טיפים למבחן ב- OOP:

1. כדי לבצע המרה מ-String ל- Double עושים:

double number = Double.parseDouble(str)

המרה מ-String ל-Integer:

int number = Integer.parseInt(numberStr);

1. בחלוקה בין מספרים לזכור לעשות casting ל-float למונה או למכנה.
2. רשימה ב-Java לא ניתנת לשינוי על ידי slicing רגיל, צריך לייצר copy שלה באמצעות

Arrays.copyOfRange(items, 0, size)

1. UML

תמונה שמכילה טקסט, גופן, צילום מסך, לבן

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, גופן, לבן, טיפוגרפיה

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

**ההבדל** בין הכלה לבין תלות היא שבהכלה אנחנו שומרים את האובייקט של המחלקה אותה אנו מכילים כמשתנה פרטי (שדה) של המחלקה שמכילה אותו. כאשר יש לנו תלות בפונקציה אחת, היא רק תשתמש בו כארגומנט ולא תשמור אותו במחלקה.

תמונה שמכילה טקסט, צילום מסך, קו, תרשים

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

1. Singleton

לזכור שכאשר אנחנו כותבים את השיטה getInstance אז לעדכן את השדה שמחזיק את ה-Singleton במידה ומייצרים סינגלטון חדש:

תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטי

כאן אנחנו מעדכנים את logger בערך חדש של Logger במידה והוא NULL.

1. עקרונות:
2. **עקרון הבחירה היחידה** – אם יש משהו בתוכנית שלנו עבורו תיתכנה מספר אפשרויות אז הרשימה המלאה של האפשרויות הללו תופיע רק במקום אחד.
3. **עקרון הפתוח סגור** – פתוח להוספה של יכולות חדשות אבל לא לשינוי של דברים שכבר משתמשים בהם.
4. **עקרון השיקוף** – לא קיים!!!!!!!
5. **עקרון העדפת הכלה על פני ירושה** –

ירושה מאפשרת לנו מיחזור קוד וגם פולימורפיזם.

ממשק – מאפשר רק פולימורפיזם.

הכלה מאפשר רק מיחזור קוד.

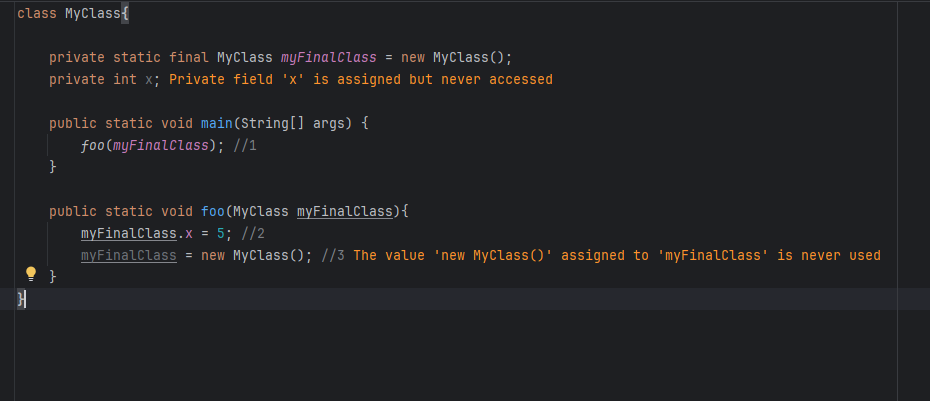
כללי אצבע:

1. קוד משותף על ידי שצריך להיות במימוש כל המחלקות – פולימורפיזם שימוש בירושה.
2. אין קוד משותף בין המחלקות – ממשק.
3. שימוש נקודתי בקוד משותף בין מחלקות – הכלה.
4. **עקרון האחריות היחידה** – לכל מחלקה צריכה להיות אחריות אחת בלבד.
5. **עקרון ה discoverability** – נשים את השיטה במקום שבו היינו מחפשים אותה.
6. **עקרון תכנות לממשק ולא למימוש** – תמיד נשאף לכתוב קוד עבור הטיפוס הגבוה ביותר ועם המטרה הכללית ביותר.
7. **עקרון הפולימורפיזם** – ממשק שמאפשר עבודה עם מספר עצמים שונים.
8. מועד ב' שנה שעברה

