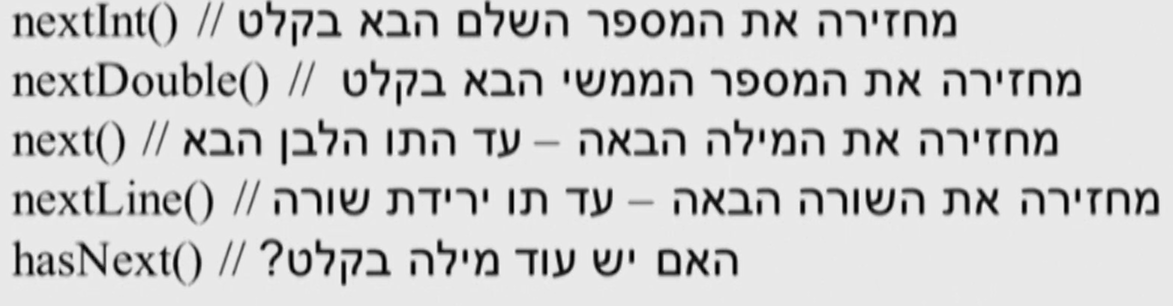
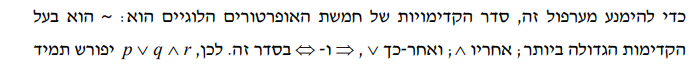
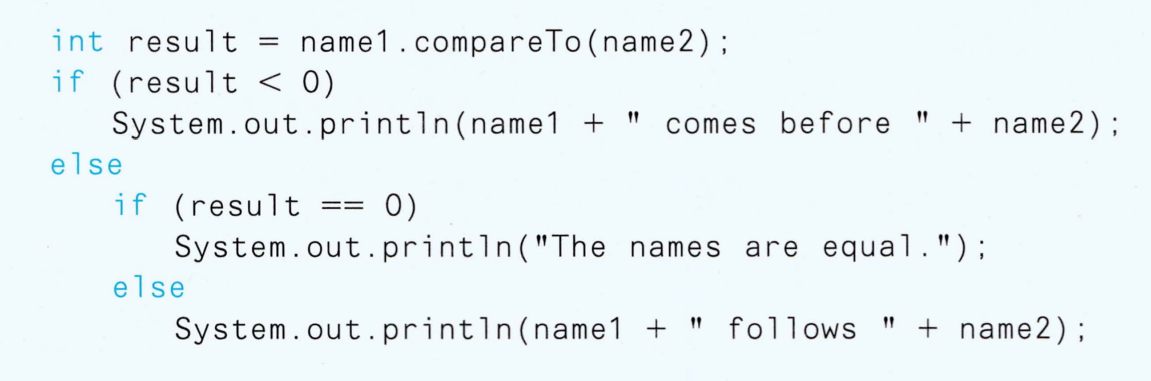
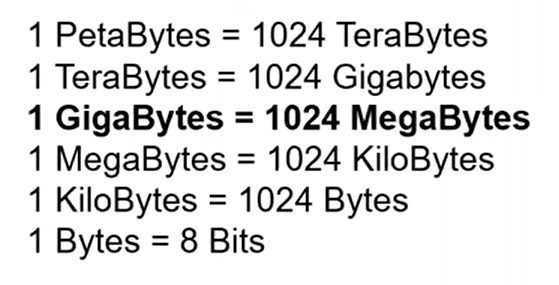
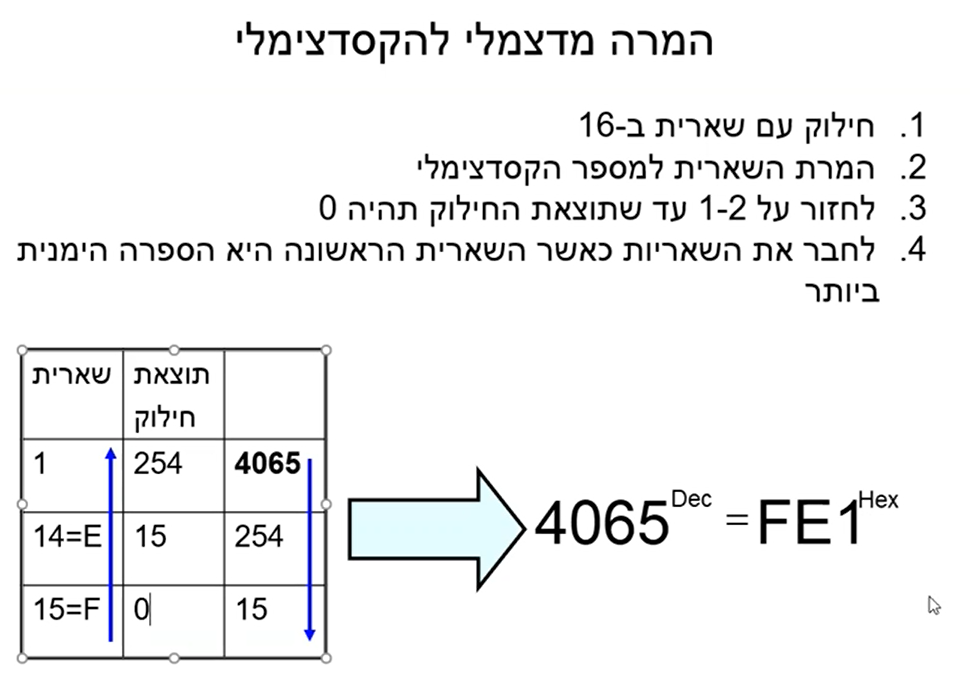
­­­­­Scanner:



