לאפס את כל תהליך העמסת המכולות מהאונייה למשאית. השיטה מופעלת בלחיצה על הכפתור "restart" בחלונית ה-JavaFX. כמו כן ב-Controller יצרתי אובייקט מסוג ConcreteCommand2 בשורה 56, ובשורה 68 ניתן לראות את ה-setOnAction שקורה בעת הלחיצה על הכפתור.

- ConcreteCommand3 בעלת תכונה view מטיפוס MainView, וכאשר מופעל ה- execute() מופעלות מס' פעולות של ניקוי ה-GUI והצגתו מחדש בצורה מעודכנת. הצורך ב-Command זה עלה מתוך הרצון לבטל את שכפול הקוד המרובה שנוצר לאורך תרגיל הבית המתגלגל הזה שבו יצרתי עוד ועוד כפתורים אשר לאחר הפעלת כל אחד מהם יש צורך "לנקות" את התצוגה ולהציגה עם הגרפיקה המעודכנת. השיטה מופעלת בלחיצה על כל אחד מהכפתורים בחלונית ה-JavaFX פרט לכפתור שמשנה את צבע הרקע שנוצר בסעיף זה קודם לכן (כי אין צורך בכך). כמו כן ב-Controller יצרתי אובייקט מסוג ConcreteCommand3 בשורה 57, והפעלת ה-execute שלו מתקיימת בכל setOnAction מטה.

27. עבור ביצוע הסעיף יצרתי במחלקה "Container" החל משורה 172 מחלקה פנימית בשם "Memento" ומספר מתודות נוספות אשר תפקידן לאפשר מימוש של תבנית העיצוב הנ"ל. האובייקטים מטיפוס Container נמצאים בהתחלה על האונייה, שהיא למעשה מחסנית, והמטרה היא לשמור את המצב ההתחלתי מבחינת המיקום המקורי (קרי המצב הפנימי) שלהן על גבי האונייה (כלומר לפני שמתחילים להפעיל את הכפתור move ושאר המניפולציות). במחלקה "Model" הוספתי תכונה חדשה בשורה 17 - מחסנית מטיפוס גנרי של Memento. החל משורה 272 יצרתי שני פונקציות - אחת שמעתיקה את המצב ההתחלתי של המכולות למחסנית Memento ואחת שמדפיסה את אותו התוכן ל-Console. כמו כן ניתן להדפיס בכל רגע נתון את המצב ההתחלתי של המחסנית על ידי לחיצה על הכפתור "get start position of containers" בחלונית ה-JavaFX. מציעה ללחוץ עליו בתחילת ההרצה לראשונה, ולאחר מכן לשחק ולהעביר מספר מכולות את המשאית, ואז ללחוץ עליו שוב כדי להיווכח שאכן המצב ההתחלתי נשמר. המצב ההתחלתי מועתק בשורה 59 ב-Controller.

28. מתועד לעיל.

נספח: הפרוייקט מקורס 10118:



אפקה המכללה האקדמית להנדסה בתל-אביב
בשיתוף פעולה אקדמי עם אוניברסיטת תל-אביב

פרויקט סיום בקורס "תכנות מונחה עצמים".