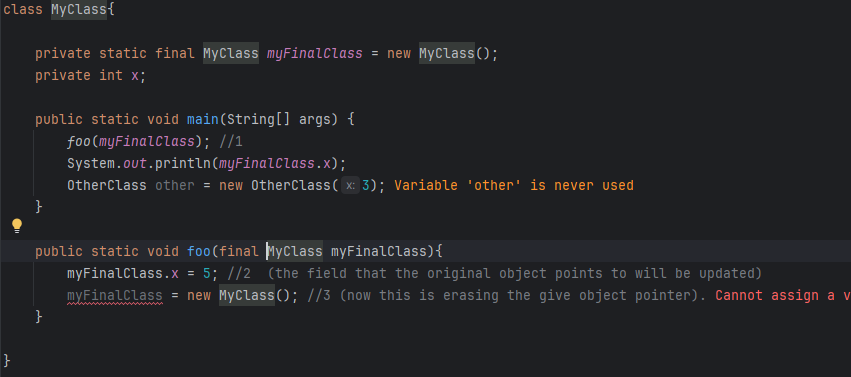
שאלה 5:

נבחין שהארגומנט myFinalClass מועבר לפונקציה foo ולכן כעט המופע שלו בפונקציה הוא מקומי, ויעלם לאחר שהוא שונה.

כאשר אנחנו מעבירים משתנה לפונקציה אחרת כאגרומנט אז נוצר העתק של המשתנה, ושינוי שלו יהיו מקומי.



כדי לפתור את הבעיה הזו אפשר להגדיר בשורת הארגומנטים בפונקציה את המקדם final :



ואז נקבל שגיאת קומפילציה.

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

שאלות תכנות:

* לזכור ששיטות סטטיות לא מכילות מופע כמו this כי הן לא משוייכות לאובייקט ספציפי אלא לכולם.
* לזכור שכאשר מבצעים ירושה אז חייבים לייצר קונסטרקטור למחלקה היורשת עם super.
* למחלקה היורשת אין גישה לשדות הפרטיים, אלא רק לשדות הדיפולטיים, הpublic וה-protected.
* לזכור:תמונה שמכילה טקסט, גופן, צילום מסך, לבן

  התיאור נוצר באופן אוטומטי
* מייצרים מחלקה יורשת לשגיאה, לא ממשים שגיאה.

1. מועד א' 2018 (הרבה שאלות לא בחומר)

שאלה 2 ג': **על אף מנגנוני הכימוס (Encapsulation) בשפת java, יכול להתקיים מצב בו מחלקה A, היורשת מ-B, אינה חשופה לחלק ממשתני האובייקט המוגדרים ב-B, אף על פי שמחלקה אחרת, C, אשר אינה יורשת מ-B, כן חשופה אליהם.**

תשובה: **נכון**

הסבר: **ישנן שתי אופציות:**

* + **אם ב-A יש שדות פרטיים ו-C מוגדרת כמחלקה פנימית (סטטית או לא) של A, אז יש ל-C נגישות לשדות אלו של-B אין. שימו לב שמחלקה פנימית אינה מוגדרת כמחלקה היורשת מהמחלקה החיצונית (היא יכולה גם לרשת, אבל יחס "פנימיות" אינו יחס ירושה)**
  + **אם ב-A יש שדות ללא modifiers (ולפיכך הם חשופים רק למחלקות בחבילה ש-A נמצאת בה) ו-C מוגדרת באותה החבילה כמו A, אבל B לא נמצאת שם, אזי C חשופה לשדות אלו ו-B איננה.**

שאלה 3:

תמונה שמכילה טקסט, צילום מסך, גופן, אלגברה

התיאור נוצר באופן אוטומטי

במקרה של shadowing בין משתנים ולא בין מתודות, והמשתנים הם public, אז ההתייחסות היא לפי הtype בהכרזה (כלומר ה reference type) שהוא A בשני המקרים, ולכן הערך שווה.

שאלה 9: כאשר הארגומנט הוא <? Super A> אז אפשר לאתחל רק עם סוגים שהם Super ל-A כלומר ש- A יורשת מהם.

כאשר הארגומנט הוא <? Extends A> אז אפשר לאתחל רק עם סוגים שהם יורשים של A.

מותר גם לאתחל ללא שום סוג אבל נקבל אזהרה (לא שגיאה) בקומפילציה.