! > && > ||

(OU V HU) => ((DC ח DE) U (SC ח SE )) ח ̚ IC ח MR



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

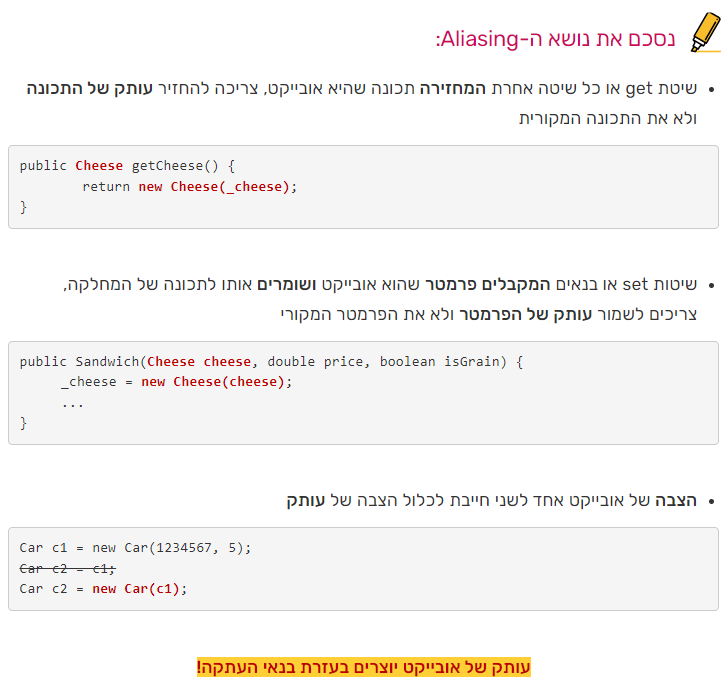
**משווים אובייקטים עם .equals("")**

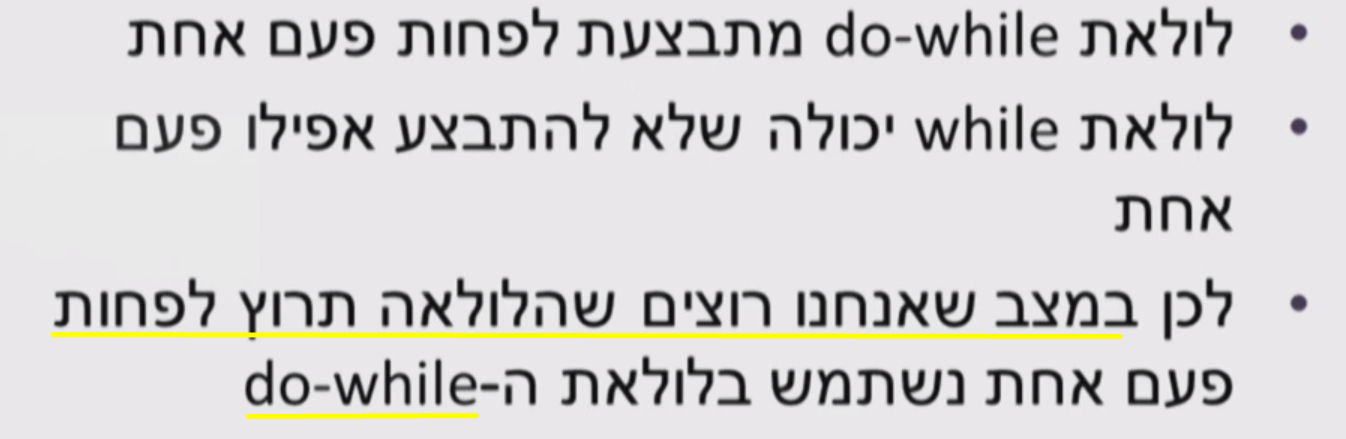
X = (…)? 3 : 7;

