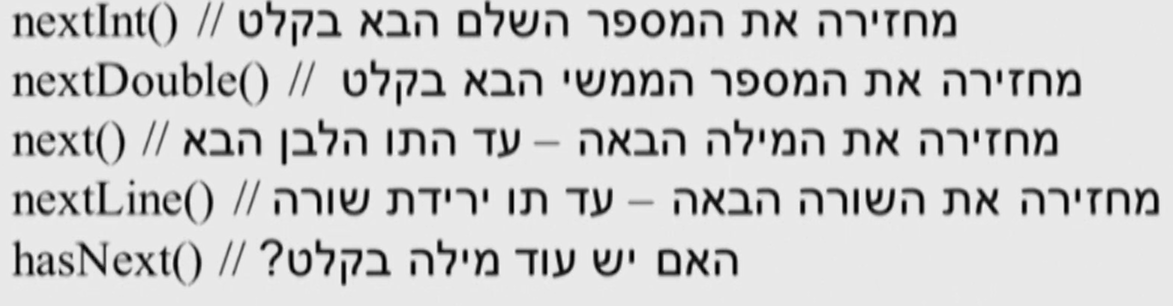
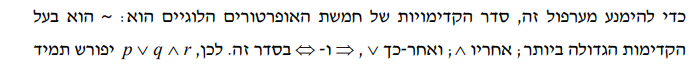
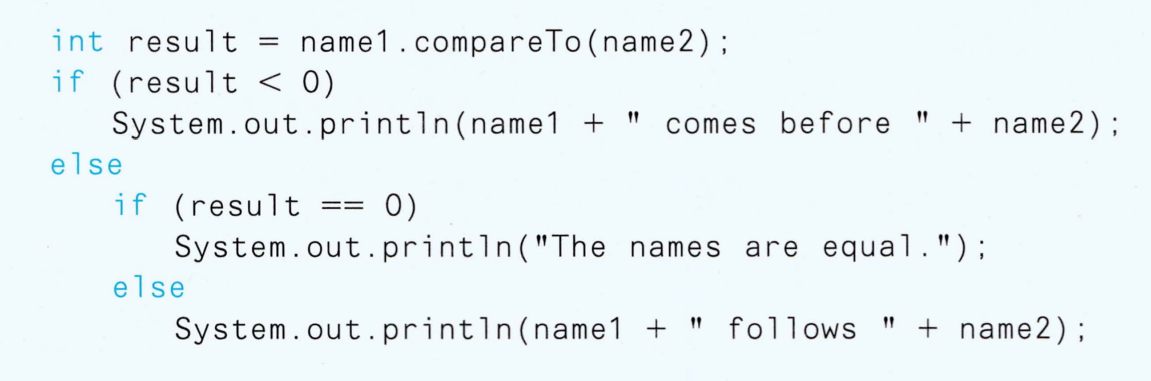
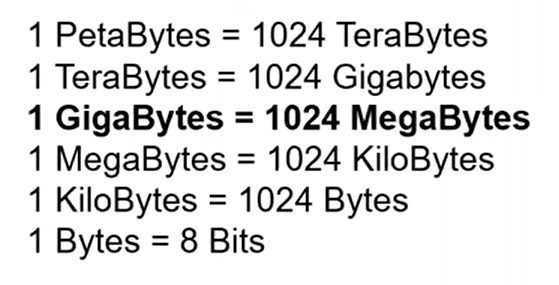
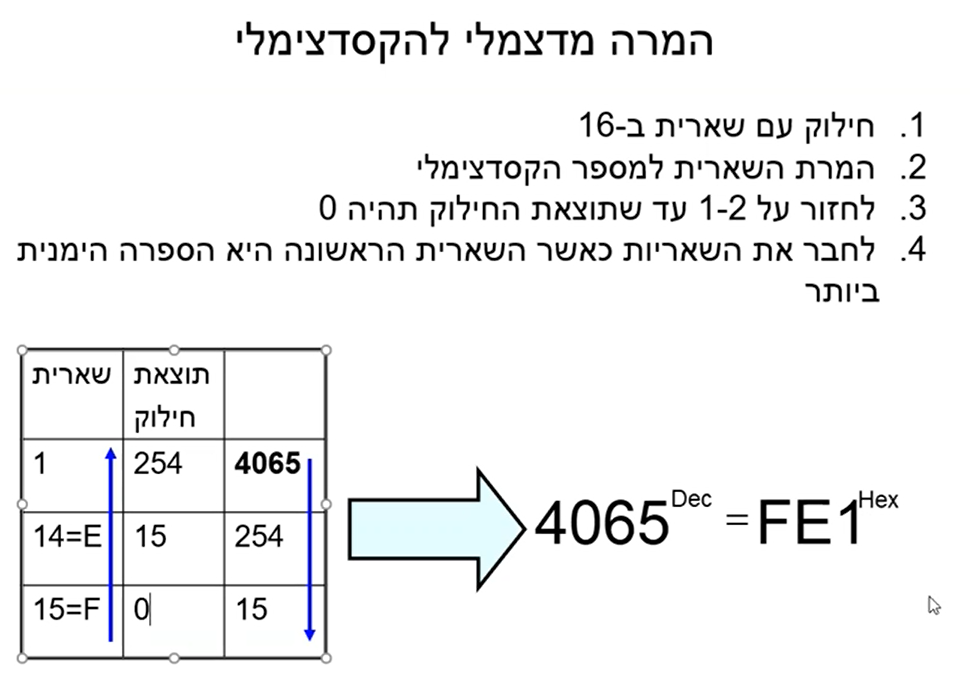
­­­­­Scanner:



