

# הצעה לסיור לימודי

## ענת בן יעקב

**פעילות מונחית על ידי המורה בשדה התעופה: דגש על תכנות מונחה עצמים ומבני נתונים**

**מיקום:** נמל התעופה בן-גוריון, תל אביב

**מטרה:** לספק לתלמידים הבנה של תכנות מונחה עצמים (OOP) ומבני נתונים באמצעות יישומים בעולם האמיתי בשדה תעופה. התלמידים יעבדו בקבוצות קטנות (2-3 תלמידים) כדי לדמות ולפתור בעיות בניהול שדה תעופה תוך שימוש בעקרונות OOP ומבני נתונים.

**הכנות המורה:**

### 1. הכנות לפני הפעילות:

- **אישורים ולוגיסטיקה:** מומלץ לוודא שיש אישורים לביקור ולוודא פרטי לוגיסטיקה עם רשויות שדה התעופה.
- **ציוד:** לארגן מחשבים ניידים או טאבלטים עם גישה לאינטרנט ומותקן Java IDE (למשל, Eclipse, IntelliJ IDEA).
- **חומרים:** להביא לוחות לבנים, טושים, ניירות ועטים לשרטוטים וכתיבה.
- **דפי מידע:** להכין דפי מידע עם סיכום של עקרונות OOP, מבני נתונים ותיאור התרחיש.
- **חלוקת קבוצות:** לתכנן איך לחלק את התלמידים לקבוצות קטנות (2-3 תלמידים כל אחת).

### 2. דפי מידע לפעילות:

- **סיכום עקרונות OOP:** סקירה קצרה של מחלקות, אובייקטים, ירושה ופולימורפיזם.
- **סיכום מבני נתונים:** הסבר על מערכים, רשימות, תורים ומפות האש.
- **תיאור התרחיש:** עבור כל אחד מהתרחישים יש תיאור מפורט של הבעיה ותוצאות צפויות.
- **דוגמאות דיאגרמות מחלקות:** דיאגרמות מחלקות לדוגמה שיעזרו לתלמידים לדמיין את התכנון שלהם.

### 3. תרחיש והצגת הבעיה:

- **תרחיש:** התלמידים ידמו ניהול תהליך בשדה התעופה, למשל הצ'ק-אין והעלייה למטוס בטיסה.
- **הצגת הבעיה:** להשתמש בעקרונות OOP כדי לדגם את התהליך וליישם מבני נתונים לניהול נתוני הנוסעים.

## מבנה הפעילות:

- **הקדמה לפעילות (10 דקות):**
  - **ברוכים הבאים ומבוא:** הסבר קצר על פעילות היום והרלוונטיות שלה ל OOP-ומבני נתונים.
  - **מטרות:** להציג את מה שהתלמידים ישיגו בסוף המפגש) הבנת עקרונות OOP, יישום מעשי באמצעות מבני נתונים, עבודת צוות ופתרון בעיות.
- **חלוקת קבוצות והצגת הבעיה (10 דקות):**
  - **יצירת קבוצות:** חלוקת התלמידים לקבוצות קטנות של 2-3 תלמידים.
  - **הצגת הבעיה:** הצגת הבעיה הקשורה לניהול שדה התעופה:
    - **תרחיש:** ניהול תהליך הצ'ק-אין והעלייה למטוס.
    - **משימה:** להשתמש בעקרונות OOP כדי לדגם את התהליך וליישם מבני נתונים לניהול נתוני הנוסעים.
- **הסבר על העקרונות (10 דקות):**
  - **יסודות OOP:** סקירה קצרה של עקרונות OOP כגון מחלקות, אובייקטים, ירושה ופולימורפיזם.
  - **מבני נתונים:** סקירה קצרה של מבני נתונים רלוונטיים כמו מערכים, רשימות, תורים ומפות האש.
- **פעילות קבוצתית: שלב התכנון (20 דקות):**
  - **משימה:** כל קבוצה תתכנן מבנה מחלקות לדגימה של תהליך הצ'ק-אין והעלייה למטוס.
  - **מחלקות:** מחלקות אפשריות כוללות **נוסע**, **טיסה**, **דלפק צ'ק-אין**, **שער עלייה למטוס**.
  - **תכונות ושיטות:** הגדרת תכונות (כגון שם הנוסע, מספר הטיסה) ושיטות (כגון צ'ק-אין, הקצאת מושב).
  - **שרטוט:** שימוש בניירות או לוחות לבנים לשרטוט דיאגרמות מחלקות המציגות את הקשרים בין המחלקות.
- **פעילות קבוצתית: שלב היישום (20 דקות):**
  - **קידוד:** כל קבוצה תכתוב תוכנית פשוטה ליישום התכנון שלהם באמצעות Java.
  - **מבני נתונים:** יישום מבני נתונים לאחסון וניהול נתוני הנוסעים (כגון תור לדלפקי הצ'ק-אין, רשימה לשערי העלייה למטוס).
- **בדיקה והצגה (10 דקות):**