הנחיות טכניות להגשה

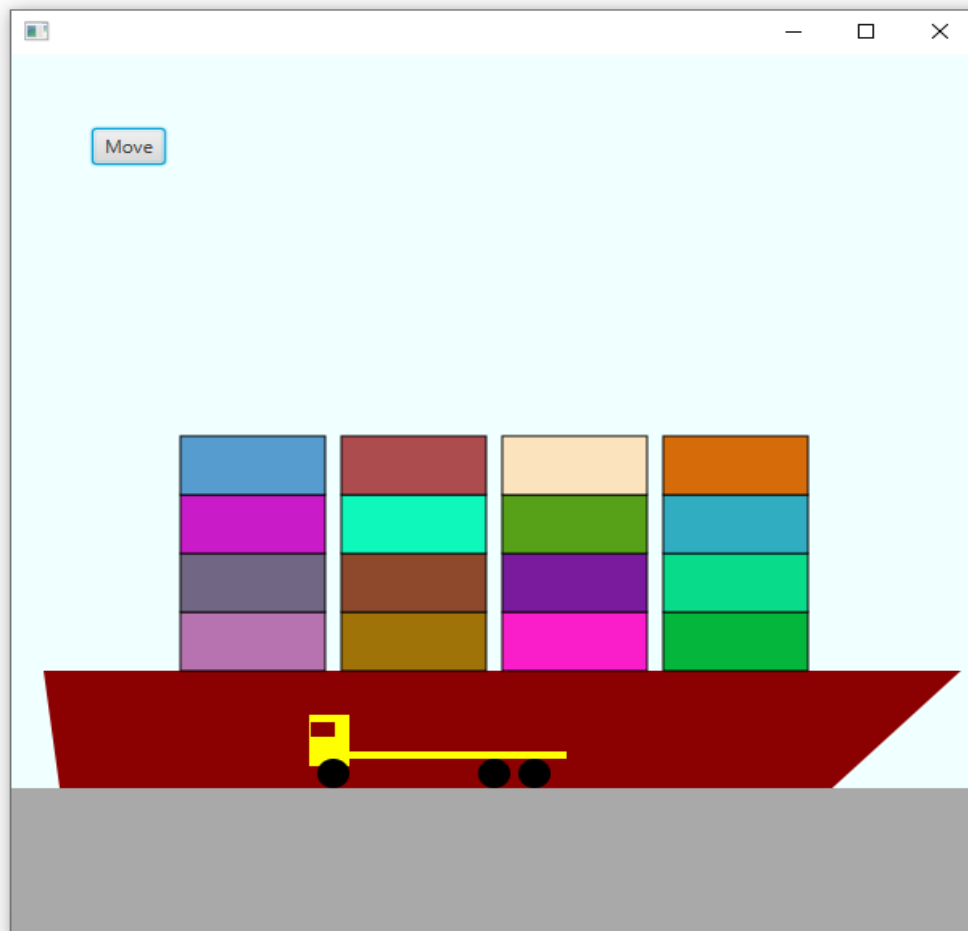
1. הגשת הפרוייקט בתיבת ההגשה במודל עד התאריך המצוין בתיבה.
2. ניתן להגיש את הפרוייקט בזוגות או ביחיד.
3. רק אחד מבני הזוג מעלה את הפרוייקט למודל

מה עליכם לעשות?

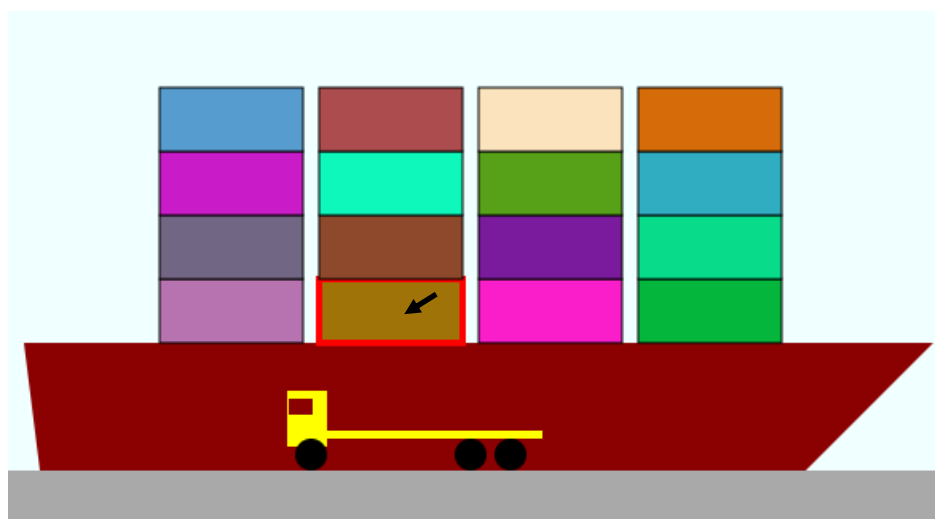
כתבו תכנית המשתמשת ב-javafx ובתבנית MVC לפתרון הבעיה המתאורת בהמשך. יש להשתמש בעקרונות של תכנות מונחה עצמים כפי שנלמד בכיתה.

תיאור הבעיה

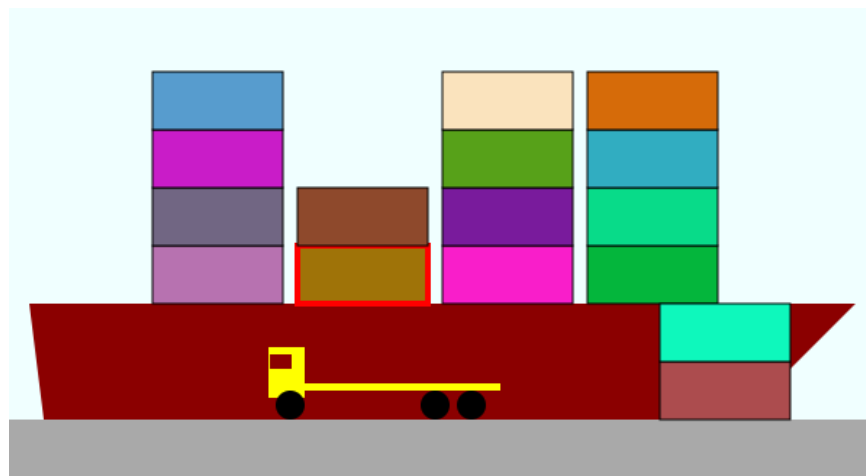
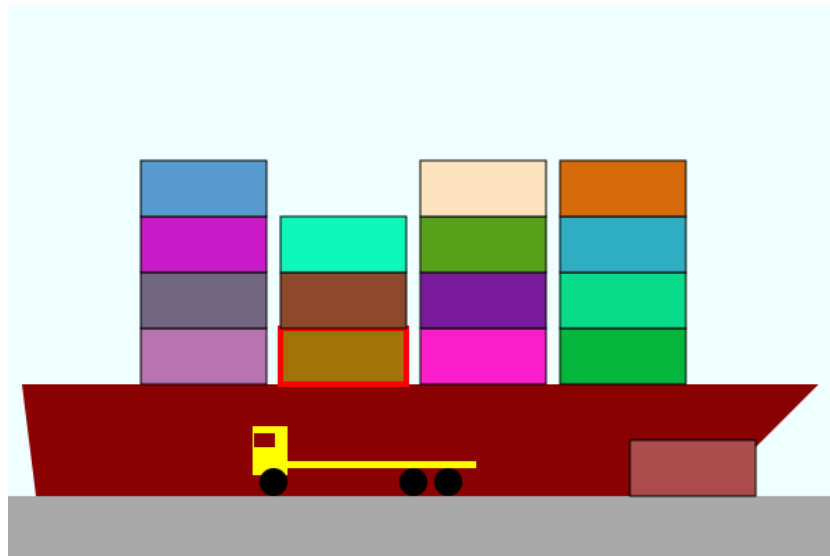
בנמל עוגנת ספינת מכולות ועל המזח נמצאת משאית אשר יכולה להעביר מכולה אחת בלבד למגרש לאחסון מכולות.

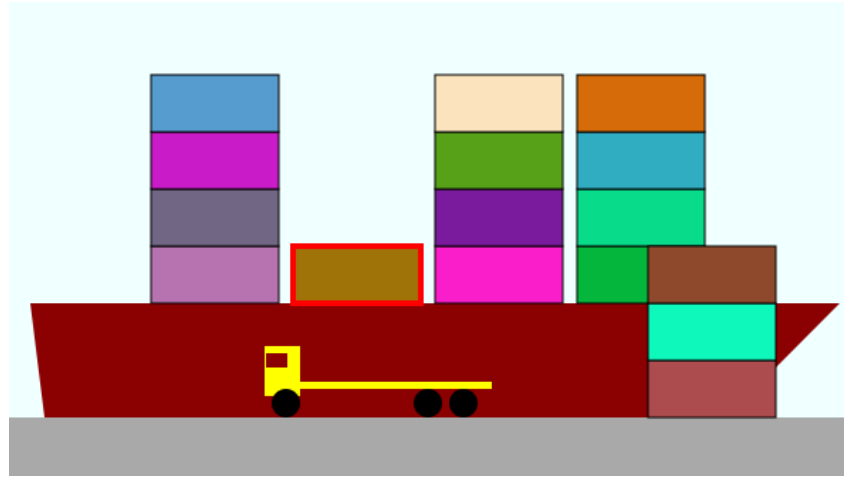


משתמש המערכת צריך לסמן מכולה (לאו דווקא עליונה) בעזרת קליק העכבר ולאחר מכן להתחיל ללחוץ על לחצן "Move" כדי לראות איך המערכת מוציאה את המכולה מהספינה למשאית. המנוף (שאינן צורך לציירו) יכול להעביר בפעולה אחת מכולה אחת מסיפונה של הספינה אל המזח או על המשאית. ניתן לשים מכולה על מכולה אחרת. להלן סימון המכולה:

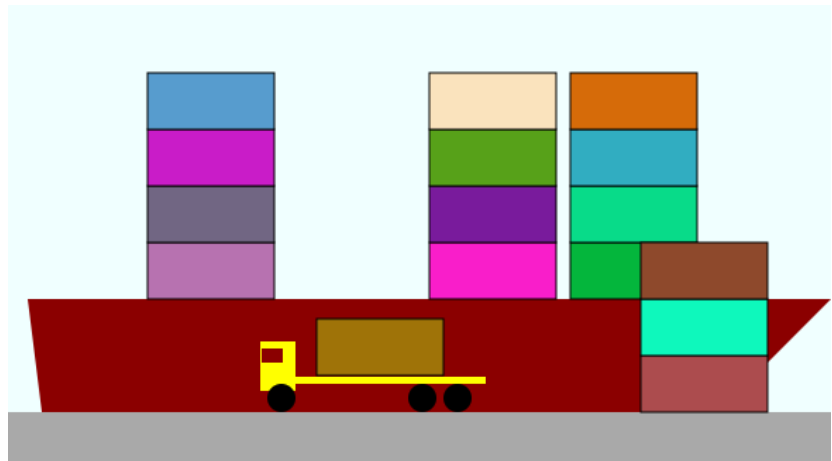


תמונות הבאות מציגות תוצאות של לחיצות עוקבות על לחצן "Move". שימו לב, כי המכולות נאספות על המזח בסדר הפוך מזה שהיה להן על הסיפון.