! > && > ||

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



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

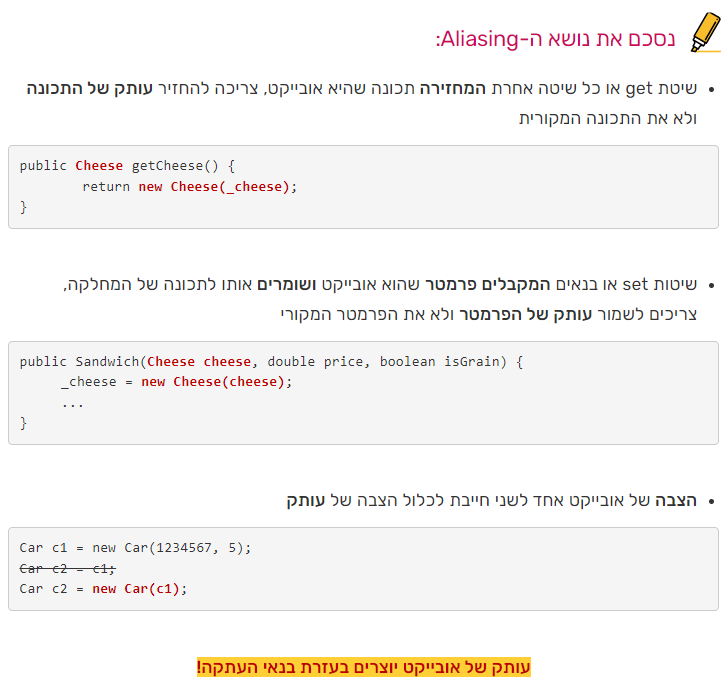
**משווים אובייקטים עם .equals("")**

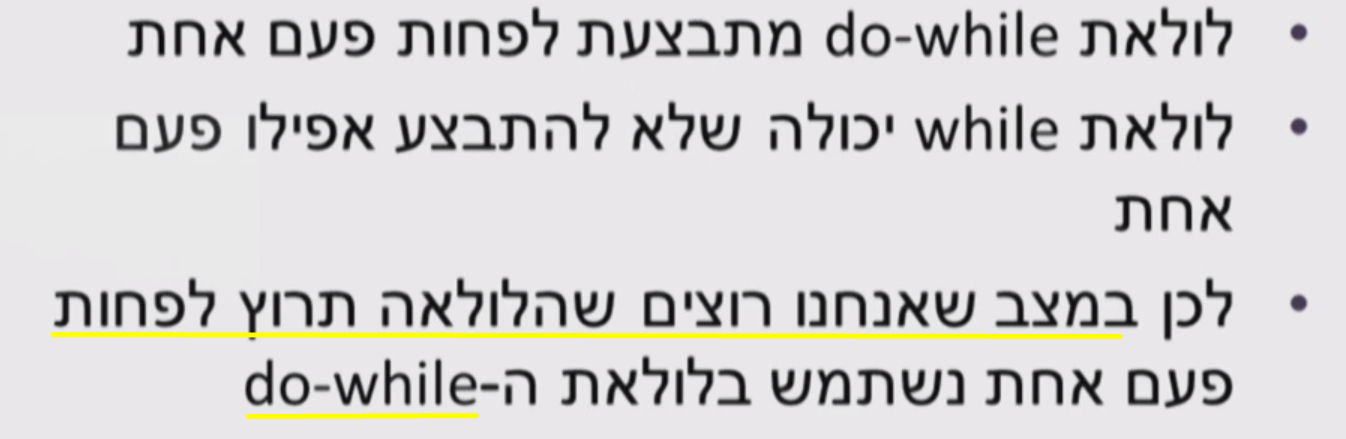
X = (…)? 3 : 7;

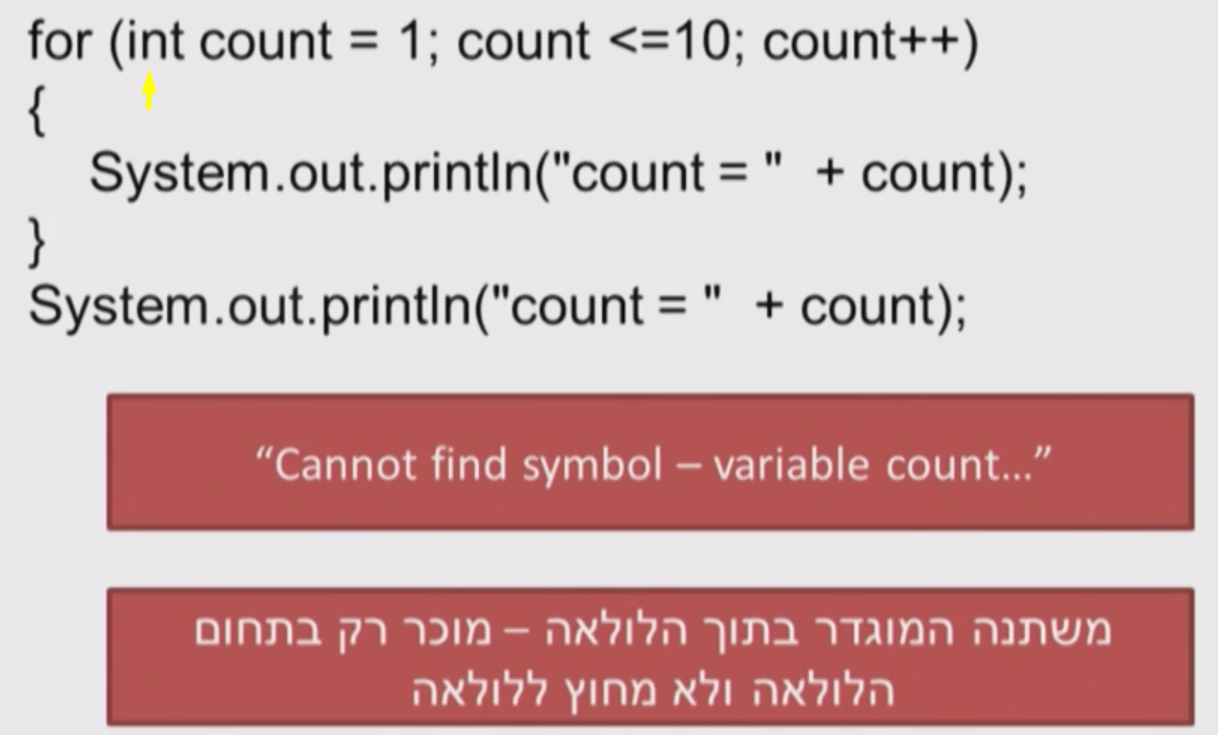
          System.out.println("Flight Elal1 is: " + (elal1.full()?"full":"not full"));

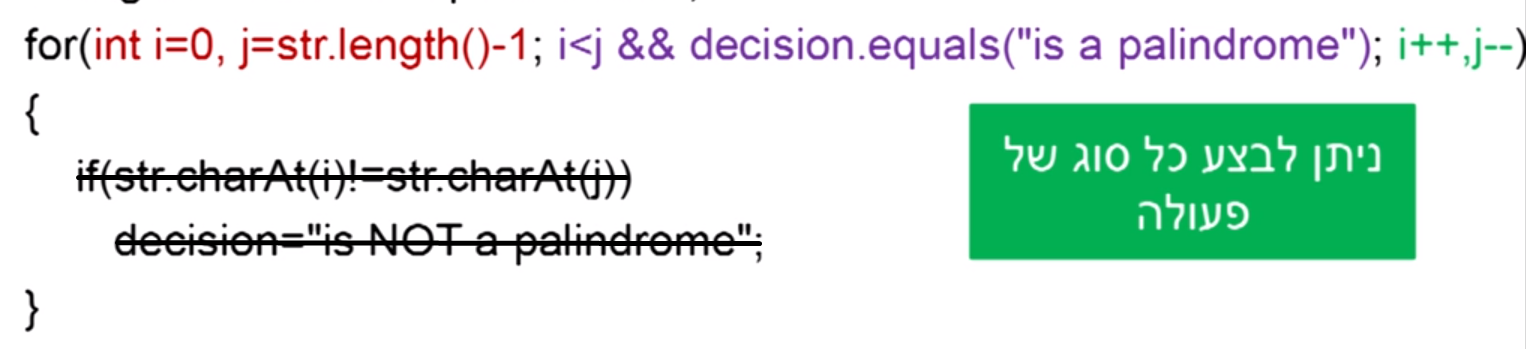
תמונה שמכילה טקסט

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.

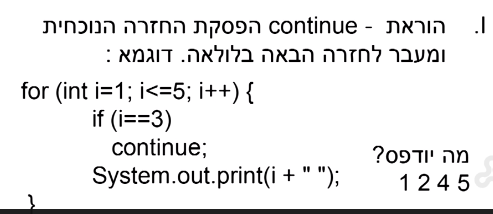
\* @param height the height of the box, must be at least 1.