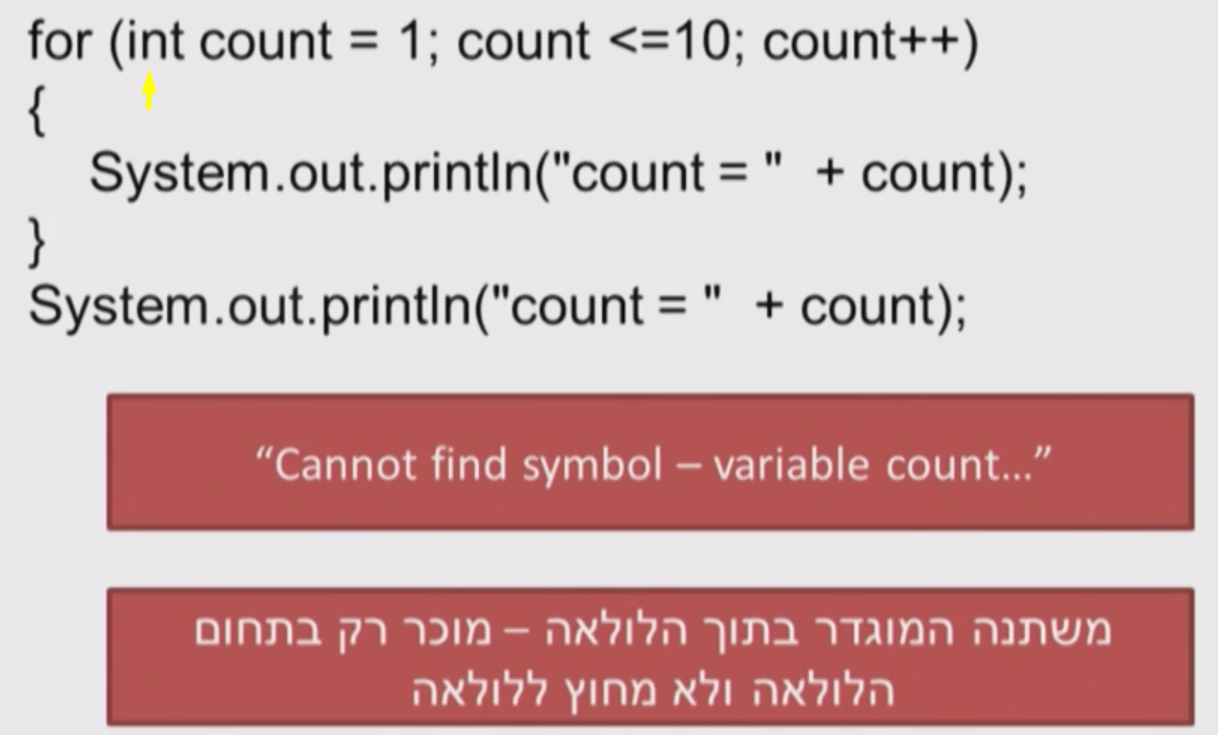
          System.out.println("Flight Elal1 is: " + (elal1.full()?"full":"not full"));

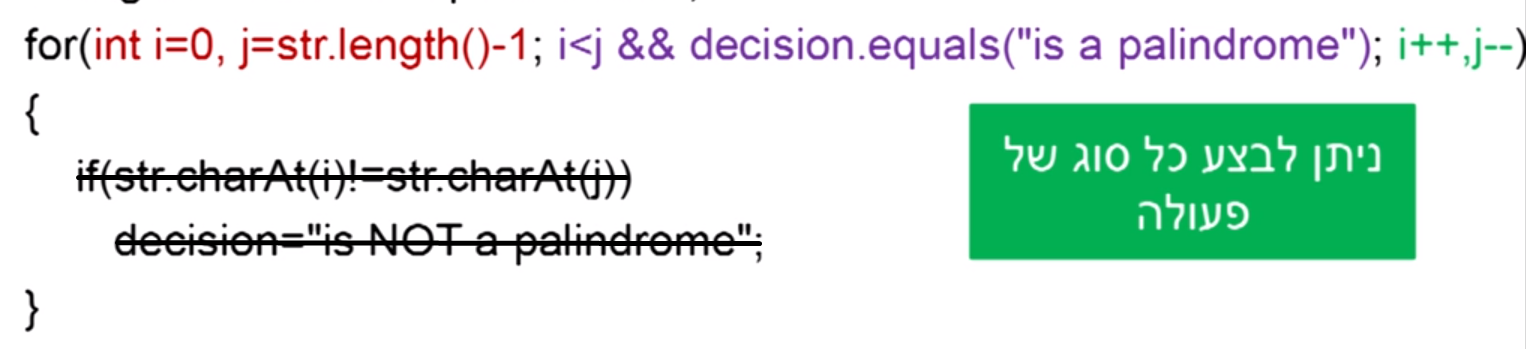
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטימניעת ailising עם בנאי העתקה









**API**

1. מתחילות בלוכסן ושתי כוכביות “\*\*/” ומסתיימות בכוכבית לוכסן “/\*”.
2. במחלקה, כל בנאי או שיטה המוגדרת public יש לכתוב עבורה הערת API .
3. דוגמא:

/\*\*

\* This class represent a three-dimension box.

\* The sides are integers value start by 1.

\* The default size is 10\*10\*10.

\*

\* @version v.0.1

\* @author Shaul

\*

\*/

לפני הבנאי נוסיף את ההערה הזו:

/\*\*

\* Constructor for create new box, getting the size of each side.

\*

\* @param length the length of the box, must be at least 1.

\* if value will be less than 1, it will be changed to 1.

\* @param width the width of the box, must be at least 1.

\* if value will be less than 1, it will be changed to 1.

