

שאלה 1:

תשובה: ה' – רק טענות 1+3 נכונות.

נימוק:

טענה 1 ו 3 נובעות מכך שב calcFibForI השדה lastCalculated נדרס תמיד עם הערך של n, מקום לקחת את הערך המקסימלי של שניהם.  
לכן, אם חישבנו עד n=10, ואחרי זה שלפנו את n=5, ה lastCalculated יהיה שווה ל 5. זה סותר את האינוריאנטה השלישית (עבור n < i כן יהיו ערכים מחושבים) וגם מחייב אותנו לבצע חישובים כפולים. אם ה lastCalculated שווה ל 5, המחלקה "שוכחת" את כל החישובים עד n=10 ולכן תצטרך לחשב אותם מחדש.

שאלה 2:

תשובה: ד' – רק אופציות 1+2 נכונות.

נימוק:

אופציה 1 נכונה בגלל התסריט הבא:

```
Fib f1 = Fib(3); Fib f2 = Fib(4); f1.get(4)
```

ב f1 חושבו רק ערכים עד 3, אבל lastCalculated שווה ל 4, לכן עבור 5 ישלף 0 מהמערך.  
אופציה 2 נכונה אם מחשבים את f1.get(5) באותו הקוד. בשביל לחשב את 5 (ה lastCalculate הוא 4 ולכן החישוב מתבצע) משתמשים ב fib(3) שחושב נכון, וב fib(4) שראינו שהוא 0, ולכן מקבלים תוצאה לא נכונה.

שאלה 3:

תשובה: ד' – רק אופציות 1+2 נכונות.

נימוק:

הקריאה ל this מאתחלת את lastCalculated ל 1. אם נוותר עליה, ערכו יהיה 0, ואז האיטרציה הראשונה של calcFibForI תתחיל ב i=1, ותנסה לגשת לתא ה i-2 שהוא שלילי.

שאלה 4:

תשובה:

נימוק:

שאלה 5:

תשובה: ד. ריצת התוכנית תסתיים ויודפס Done

נימוק:

התוכנית מייצרת בלולאה אובייקטים מסוג Finalize שאף אחד לא מצביע עליהם. לכן, בשלב מסויים ה GC יתחיל לפעול ויקרא ל finalize על כל אחד מהם. ה finalize מבצע ++ על שדה סטטי מסוג int שבסופו של דבר יגיע לגודל המקסימלי של int ואז ערכו יהפוך להיות שלילי. זה מה שיגרום לתנאי ה break בתוך הלולאה להתקיים.

שאלה 6:

תשובה: ח. כל הפונקציות לא מתקמפלות

נימוק:

f1 לא מתקמפלת כי |2 יכולה להיות רשימת Double-ים ו |1 יכולה להיות רשימת Integer-ים. לו הקוד היה מתקמפל, היינו יכולים להכניס Double לרשימה של Integer-ים.  
f2 לא מתקמפלת כי |2 יכולה להיות רשימה של מחרוזות ו |1 יכולה להיות רשימת Integer-ים. אם ההשמה הזו חוקית, אז |1 תצביע ל |2, ואז בעצם |1 תוכל למשל לשלוף איבר מהרשימה, לחשוב שהוא integer ולקבל מחרוזת.  
F3 לא מתקמפלת בגלל אותה הסיבה של f2. עבור השמה כלשהי של S, למשל Integer יכולנו לשלוח ב |2 משהו מטיפוס שונה מ S, נגיד String, ושוב מקבלים השמה שלו היתה חוקית היינו יכולים להגיע לבעיית בטיחות טיפוסים.

שאלה 7:

תשובה: ה. false 10 7 6 4

נימוק:

ה Supplier מייצר את הערכים הבאים: 10 ואחריו 6. ה 10 עובר את הפילטור הראשון, מודפס ב peek, הופך ל 7, מודפס שוב ב peek, ואז מגיע ל allMatch. הוא מקיים את התנאי ולכן ממשיכים לאיבר הבא שהוא 6. הוא מודפס, הופך ל 3, מודפס שוב ואז מגיע ל anyMatch. 3 לא מקיים את התנאי ולכן ה allMatch מחזיר false ועוצר את הזרם.

שאלה 8:

תשובה: א. כל הטענות נכונות.