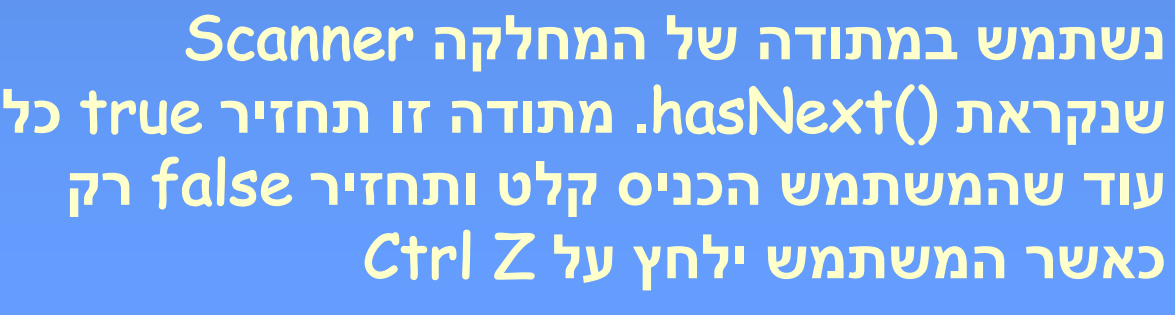
\* if value will be less than 1, it will be changed to 1.

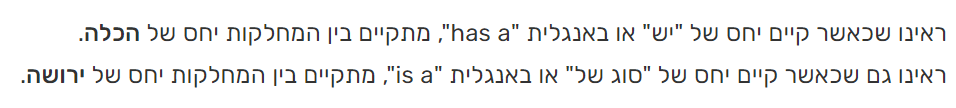
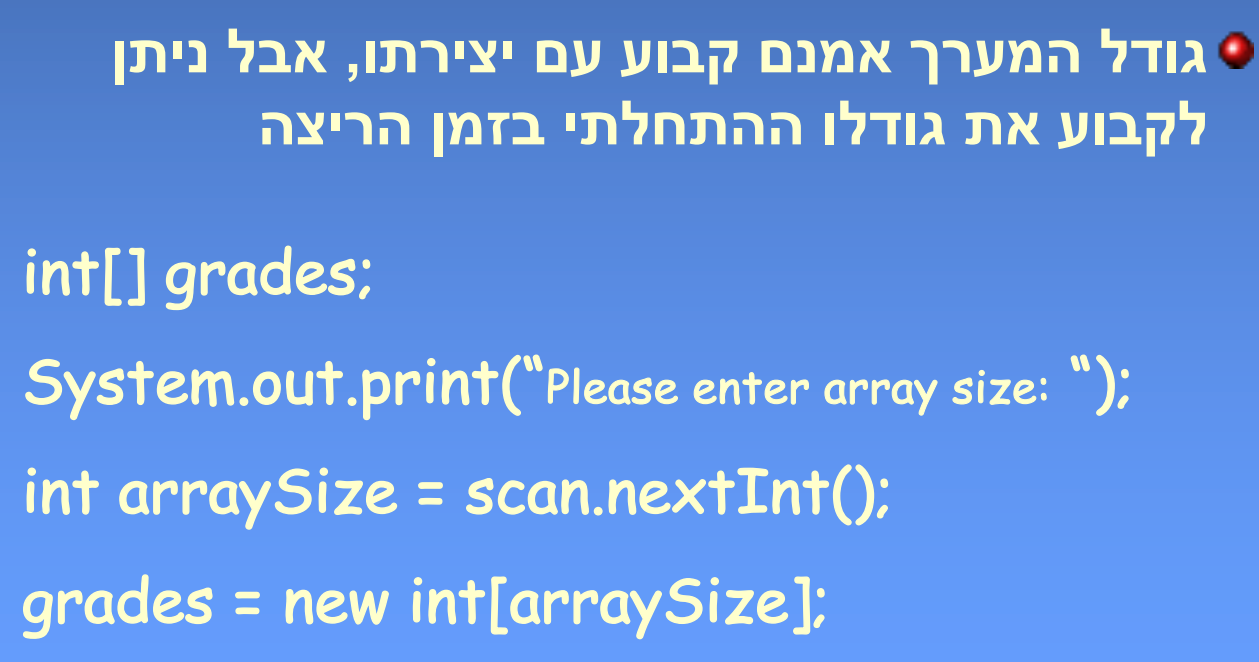
\*/

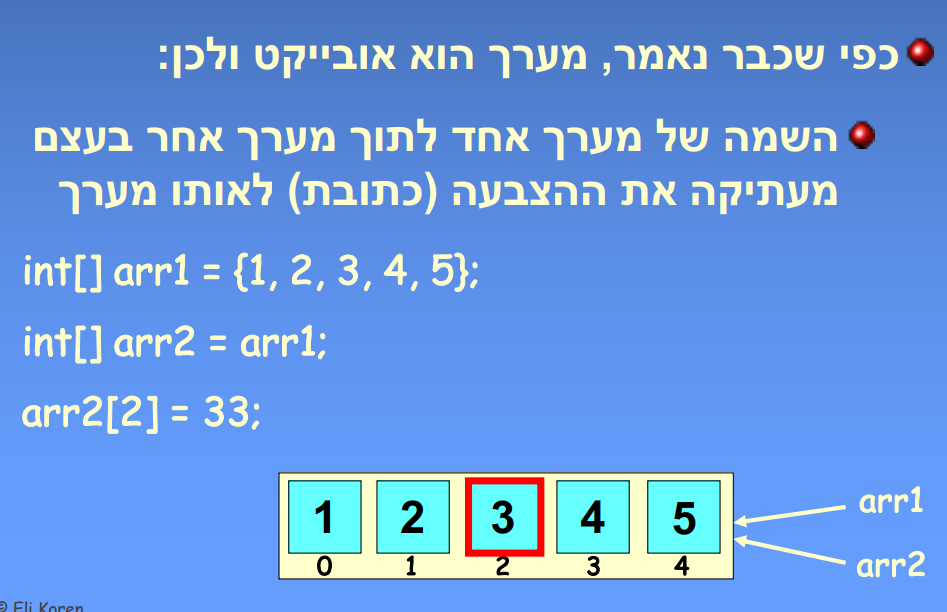
תמונה שמכילה טקסט, מכתב

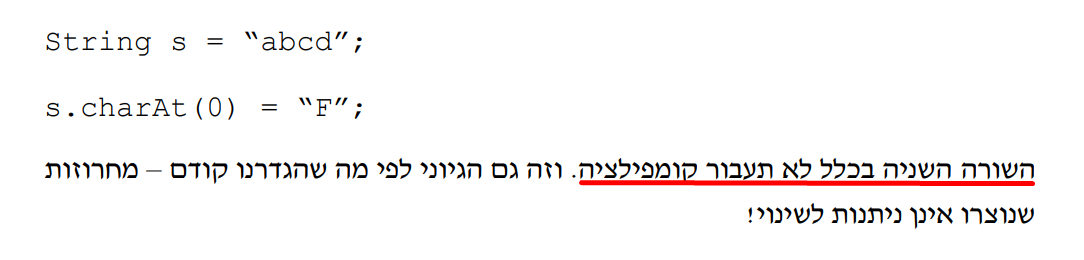
התיאור נוצר באופן אוטומטי





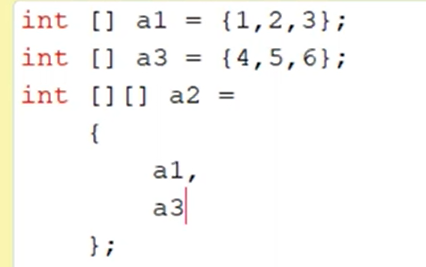






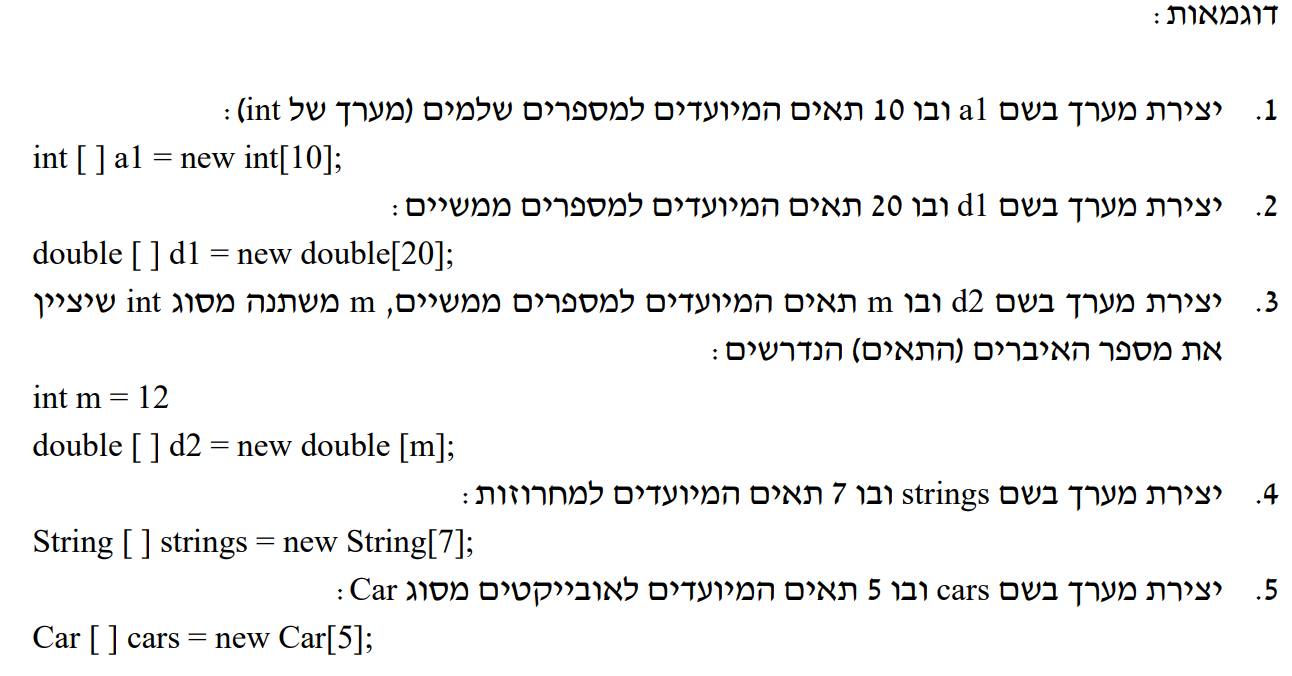
מערך:





כאשר מאתחלים מערך דו מימדי אפשר את מספר המערכים שהוא יכיל מבלי לקבוע את אורכם

Int [][] a = new int[1][]

תמונה שמכילה תרשים

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

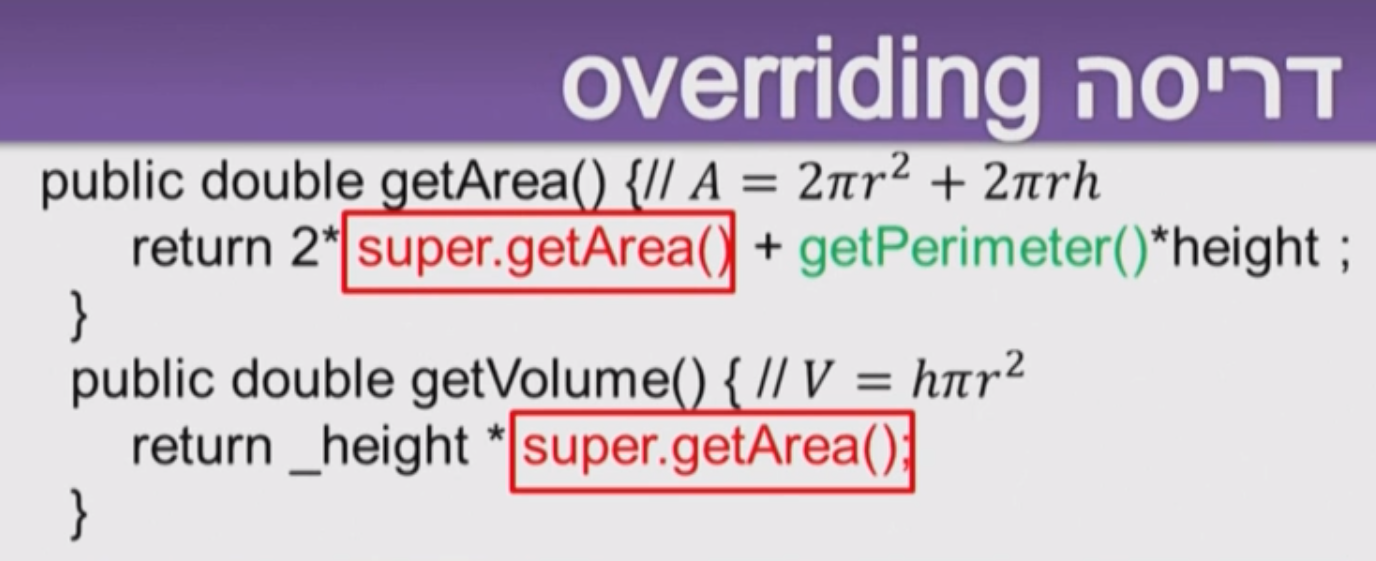
התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

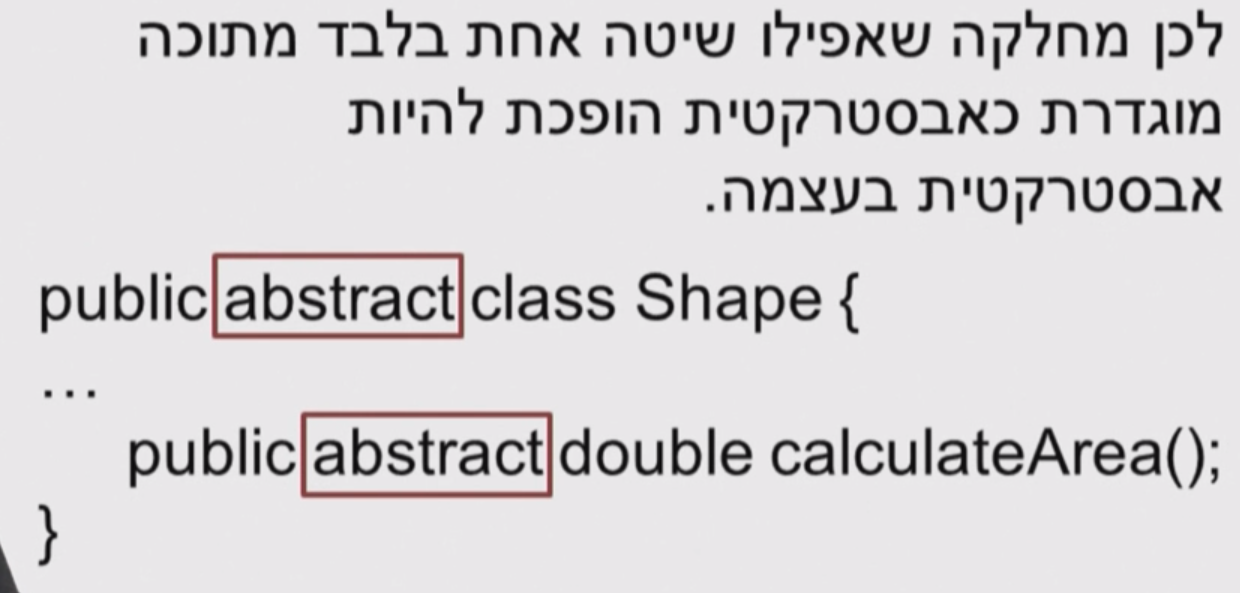
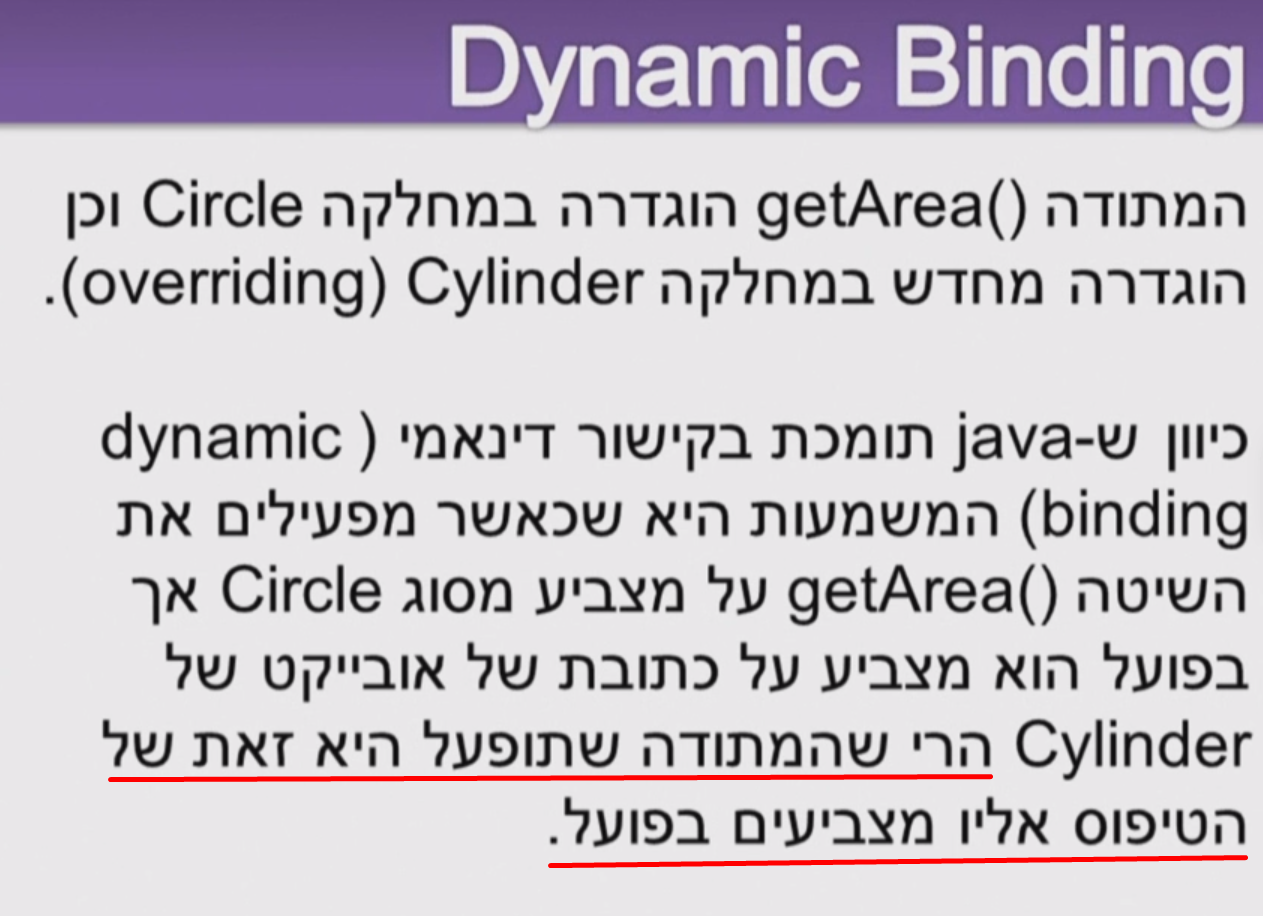
תמונה שמכילה טקסט

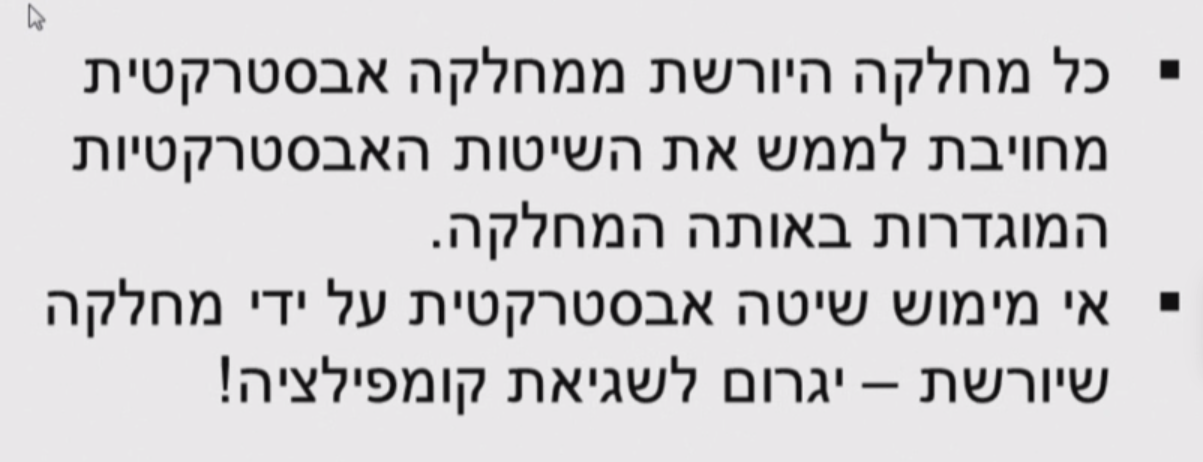
התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט

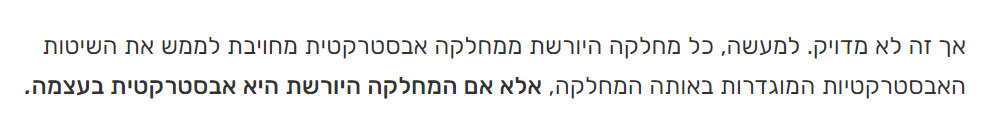
התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

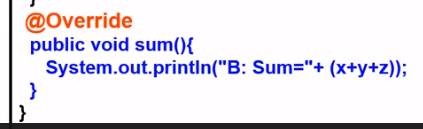
התיאור נוצר באופן אוטומטי



תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

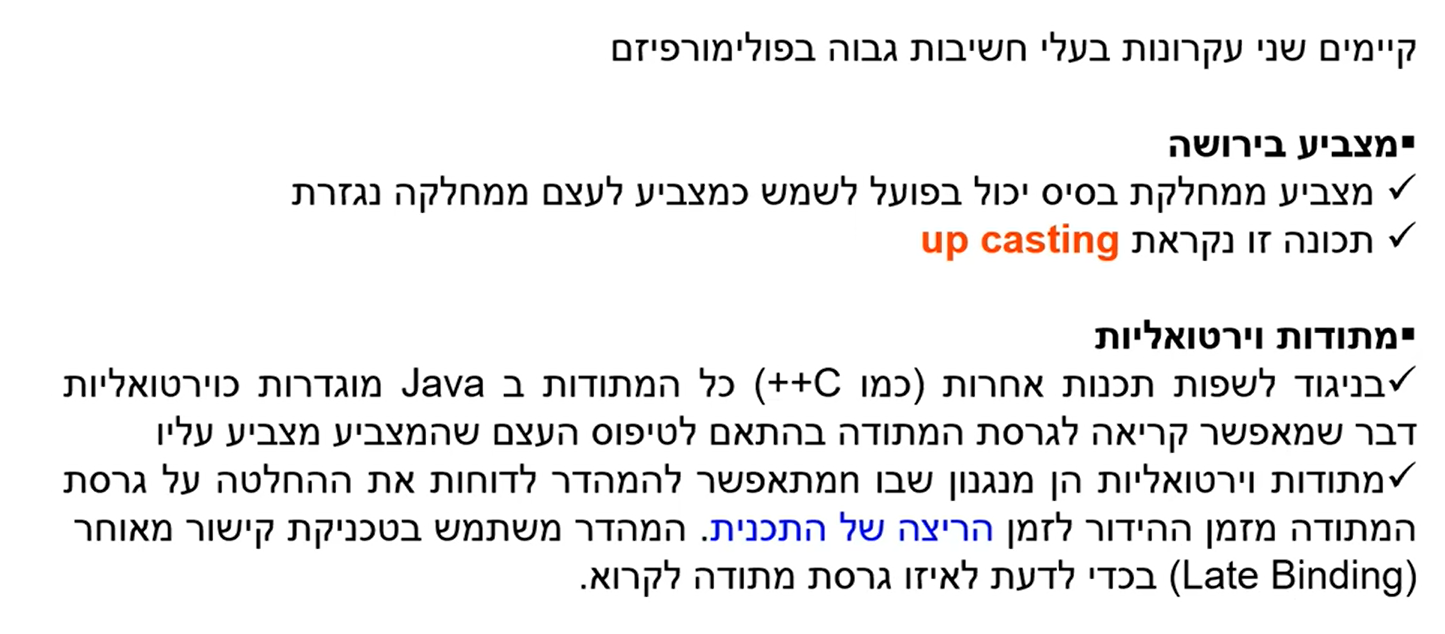
**7 ירושה – החוקים של שאדי:**

1. כל מחלקה יורשת באופן ישיר או עקיף מobject
2. במחלקה היורשתבשורה הראשונה של כל בנאי צריך להפעיל בנאי של מחלקת האב ממנה יורשים (super)
3. במידה ולא נכתבה באופן מפורט קריאה לבנאי של מחלקת האב java תבצע קריאה אוטומטית לבנאי הריק של מחלקת האב super()
4. במידה ו java פנתה לבנאי הדיפולטיבי והוא לא קיים, תתרחש syntax error שגיאת קומפילציה
5. סדר הביצוע של הבנאים הפוך מסדר הקריאות
6. כאשר במחלקה לא מוגדר שום בנאי – java מגדירה בנאי דיפולטיבי ריק לא מקבל כלום ולא עושה כלום
7. כאשר במחלקה מוגדר בנאי כלשהו – java לא תגדיר את הבנאי הדיפולטיבי
8. מחלקה שיורשת ממחלקה אחרת תוכל להשתמש גם בשיטות וגם במשתנים שלה בהתאם להרשאות גישה
9. @override מנגנון של גאווה בודק האם השיטה דורסת שיטה שהתקבלה בירושה - זהה בפרמטרים, בשם ואפילו בסוג הערך המוחזר.
10. במצב של override גאווה תשתמש בשיטה העדכנית ביותר של השיטה
11. תכנות ושיטות בגאווה הן באופן דיפולטיבי "משתני מקרה" ו"שיטות מקרה" – מיוחדות עבור כל אובייקט . כדי ליצור שיטה או משתנה משותף נקדים אותו בהגדרה static
12. משתנים ושיטות סטטיים של מחלקה קיימים הרבה לפני יצירת האובייקטים ולפני משני המקרה. סטטיים קיימים בהקשר של המחלקה ולא של אובייקט.
13. שיטות אינסטנס (פר אובייקט) יכולות לפנות לשיטות ולמשתנים סטטיים
14. שיטה סטטית לא יכולה לפנות בצרוה ישירה לשיטה או למשתנה מסוג אינסטנס (היא קיימת לפניהם )
15. גם שיטות וגם משתנים סטטים עוברים בירושה למחלקות נגזרות – בהתאם להרשאות
16. דריסה ושיטות סטטיות: לא ניתן לדרוס שיטה סטטית בכלל

* לא ניתן לדרוס שיטה סטטית ע"י שיטה אינסטנס
* לא ניתן לדרוס שיטה סטטית ע"י שיטה סטטית – זה חוקי אך זו לא דריסה אלא red-define או hiding

1. שיטה סטטית לא יכולה לדרוס שיטה לא סטטית
2. מתוך שיטה *סט טית* לא ניתן לגשת למשתנה/לשיטה ע"י שימוש בsuper - או ב- this
3. מתוך שיטת *אינסטנס* ניתן לגשת למשתנה/לשיטה סטטית ע"י שימוש בsuper - או ב- this

**8 פולימורפיזם חוקים**



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

**2)** כאשר ההצבעה מתבצעת כך: DOWN CASTING

אב = צאצא;

הקוד אינו מתקמפל, נדרשת המרה (casting) כדי שהקוד יתקמפל:

אב(צאצא) = צאצא;

קוד זה ירוץ רק אם יהיו לאב את התכונות והשיטות הנחוצות כדי להפוך לאובייקט מסוג צאצא (אחרת שגיאת ריצה)

**2)** כאשר ההצבעה מתבצעת כך: UP CASTING

;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);

8) חשוב לזכור שבשיטות יש קישור דינמי, ואילו בתכונות אין קישור דינמי.

התכונה הנבחרת היא לפי טיפוס המצביע, השיטה הנבחרת היא גם לפי טיפוס המצביע וגם לפי האובייקט בזכרון, כפי שתרגלתם בסעיף הקודם.

9) בשיטות סטטיות אין קישור דינמי – תופעל השיטה לפי סוג המצביע

הערות:

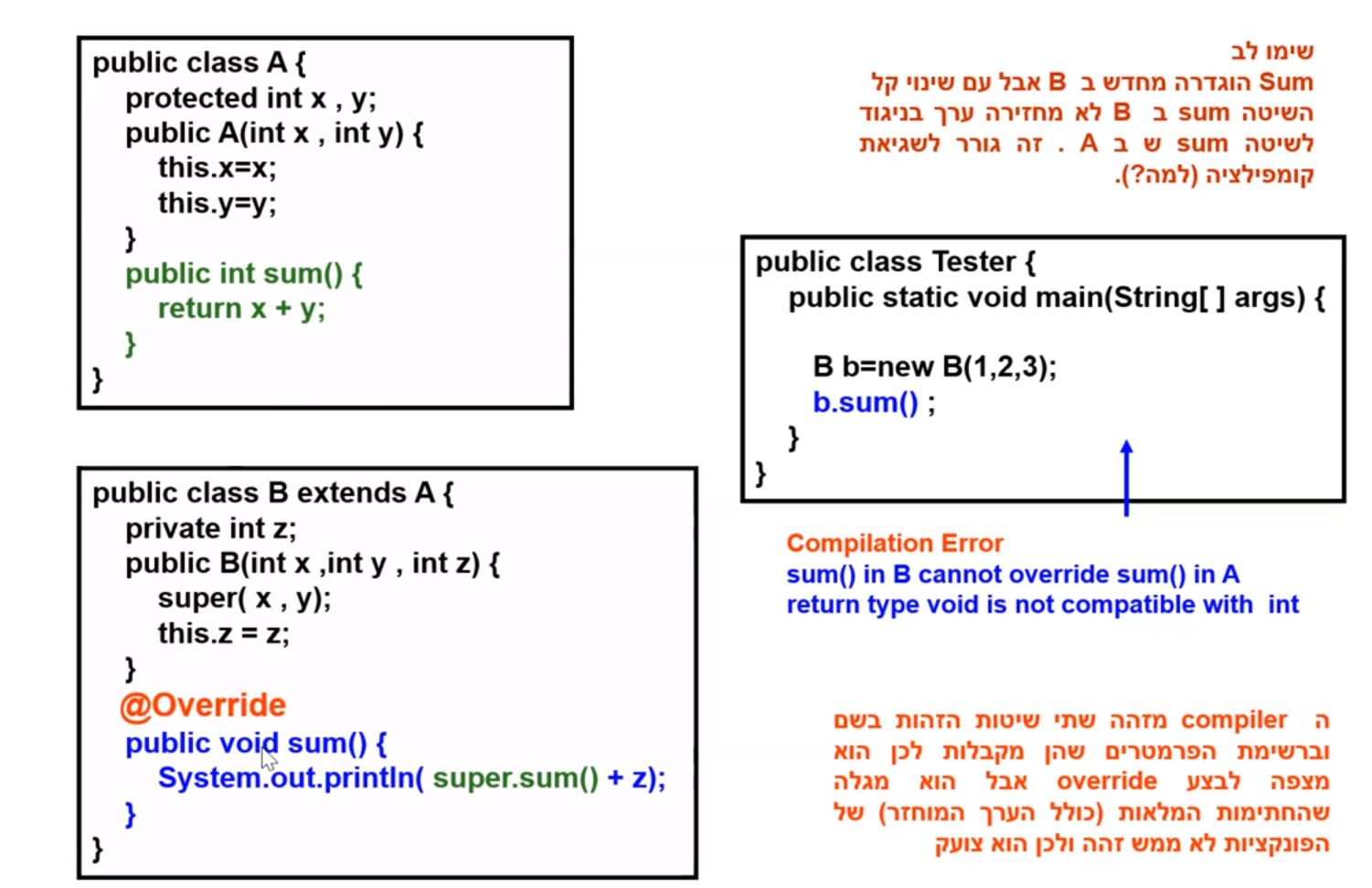
* כל מחלקה יורשת מobject באופן ישיר או עקיף
* בכל בנאי של המחלקה היורשת חייב לקרוא לאחד הבנאים של המחלקה מעל super

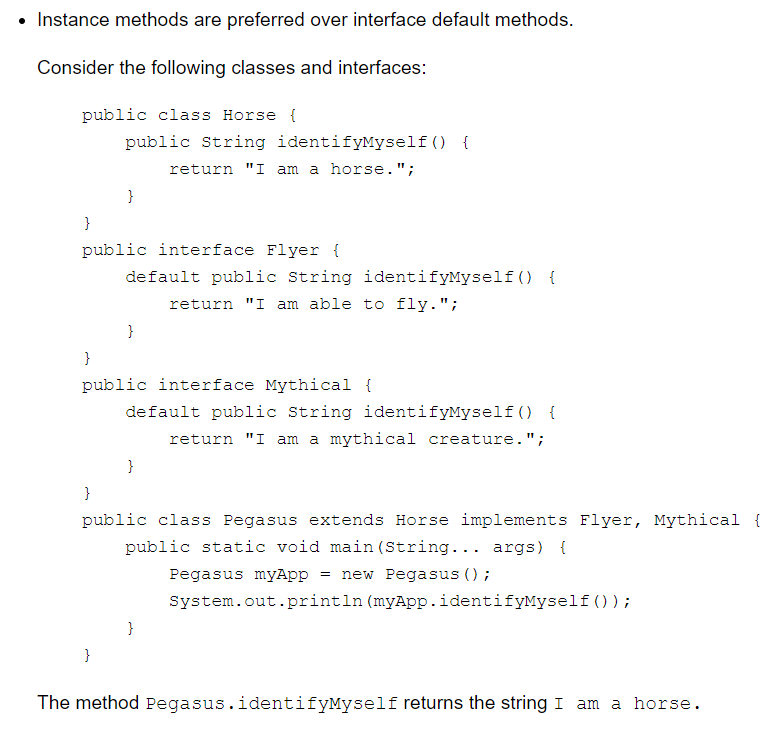
– אחרת תתבצע קריאה אוטומטית לבנאי הריק super()

