טיפים לקראת הבחינה

זהו קובץ המיועד להכיל אוסף של טיפים לקראת הבחינה שנכתב בידכם ובשבילכם!
תוסיפו כל דבר העולה בדעתכם שיכול לעזור לסטודנטים שבאים אחריכם.
איך מוסיפים? (1) מורידים את הקובץ, (2) מוסיפים (ולא מוחקים משהו קיים), (3) מעלים קובץ חדש לתיקייה כשבשם כתובה גרסה גבוהה יותר ממה שהורדתם.

1. מומלץ לבצע דיבאג על לפחות שאלת פולימורפיזם / הורשה אחת על מנת להבין באמת לעומק מה קורה בזמן ריצה.
2. אפשר לזכור את ההדפסות $preOrder, inOrder, postOrder$ לפי:

$preOrder$	$inOrder$	$postOrder$
$\mathbf{mid} \rightarrow left \rightarrow right$	$left \rightarrow \mathbf{mid} \rightarrow right$	$left \rightarrow right \rightarrow \mathbf{mid}$

3. הדפסת $inOrder$ של עץ חיפוש בינארי נותנת לנו מערך ממוין.
ואילו ההדפסות $preOrder, postOrder$ לא נותנות לנו סדר מיון מיוחד.
4. לא ניתן לשחזר עץ חיפוש בינארי מהדפסת $inOrder$. דוג' נגדית:

5. ניתן לשחזר עץ חיפוש בינארי מהדפסת $postOrder$ או $preOrder$.
ממיינים את ההדפסה על מנת לקבל הדפסת $inOrder$
ומכאן ניתן לקבל בעזרת שתי ההדפסות את העץ המתאים.
6. ניתן לשחזר עץ בינארי (לא בהכרח חיפוש) מהדפסת $postOrder$ וגם $inOrder$.
נבנה את העץ ממש כמו סעיף (5).
7. בפולימורפיזם, נחלק את הפעולות שמתקיימות בתוכניות לשני שלבים: (1) שלב הקומפילציה (2) ושלב הריצה.
עבור שורת הקוד:

```
A ab = new B();
```

כאשר: $A \leftarrow B$ (מחלקה B יורשת ממחלקה A)
בזמן הקומפילציה – ab מטיפוס A.
בזמן ריצה – ab מטיפוס B.

עבור שורות הקוד:

```
public class A{
    public void foo(){
        System.out.println("I'm in A");
    }
}
```

```
public class B extends A {
    public void foo() {
        System.out.println("I'm in B");
    }
}
```

```
public class Poly {
    public static void main(String[] args) {
        A ab = new B();
        ab.foo();
    }
}
```

נדגיש כי תמיד מדובר בקבצים שונים תחת אותה תיקייה (או אותה חבילה ($package$)).

קורן בן עזרא

בזמן קומפילציה – ab מטיפוס A, כאשר אנו קוראים לשיטת מופע foo שלו, אנו הקומפיילר מחפש את השיטה במחלקה A. הקומפיילר ימצא את השיטה (עם אותו השם כמובן) המקבלת את הפרמטר הכי מתאים למה שהוא מחפש. במקרה שלנו השיטה מחפשת שיטה בשם foo שלא מקבלת אף פרמטר. יש כזאת לכן לא תהיה שגיאת קומפילציה וניתן יהיה להריץ את התוכנית.
בזמן ריצה – התוכנית תיגש לשיטת מופע foo עם אותה חתימה כמו של שיטת המופע שמצאת בזמן הקומפילציה (השיטה שנמצאת ב A) שהכי קרובה למחלקה B בין המחלקות היורשות.
כלומר אם למחלקה B קיימת שיטה foo עם אותה חתימה בדיוק, התוכנית תיגש לשיטת המופע הנמצאת ב B.

לבסוף יודפס:

```
Output:
I'm in B
```

הפעם נראה מקרה אחר, עבור שורות הקוד:

```
public class A{
    public void foo(Object obj){
        System.out.println("im in A");
    }
}
```

```
public class B extends A {
    public void foo(int num) {
        System.out.println("im in B");
    }
}
```

```
public class Poly {
    public static void main(String[] args) {
        A ab = new B();
        ab.foo(1);
    }
}
```

בזמן קומפילציה – ab מטיפוס A ובקריאה לשיטת המופע $foo(1)$ נחפש את השיטה המתאימה ביותר ששמה foo וגם מקבלת פרמטר מטיפוס int. נשים לב כי כל טיפוס ממנו int יורש מתקבל על הדעת אם לא קיימת שיטה המקבלת int. במחלקה A באמת לא קיימת שיטה כזאת המקבלת פרמטר מטיפוס int אבל כן יש שיטה המקבלת $Object$ וכיוון שכל המחלקות יורשות מ $Object$ ולא קיימת שיטה המקבלת פרמטר הקרוב יותר ל int מאשר $Object$. אז הקומפיילר יבחר בשיטה הנ"ל.

בזמן הריצה – התוכנית תחפש בין המחלקות ש B יורשת מהן עד ל A (כאן יש רק שתיים אבל אם יש יותר אז תחפש בכולן) שיטת מופע המתאימה בדיוק לחתימה של שיטת המופע שנמצאה בזמן הקומפילציה (השיטה של A). נשים לב שהשיטה של מחלקה B בעלת חתימה שונה (מקבל פרמטר מטיפוס int ולא $Object$ כפי שאנו מחפשים) לכן נאמר כי אין עוד שיטה כזאת המתאימה בין המחלקות היורשות שקרובה יותר למחלקה B והשיטה שתרוץ הינה השיטה שנמצאת במחלקה A.

לבסוף יודפס:

```
Output:
I'm in A
```

.8