נימוק:

טענה 1 נכונה כי אם EDGAR אינו final, נוכל באיזשהו מקום בקוד לבצע את ההשמה EDGAR=CLAUD ואז השוויון יתקיים.  
טענה 2 נכונה כיוון ש MyEnum הוא enum עם שלושה מופעים, ולא ניתן לרשת ממנו. לכן, לא אפשרי שיהיה קיים אובייקט MyEnum נוסף על השלושה שמוגדרים במחלקה.  
טענה 3 נכונה כיוון ש name הוא לא private, ולכן קוד מחוץ למחלקה (במחלקה אחרת באותו ה package יכול היה לבצע את ההשמה "Paul" = MyEnum.EDGAR.name).

שאלה 9:

תשובה: ח. כל הפונקציות לא מתקמפלות

נימוק:

בשורת ההדפסה הראשונה ב main הטיפוס הסטטי הוא AA ולכן החתימה הנבחרת היא getComp שמקבלת Object – היחידה שמופיעה ב AA. בזמן ריצה היא נדרסת ע"י הפונק' הראשונה ב BB.  
בשורה ההדפסה השניה ב main הטיפוס הסטטי הוא BB, ולכן החתימה הנבחרת היא getComp שמקבלת מחרוזת (השניה ב BB) – היא מתאימה יותר מזו שמקבלת Object. היא גם זו שרצה בזמן ריצה.

שאלה 10:

תשובה: ז. כל המימושים לא מייצרים את הפלט הרצוי

נימוק:

מימוש 1 – מתקמפל, אבל אין לו דרך לייצר 0 כי החיסור הוא לא בין שני ערכים זהים.  
מימוש 2 – לא יתקמפל. המימוש באמצעות פונקציית הלמבדא מעדכן את השדה i של Decidable, אלא ש i הוא final (כמו כל שדה שמוגדר במנשק).  
מימוש 3 – לא יתקמפל. printDecision צריכה לקבל Decidable. בשביל לממש Decidable עם פונק' למבדא צריך פונק' שמקבלת שני char-ים ומחזירה int. המימוש הנתון פה הוא של פונק' שמחזירה Decidable ולא int.

שאלה 11:

תשובה:

נימוק:

שאלה 12:

תשובה:

נימוק:

שאלה 13:

תשובה:

נימוק:

שאלה 14:

תשובה:

נימוק:

שאלה 15:

תשובה: ח. כל הפונקציות לא מתקמפלות

נימוק:

הפונק'  $func1$  צריכה להחזיר משהו מטיפוס  $K$  שזה הטיפוס של הפרמטר הראשון. היא מחזירה  $\langle Integer, Box \rangle$ .  $K$  הוא טיפוס שיכול להרחיב את  $\langle Integer, Box \rangle$ , לכן אם הפרמטר הראשון הוא משהו ששורש מ  $\langle Integer, Box \rangle$ , יש חוסר תאימות.

הפונק'  $func2$  מפעילה בנאי של טיפוס גנרי  $K$ , מה שאסור (אף אחד לא הבטיח שב  $K$  הספציפי שנקבל יהיה מוגדר בנאי שמקבל פרמטר יחיד).

הפונק'  $func3$  מחזירה תמיד  $\langle Integer, Box \rangle$  בעוד שמה שאמור לחזור זה  $\langle T, Box \rangle$ ,  $T$  הוא לאו דווקא  $Integer$ .

שאלה 16:

תשובה: ה. רק טענה 1+2 נכונות.

נימוק:

בשביל שההשמה בשורה השניה תתקמפל, C חייבת לרשת מ A, כי אחרת היינו נדרשים ל casting מפורש. טענה 1: עבור B שירש מ C שירש מ A הקוד יתקמפל וירוץ ללא שגיאות, כיוון שכל ההמרות הן כלפי מעלה.

טענה 2: עבור C יורש מ A שירש מ B הקוד יתקמפל וירוץ ללא שגיאות כיוון שההמרות הן או כלפי מעלה או כלפי מטה בעץ הירושה A. אם נשלח  $b2 = \text{new A}()$ ,  $b1 = \text{new B}()$ , השורה הראשונה בלבד תעוף בזמן ריצה על Casting. אם נהפוך את  $b1$  ו  $b2$ , השורה הראשונה תרוץ והשניה תיכשל על Casting.

שאלה 17:

תשובה:

נימוק:

שאלה 18:

תשובה:

נימוק:

שאלה 19:

תשובה:

נימוק:

שאלה 20:

תשובה:

נימוק:

שאלה 21:

תשובה:

נימוק: