#### **חלק 1 (50 נקודות):**

**שאלה 1:** מהי התפקיד של האנוטציה @BeforeEach בבדיקות JUnit?

**A. להפעיל קוד לפני כל בדיקה במחלקה.**

B. להפעיל קוד אחרי כל בדיקה במחלקה.

C. להפעיל קוד פעם אחת בלבד לפני הבדיקה הראשונה במחלקה.

D. להפעיל קוד פעם אחת בלבד אחרי הבדיקה האחרונה במחלקה.

**פרט:**

**כאשר רוצים לבצע פעולה מסוימת לפני כל טסט נשתמש ב@beforeHach לדוגמא: אתחול אובייקטים.**

**שאלה 2:** מהו השימוש העיקרי בטסטים פרמטריים ב-JUnit?

**A. לבצע אותו טסט עם ערכי קלט שונים במקום לכתוב מספר טסטים נפרדים.**

B. לבדוק את זמן הריצה של הטסטים ולאופטימזציה.

C. לקבוע את סדר הרצת הטסטים.

D. לקבוע אילו טסטים ירוצו במקביל.

**פרט:**

**כאשר נרצה לבצע טסט מסוים עם כמה אפשרויות של קלט נשתמש בטסטים פרמטריים שאוטומטית יופעל הטסט עם הפרמטרים השונים שנגדיר מראש או בפונקציה או בקובץ או כרשימה מעל הטסט, לדוגמא: ערכים רנדומלים.**

**שאלה 3:** באילו מקרים מומלץ להשתמש באנוטציה @Disabled ב-JUnit 5?

A. כאשר רוצים להריץ טסט באופן אוטומטי בזמן ההרצה.

**B. כאשר הטסט עדיין בפיתוח ולא מוכן להרצה.**

C. כאשר הטסט מבוצע על פונקציונליות שהוסרה או שונתה.

D. כאשר המנהל מבקש לדלג על הטסט.

**פרט:**

כאשר טסט מסוים עדיין לא מוכן, או כאשר רוצים להשהות את הרצת הטסט זמנית, ניתן להשתמש ב- @Disabled זה מאפשר להימנע משגיאות בהרצה אך שומר את הטסט לעתיד.

#### **חלק 2 (50 נקודות):**

בתרגיל הזה תכתבו בדיקות למחלקה Calculator.

מחלקה זו מספקת פעולות בסיסיות: חיבור, חיסור, כפל וחילוק. התרגיל כולל יצירת Extension מותאם אישית ב-JUnit 5, שימוש בבדיקות פרמטריות, ויישום TestWatcher למעקב אחרי תוצאות הבדיקות.

**מחלקת Calculator:**

|  |
| --- |
| **public class Calculator {   public int add(int a, int b) {  return a + b;  }   public int subtract(int a, int b) {  return a - b;  }   public int multiply(int a, int b) {  return a \* b;  }   public int divide(int a, int b) {  if (b == 0) {  throw new ArithmeticException("Cannot divide by zero");  }  return a / b;  } }** |

### **דרישות למחלקת הבדיקות:**

1. **Lifecycle Annotations**:
   * השתמשו ב-@BeforeAll, @BeforeEach, @AfterEach, ו-@AfterAll לניהול פעולות הכנה וניקוי סביבת הבדיקות.
2. **Custom JUnit Extension**:
   * צרו הרחבה מותאמת אישית שתאתחל לוג של פעולות לפני תחילת הבדיקות ותנקה אותו לאחר סיום הבדיקות.
3. **Parameterized Tests**:
   * השתמשו בבדיקות פרמטריות כדי לבדוק את שיטות add ו-divide עם סטים שונים של ערכים. בחרו 2 שיטות שונות לביצוע פרמטריות.
4. **TestWatcher**:
   * יישמו TestWatcher כדי לרשום תוצאות הצלחה או כישלון של כל בדיקה.