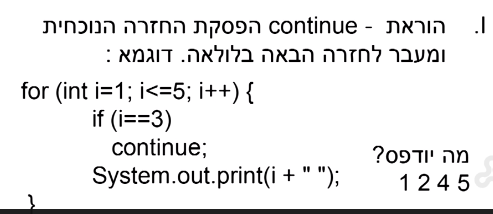
\* @param height the height of the box, must be at least 1.

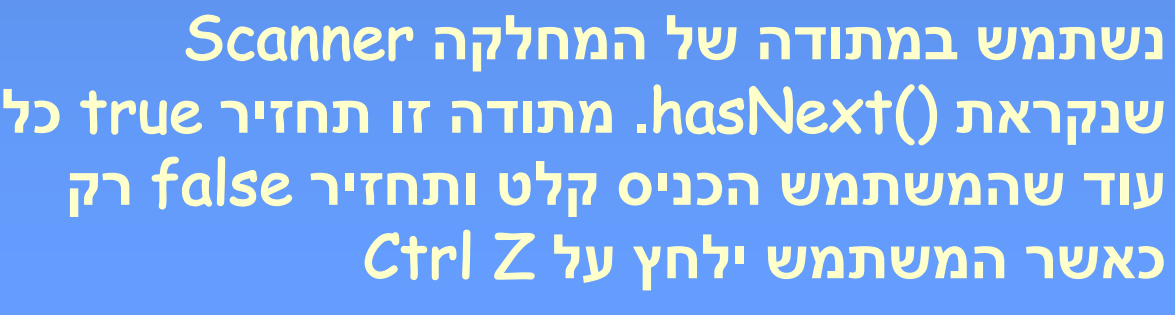
\* if value will be less than 1, it will be changed to 1.

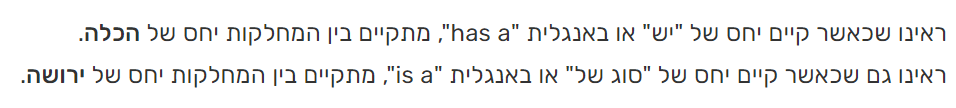
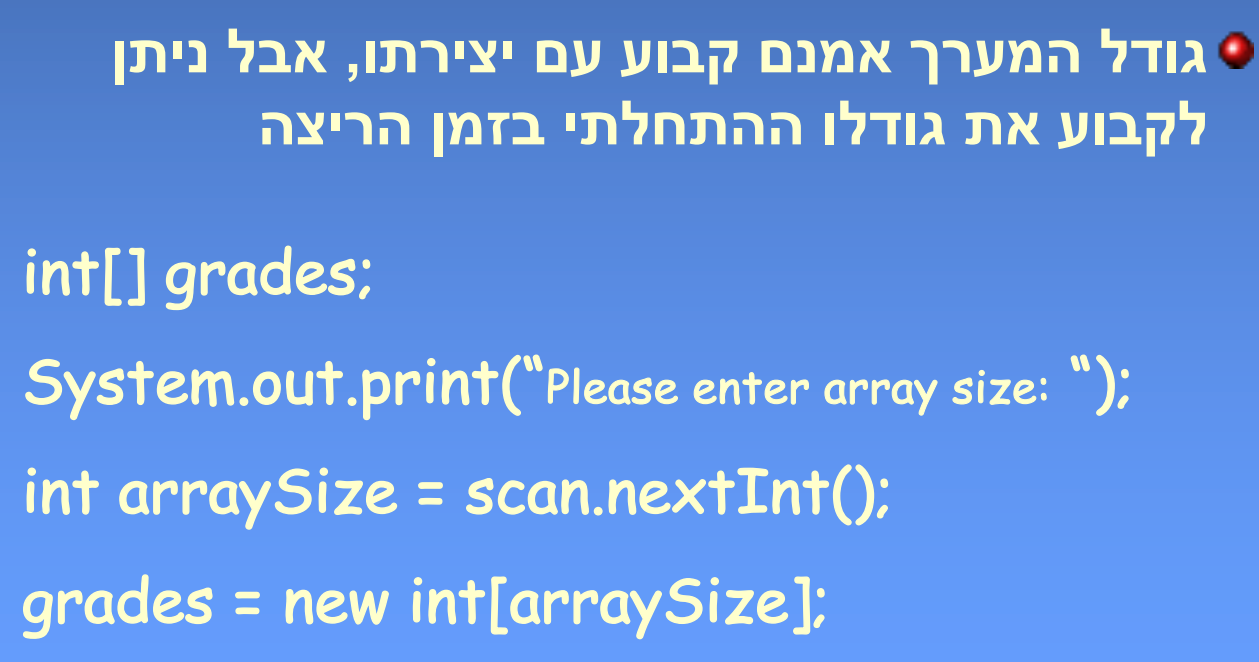
\*/

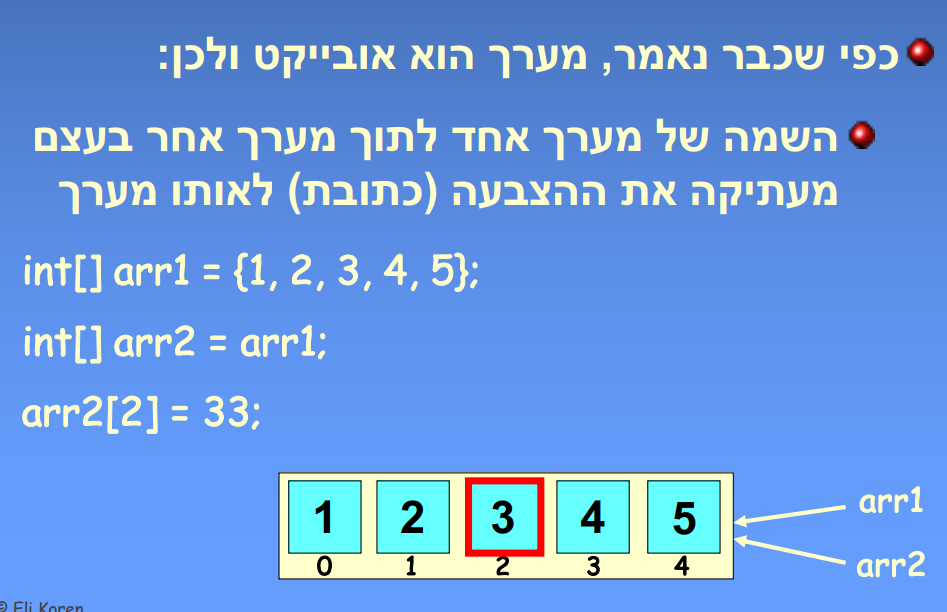
תמונה שמכילה טקסט, מכתב

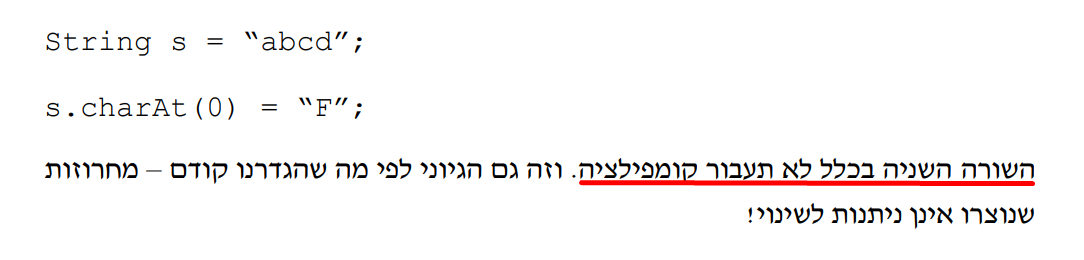
התיאור נוצר באופן אוטומטי





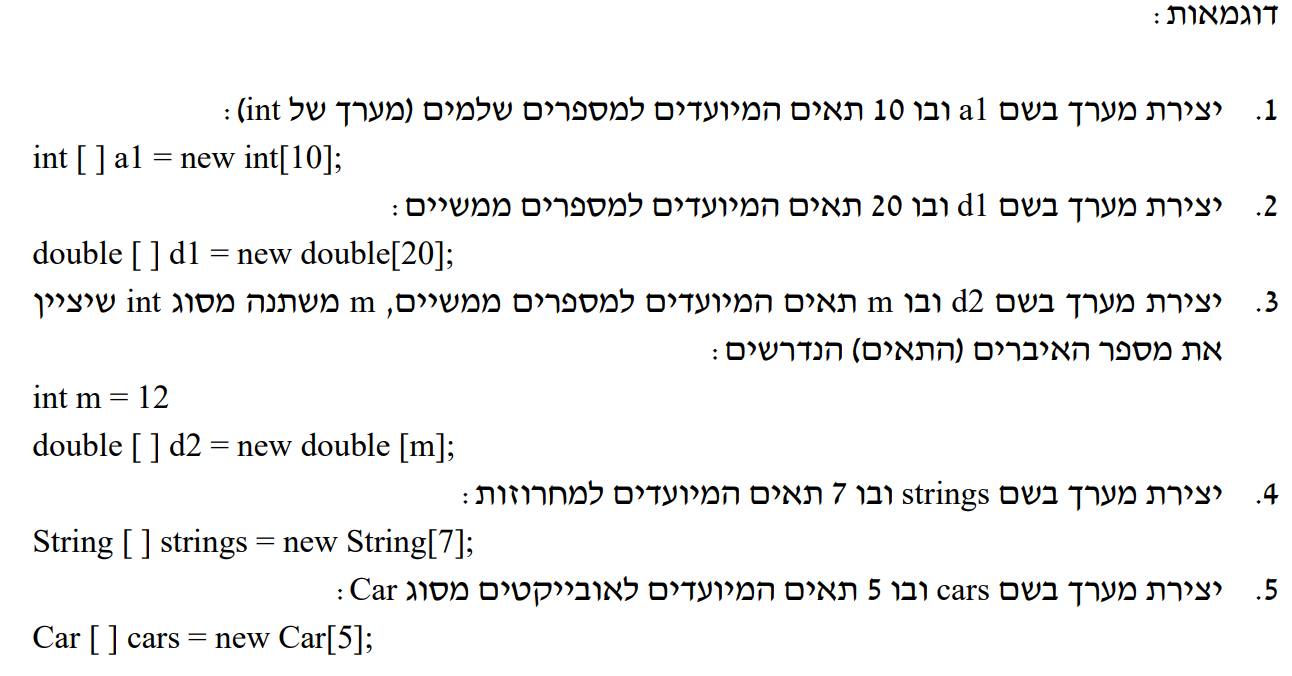






מערך:



תמונה שמכילה תרשים

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

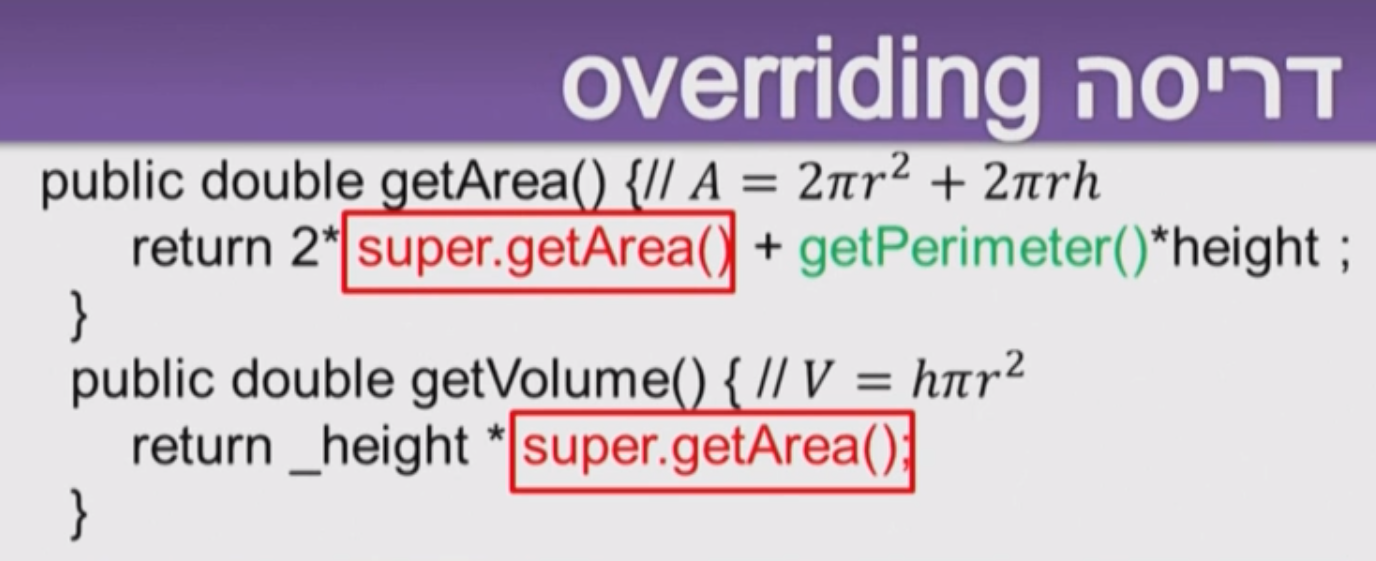
התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