- **בדיקה**: הקבוצות יבדקו את התוכניות שלהן על ידי הרצת סימולציות ובדיקת הלוגיקה.
- **הצגה**: כל קבוצה תציג את הפתרון שלה, תסביר את התכנון המחלקתי ואת יישום מבני הנתונים.
- **סיכום ודיון (10 דקות)**:
  - **רפלקציה**: דיון על מה שהתלמידים למדו על OOP ומבני נתונים.
  - **יישומים בעולם האמיתי**: איך עקרונות אלו משמשים במערכות ניהול שדה תעופה בעולם האמיתי.
  - **שאלות ותשובות**: פתיחת הזדמנות לשאלות ולדיון נוסף.
- **חומרים נחוצים**:
  - **טכנולוגיה**: מחשבים ניידים או טאבלטים עם Java IDE (מותקן) למשל, Eclipse, IntelliJ IDEA).
  - **כלי שרטוט**: ניירות, עטיים, לוחות לבנים וטושים.
  - **דפי מידע**: סיכומים של עקרונות OOP, מבני נתונים, תיאור התרחיש ודיאגרמות מחלקות לדוגמה.
  - **קוד מוכן מראש**: קטעי קוד לדוגמה להתייחסות מהירה.

#### דוגמת קוד:

```
class Passenger {
    private String name;
    private String passportNumber;

    public Passenger(String name, String passportNumber) {
        this.name = name;
        this.passportNumber = passportNumber;
    }

    public String getName() {
        return name;
    }
}
```

```

public String getPassportNumber() {
    return passportNumber;
}
}

class Flight {
    private String flightNumber;
    private List<Passenger> passengerList;

    public Flight(String flightNumber) {
        this.flightNumber = flightNumber;
        this.passengerList = new ArrayList<>();
    }

    public void addPassenger(Passenger passenger) {
        passengerList.add(passenger);
    }

    public List<Passenger> getPassengerList() {
        return passengerList;
    }
}

class CheckInCounter {
    private Queue<Passenger> queue;

    public CheckInCounter() {
        this.queue = new LinkedList<>();
    }
}

```

```

    }

    public void checkIn(Passenger passenger) {
        queue.add(passenger);
    }

    public Passenger processCheckIn() {
        return queue.poll();
    }
}

class BoardingGate {
    private List<Passenger> boardedPassengers;

    public BoardingGate() {
        this.boardedPassengers = new ArrayList<>();
    }

    public void board(Passenger passenger) {
        boardedPassengers.add(passenger);
    }

    public List<Passenger> getBoardedPassengers() {
        return boardedPassengers;
    }
}

public class AirportSimulation {
    public static void main(String[] args) {

```

```
Passenger passenger1 = new Passenger("John Doe", "P12345678");
Flight flight = new Flight("BG123");
CheckInCounter checkInCounter = new CheckInCounter();
BoardingGate boardingGate = new BoardingGate();

checkInCounter.checkIn(passenger1);
Passenger checkedInPassenger = checkInCounter.processCheckIn();
flight.addPassenger(checkedInPassenger);
boardingGate.board(checkedInPassenger);
System.out.println("Flight " + flight.getFlightNumber() + " has boarded passengers:");
for (Passenger p : boardingGate.getBoardedPassengers()) {
    System.out.println("Name: " + p.getName() + ", Passport Number: " + p.getPassportNumber());
}
}
}
```