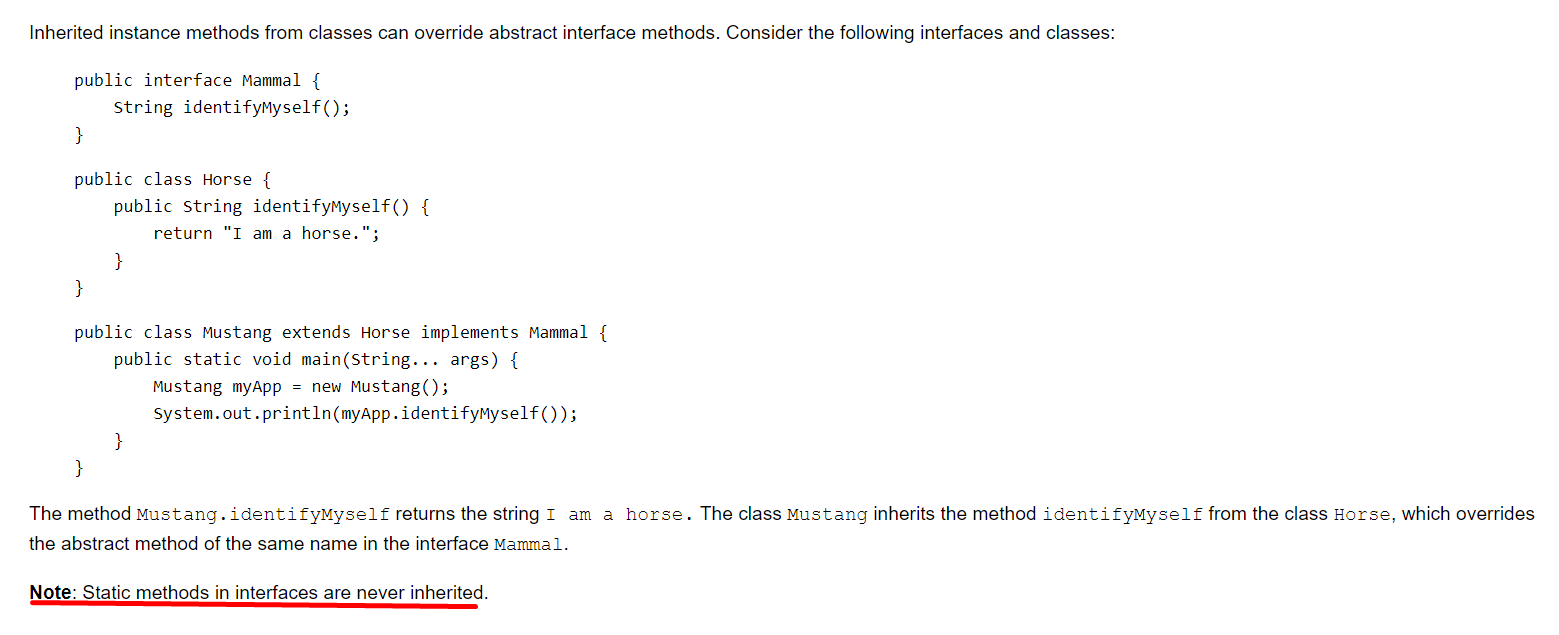
– אם לא קיים הבנאי הריק תתרחש שגיאת קומפילציה

* סדר הקראות לבנאים הפוך מסדר הביצוע
* מחלקה יורשת יכולה להשתמש בשיטות של מחלקת האב לפי הרשאות השימוש



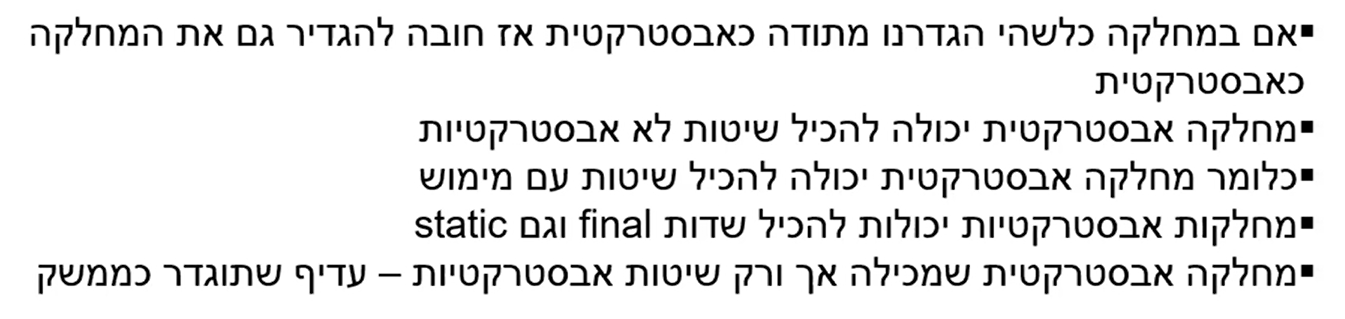


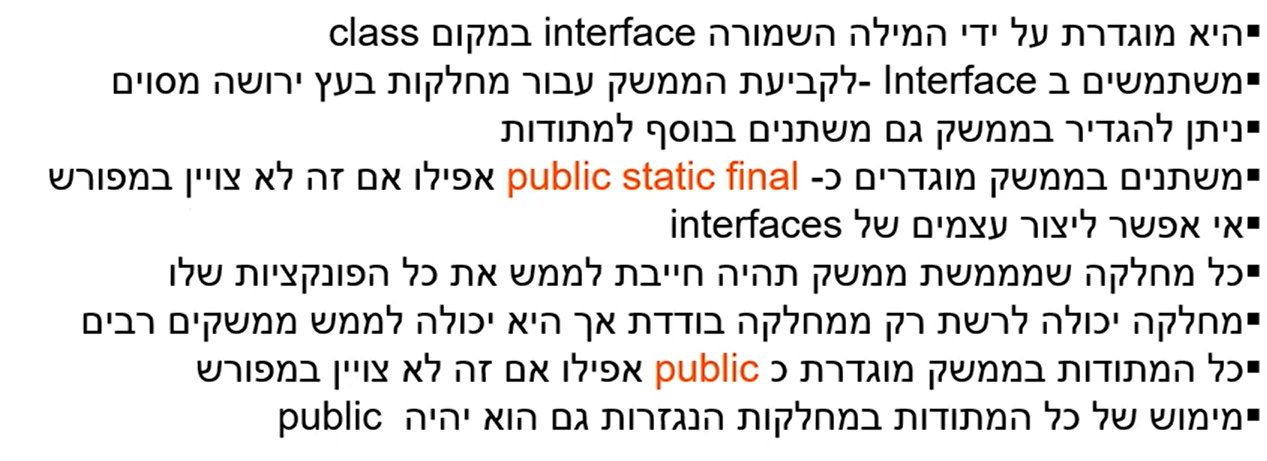




**יחידה 8 פולימורפיזם – שאדי**

1) אם במחלקה מוגדרת שיטה אבסטרקטית – כל המחלקה הופכת אבסטרקטית.





**יחידה 9 סיבוכיות**

אין חשיבות לקבועים, ולכן נותר לבדוק רק את הלולאות.

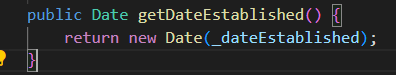
* בלולאות יש לבדוק רק את מספר חזרות הלולאה כתלות ב־ nואין חשיבות למספר הפעולות הבסיסיות בגוף הלולאה.
* לעומת זאת, אם בגוף הלולאה מופיעה פעולה מורכבת, כמו פנייה לשיטה אחרת או לולאה נוספת )לולאה מקוננת פנימית(, אזי יש להביא זאת בחשבון ולספור את הפעולות במפורש.
* בפולינומים החזקה הגבוהה היא שקובעת את הסיבוכיות. הסיבה לכך היא שעבור n־ים גדולים החזקות הנמוכות זניחות.­­
* שבתוכנית בה אין לולאות כלל, אין צורך לספור את הפקודות )הקבועים) וניתן להסיק מיידית שהסיבוכיות היא.O(1)

מספר הספרות של מספר הוא לוג לפי בסיס 10 של המספר

סיבוכיות מקום –

סיכום סיבוכיות המקום תהיה (1(O בכל שיטה שבה מוגדרים משתנים פרימיטיביים שמספרם אינו תלוי בקלט, אפילו אם הם רבים. סיבוכיות המקום תהיה שונה מ־(1(O כאשר מספר תאי הזיכרון המוגדרים בשיטה תלוי בקלט.

**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: 