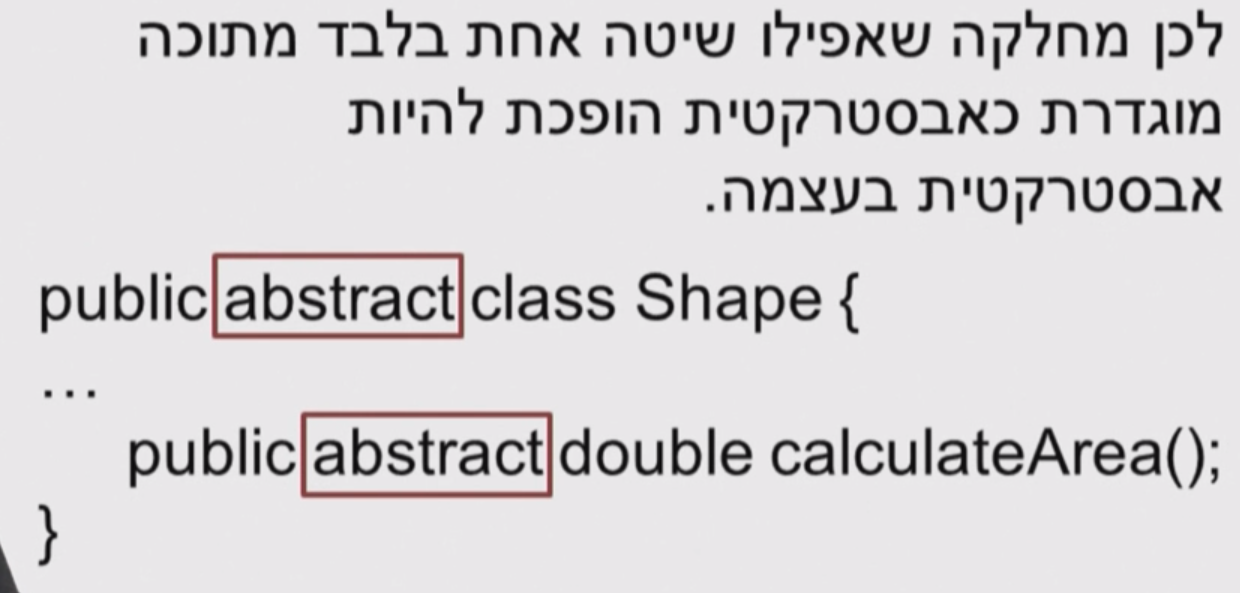
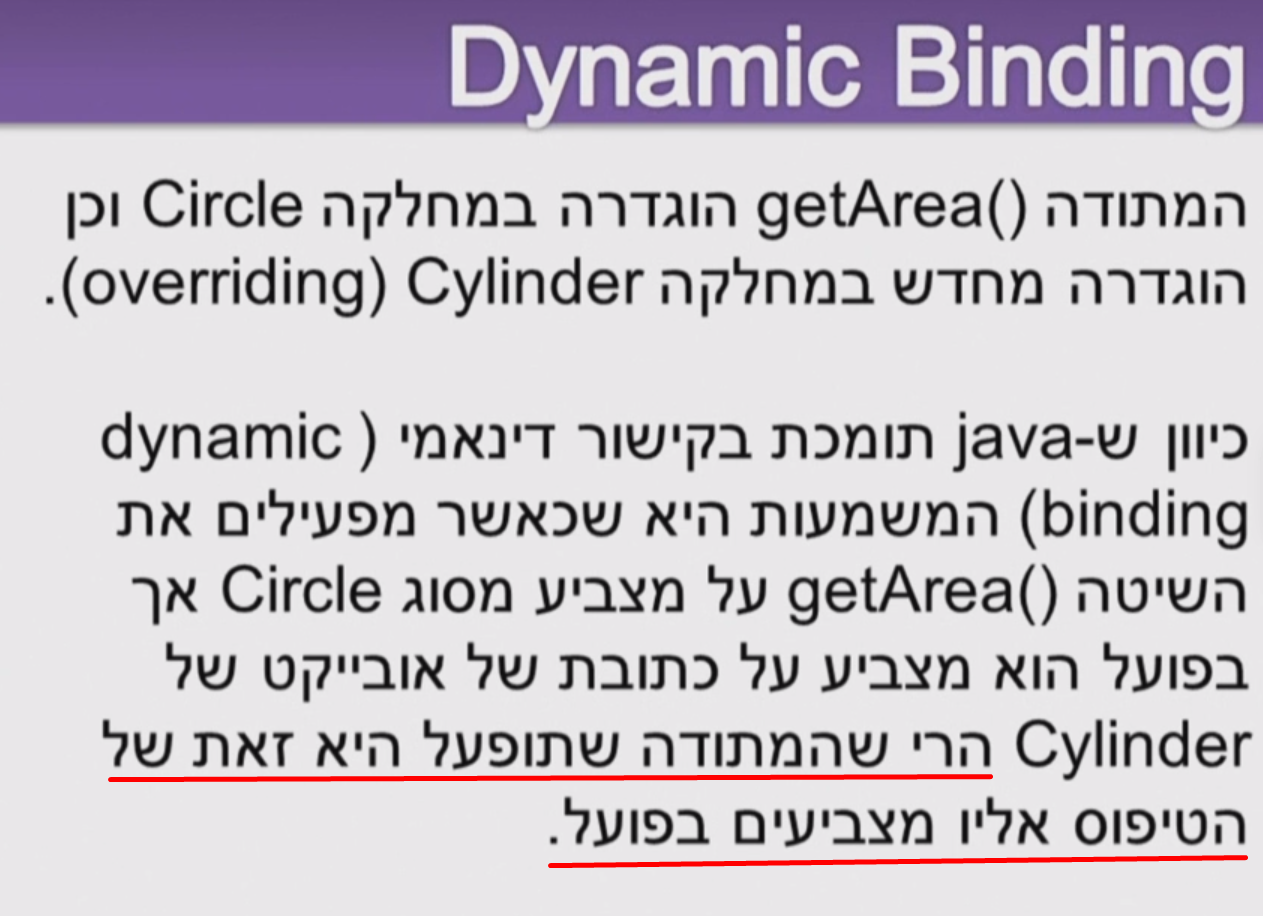
תמונה שמכילה טקסט

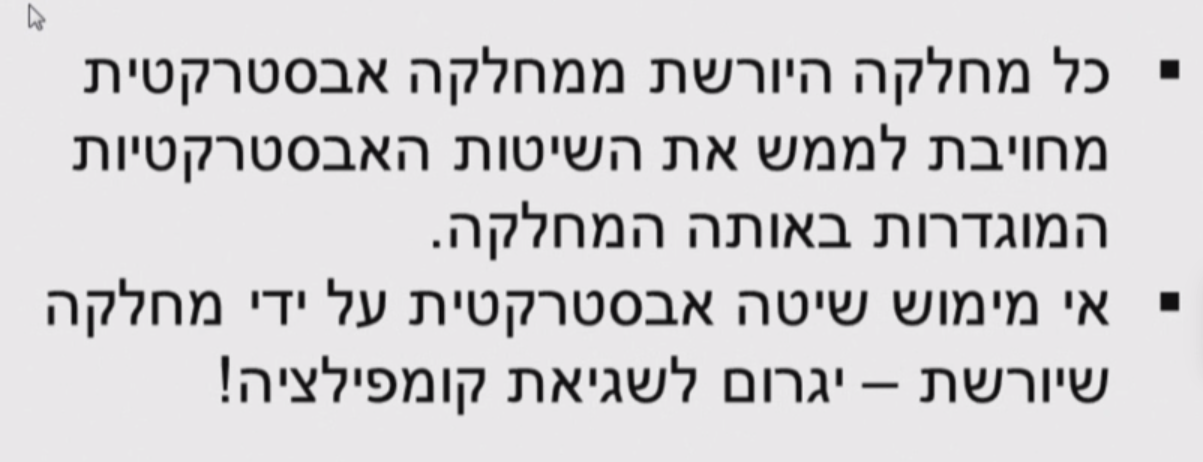
התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט

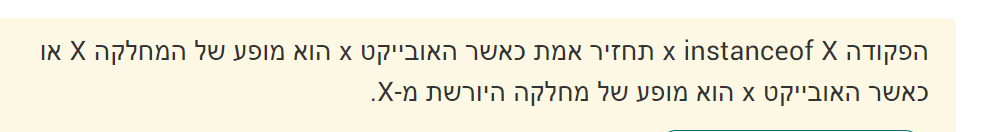
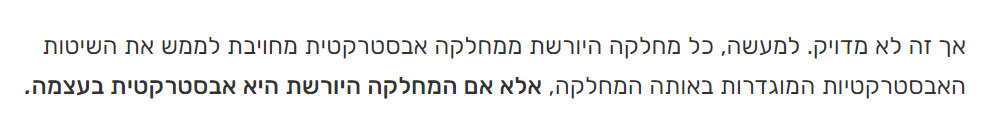
התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי



**פולימורפיזם חוקים**

**1)** כאשר מבצעים המרה בין שתי מחלקות שלהן קשר ירושה ישיר (כלומר, שתיהן נמצאות על אותו ענף בעץ הירושה), ההמרה חוקית ומתקמפלת.

אך בזמן ריצה נוצר אובייקט מסוג GrandFather ואין אפשרות להמיר אותו ל-Father מכיוון שלאובייקט מטיפוס GrandFather אין את התכונות והשיטות בזכרון שיאפשרו לו להפוך ל-Father. תתקבל שגיאת ריצה.

קוד שנכתב כך:

;אב = צאצא

אינו מתקמפל ונדרשת המרה (casting) כדי שהקוד יתקמפל:

;אב(צאצא) = צאצא

קוד זה ירוץ רק אם יהיו לאב את התכונות והשיטות הנחוצות כדי להפוך לאובייקט מסוג צאצא

**2)** כאשר ההצבעה מתבצעת כך:

;צאצא = אב

אין צורך בהמרה מפורשת, ההמרה מתבצעת באופן אוטומטי ותמיד תתקמפל

**3)** אין אפשרות לבצע המרה בין שתי מחלקות שאין בניהן קשר ירושה ישיר (כלומר, הן לא נמצאות על אותו ענף בעץ הירושה).

4) כאשר יש שיטות דורסות במחלקות, מופעלת השיטה הדורסת האחרונה בהיררכיית המחלקות, זו שקרובה ביותר למחלקה.

5) מצביע לאובייקט יכול להיות מטיפוס ממשק ולא רק מטיפוס מחלקה.

[interface] z = new [class]();

6) הפעלת שיטה תלויה במצביע של f1, באובייקט של f1, ובמצביע של f2.

f1.equals(f2);

) לאובייקט שמתקבל כפרמטר אין חשיבות בבחירת השיטה המופעלת, רק למצביע שמתקבל כפרמטר.)

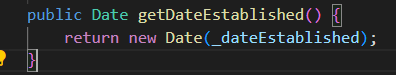
7) האובייקט f2 הוא מטיפוס Son ולא ניתן להמיר אותו להיות GrandSon, לכן תתקבל שגיאת ריצה.

Father f1 = new GrandSon();

Father f2 = new Son();

f1.equals((GrandSon)f2);

**דגשים לבדיקה**

* אות גדולה לשם מחלקה
* משתנה סטטי סופי יכול להיות ציבורי : public static final int PRICE = 5;
* **לבדוק שאובייקטים לא ריקים כאשר מפעילים עליהם שיטות**
* מימוש תוכנית ייחשב טוב יותר אם הוא יהיה ללא שימוש בפקודה break או return מתוך גוף הלולאה.
* השוואת אובייקטים ומחרוזות רק עם equals
* כאשר יש מערך שמכיל אובייקטים ומפעילים עליהם שיטות חייב לבדוק שהם לא null בוודאות
* קבועים
* תיעוד
* השוואת דאבל היא בעייתית
* חלוקה בעשר
* בget להחזיר עותק של אובייקט 
* להימנע מביטוי מפגר כמו: 