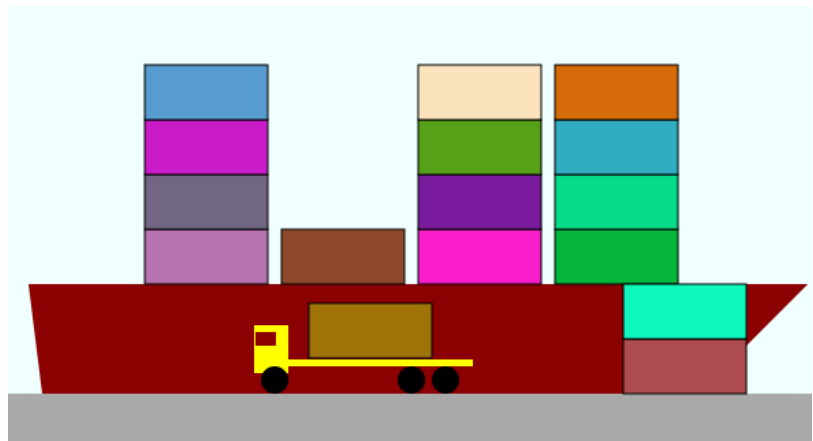


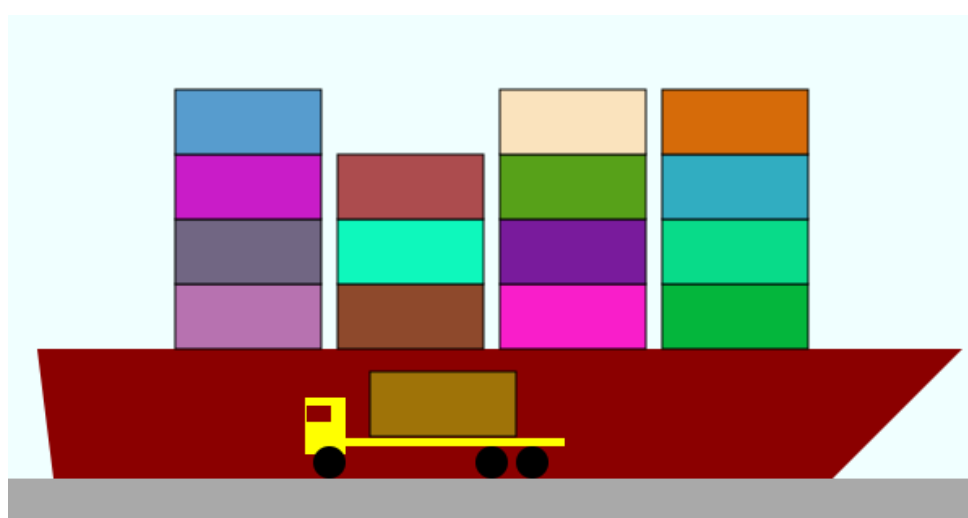
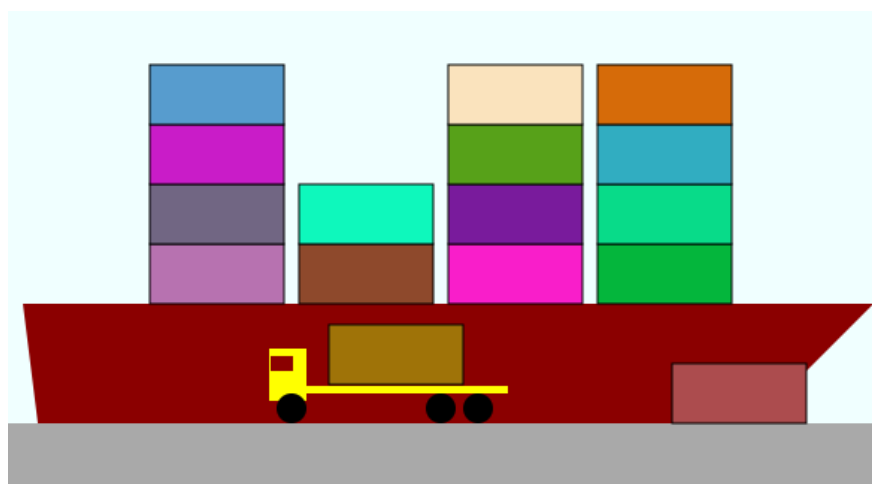


כאשר מגיע תורה של המכולה המסומנת על ידי המשתמש היא עוברת ישירות אל המשאית:



לחיצות נוספות על לחצן "Move" מחזירות את המכולות מהמזח חזרה אל הסיפון





בהצלחה!

נספח 2: המטלה הנוכחית בקורס 10119:

חלק 1: Collections, Generics, Iterator, Lambda Expression

12. יש להוסיף אפשרות בתפריט, שהפעלתה תעביר את אחד מהמערכים של אובייקטים / ArrayList הקיימים, לבחירתך, ל- Collection הכי מתאים, כך שבכל שלב אם נרצה להדפיס את תוכן ה- Collection הזה, נקבל תמיד סדר לפי מיון יורד, כאשר המיון נקבע לפי אחת התכונות באובייקט לבחירתך.
- יש לבחור תכונה שהיא לא מספר מזהה או משהו ייחודי לכל אובייקט, אלא תכונה עם ערך שחוזר על עצמו פה ושם באובייקטים השונים.**
- יש לבצע כך, שכל האיברים מהמערך המקורי יוכנסו תמיד, אפילו אם כל התכונות שלהם זהות לחלוטין. יש להדפיס את תוכן ה- Collection בעזרת Iterator כפי שלמדנו בכיתה.
13. כעת, יש להוסיף אפשרות בתפריט, שהפעלתה תעביר את כל תוכן ה- Collection הקודם ל- Collection נוסף מתאים, כאשר הדגש יהיה כעת על המהירות (הוספה / חיפוש / הסרה), ואין משמעות למיון כלשהו. בנוסף, ה- Collection הפעם לא יכניס איברים חדשים שהם כפילות עם מה שכבר נמצא בו. כפילות מוגדרת לפי שוויון מוחלט בין 2 תכונות לבחירתך.
- יש לבחור תכונות שהן לא מספר מזהה או צירוף ייחודי לכל אובייקט, אלא תכונות עם צמד ערכים שחוזרים על עצמם פה ושם באובייקטים השונים.**
14. יש להדפיס את תוכן ה- Collection בעזרת Iterator, כפי שלמדנו בכיתה, ורוצים שההדפסה תהיה לפי סדר ההכנסה של האיברים ל- Collection החדש, כלומר לפי הסדר המקורי שהיה בעצם עם המיון, אבל בלי הכפילויות.
15. בקוד שאתה מוסיף, באם יש צורך, וניתן, אזי יש לעבוד עם Lambda Expression.
16. בקובץ ה- Known Issues, יש להוסיף הסבר, באיזה מערך / ArrayList מקורי השתמשתם מתוך הפרויקט, איפה הוא נמצא בקוד (שם הקובץ, מספרי שורות וכו'), ולפי איזה תכונה מתבצע המיון, ולפי איזה תכונות נקבעת הכפילות ב- Collection השני, וכיצד להפעיל את שני הסעיפים הנ"ל דרך התפריט או דרך איזה כפתורים.

חלק 2: ArrayList, Design patterns: Iterator, Observer

17. יש להוסיף אפשרות נוספת בתפריט, שבה ניצור מחלקה משלנו שתיקרא `MyArrayList`, שהיא תבצע בעצם את מה ש-`ArrayList` מבצעת באופן הכי בסיסי, כלומר תהיה לנו רק מתודה `add`.
(אין צורך לרשת מ-`Collection/List` כמו ב-`ArrayList` המקורי)
18. יש להעביר את כל האיברים מ-`Collection` של הסעיף הקודם לזה שבלי הכפילויות), אל `MyArrayList`, לפי הסדר שבו הם נמצאים ב-`Collection`.
19. בעזרת ה-`design pattern` של ה-`Iterator`, יש להוסיף למחלקה מימוש של `Iterator`, ואז להפעיל אותו בדיוק כמו שמפעילים `Iterator` רגיל של `Java`, כלומר להדפיס את תוכן ה-`MyArrayList`.
(הפלט צריך להיות זהה לפלט של ה-`Collection` מהסעיף הקודם).
20. יש להוסיף גם מימוש למתודה `remove` של ה-`Iterator`, ולהציג תפועול שלה.
21. כעת, יש להציג גם במקביל, אותו דבר בדיוק עם `ArrayList` של `Java` ועם `Itrator` מובנה של `Java`, ולוודא שקיבלנו אותן תוצאות.
22. עבור כל הנ"ל, בקובץ ה-`Known Issues`, יש להוסיף הסבר, כיצד להפעיל את הסעיף הנ"ל דרך התפריט או דרך איזה כפתורים, ומה בדיוק מבצע ה-`remove` ו/או איך להפעיל אותו, הן במה שכתבת והן בהרצת 'הביקורת' של ה-`Iterator` המקורי של `java`.
23. בשלב הזה, יש לעבוד עם ה-`design pattern` של `Observer` שלמדנו בכיתה.
אין להשתמש במחלקות מוכנות של `java` כגון `Observable` וכך הלאה.
יש להניח ש-`MyArrayList` הוא ה-`Subject`. כאשר אנחנו יוצרים `Iterator` חדש כפי שבנינו בני"ל, ה-`Subject` יודיע ל-`Listeners (Observers)` שלו על כך שנוצר `iterator` חדש והוא מוכן לפעולה. יש לצור שתי מחלקות משלך, אחת של `Button` ואחת של `Label`, שהם יהיו ה-`Listeners (Observers)`. יש להשתמש בירושה בהתאם.
- כעת (תוך שימוש ב-`setVisible`, `setEnabled` וכו'): כאשר ה-`Listeners` מקבלים הודעה שנוצר ה-`Iterator`, ה-`Label` מופיעה, ומציגה צבע ירוק עם כיתוב מתאים, וגם ה-`Button` מופיע ומאפשר עצמו לפעולה, כך שעכשיו אם נלחץ עליו נקבל רק את הפעלת ה-`Iterator` הזה שכבר נוצר, של `MyArrayList`. (יש לוודא שמקבלים אותו פלט כמו קודם)
24. הערה: בשלב זה, אין צורך לשגר בחזרה ל-`Subject` מידע כלשהו.
25. עבור כל הנ"ל, בקובץ ה-`Known Issues`, יש להוסיף הסבר, כיצד להפעיל את הסעיף הנ"ל דרך התפריט או דרך איזה כפתורים, איך נקראות המחלקות של ה-`Listeners`, ואיך להשוות עם הפלט הקודם.

חלק 3: Design patterns: Command, Memento

26. עליך להעביר שלוש פעולות, לבחירתך, שמופעלות מהתפריט / כפתורים שבנית, לעבודה לפי **תבנית ה-Command**, כפי שנלמדה בכיתה, וזאת בכדי לצור צימוד רופף.
(אפשר להמיר כפתורים / פקודות תפריט שבנית כאן בסעיפים הקודמים, או מהפרויקט בכלל.)
27. **בעזרת תבנית ה-Memento**, עבור מחלקה מרכזית בפרויקט (לבחירתך), שאובייקטים מהטיפוס שלה נמצאים בתוך רשימה או מערך, יש לשמור את המצב הפנימי של כל אחד מהאובייקטים שברשימה. (כלומר, לשמור את המצב הפנימי של התכונות שהן private של כל אחד מהאובייקטים שנמצאים ברשימה). לאחר מכן, הוסף אפשרות לשחזר את כל הרשימה עם כל האובייקטים כפי שנשמרו. הדפס לפני ואחרי כדי לוודא פלט זהה.
(אין צורך לשמור בקובץ, אלא רק להדגים את פעולת התבנית, כפי שעשינו בכיתה, למשל להעביר את האובייקטים ל-Collection אחר מטיפוס גנרי של Memento.)
28. עבור כל הנ"ל, בקובץ ה-Known Issues, יש להוסיף הסבר, כיצד ניתן להפעיל את שני הסעיפים הנ"ל דרך התפריט או דרך איזה כפתורים, וזאת כדי להדגים את פעולתה המלאה של כל תבנית, והיכן נמצאות המחלקות הקשורות בפעולת כל תבנית.