בית הספר למדעי המחשב

שם הקורס: מבוא לתכנות מונחה עצמים

קוד הקורס: 2-7029910-2, 2-7029910-3, 2-7029910-4

תאריך בחינה: יום ה 28/03/2024 יח' אדר ב' תשפ"ד

סמסטר: א' מועד: הערכה חלופית

משך הבחינה: 11.4.2024, 23:59

שם המרצה: ד"ר אור חיים אנידג'ר, ד"ר ליאת כהן

פירוט הניקוד לכל שאלה:

שאלה	ניקוד מקס'	ניקוד בפועל
1.a	20	
1.b.1	10	
1.b.2	10	
2	30	
3.1	10	
3.2	10	
3.3	10	
סה"כ	100	

הוראות כלליות:

1. **המטלה הינה אישית, נועדה לעבודה עצמאית בלבד ללא שיתוף פעולה עם אף אחד, בפרט יתר הנבחנים. העתקות יענו בחומרה.**
2. **המטלה וכלל הוראותיה, שאלותיה, ותשובותיה – הינן בשפת JAVA בלבד, אלא אם צוין אחרת.**
3. **למטלה אין מועד ב'.**
4. **ע"פ התקנון – לא ניתן לערער על המטלה!**
5. **הניסוח הוא בלשון רבים מטעמי נוחות ומתייחס לכולם!**
6. **במטלה ניתן לצבור עד 100 נקודות, ואיננה מכילה שאלות בחירה; יש לענות על כל השאלות ללא יוצא מן הכלל.**
7. **יש להקפיד לענות תשובות מלאות, ומפורטות ככל הניתן. יחד עם זאת, תשובות לא מדויקות, או לא מלאות, או מלאות שלא לצורך - יגררו ירידת נקודות.**
8. **המטלה מחולקת לשני חלקים:**
   a. **שאלת תכנות וחשיבה תכנותית המבוססת על ההרצאות ומטלות (70 נקודות).**
   b. **שאלה תכנותית המבוססת על התרגולים (30 נקודות).**
9. **יש לציין בפתרון את מספר השאלה אשר עבורה ניתנת התשובה.**
10. **יש להגיש את הפתרון בקובץ PDF, ללא תמונות או צילומי כתב יד, וללא עמודי טיוטה – על הפתרון להיות דיגיטלי לחלוטין!**
11. **ניתן להשתמש בחומר עזר: הרצאות ותרגולים בלבד.**
12. **חובה להגיש קוד מלא והסברים מודפסים (לא סרוקים בכתב יד).**

בהצלחה! 😊

חלק א' - שאלת תכנות וחשיבה תכנותית המבוססת על ההרצאות (70 נק')

שאלה 1 – עיצוב בהנחיה (40 נק'):

מבוא:

מטרת התרגיל היא לפתח מערכת ניהול טיסות מתקדמת שתאפשר יצירה, ניהול ומעקב אחר טיסות ונוסעים. המערכת תשתמש בתבניות עיצוב לצורך פתרון אתגרים שונים ולהגדלת גמישות ואפשרות להרחבה בעתיד. בפתרונכם, יש להדגיש את החתימות של המטודות השונות.

a. מימוש המערכת:

עליכם לפתח מערכת כולל מימוש **קוד מלא**, תוך שימוש במחלקות אבסטרקטיות וממשקים, אשר יתמוך בשלושת הפונקציות הבאות ע"י שימוש בתבניות עיצוב:

Strategy Pattern: פיתוח מנגנון חיפוש טיסות שיכלול אלגוריתמים שונים לחיפוש טיסה (למשל, לפי מחיר, לפי זמן המראה, לפי משך הטיסה). המנגנון יאפשר למשתמש לבחור את אסטרטגיית החיפוש הרצויה בזמן ריצה.

Composite Pattern: בניית מודל היררכי של חברות תעופה, שיכלול תתי-חברות וטיסות תחת כל חברה. המודל יאפשר ניהול ודיווח מרוכז של נתונים עבור חברות תעופה גדולות המחזיקות במספר תתי-חברות.

Observer Pattern: הטמעת מערכת התראות לעדכונים על טיסות, כמו שינויים בזמני המראה, ביטולים, או מבצעים חדשים. נוסעים ועובדי חברת התעופה יכולים להירשם כצופים (observers) ולקבל עדכונים בזמן אמת.

b. שאלות ידע ובקיאות:

1. תבניות עיצוב:

1. הסבירו את הבחירה בתבניות עיצוב: פרטו והסבירו מדוע כל אחת מתבניות העיצוב משמשות לפתרון הבעיות הנתונות בפרויקט. יש להשתמש בדוגמאות מהקוד לכל תבנית עיצוב.
2. יתרונות ומטרות: תאר את היתרונות שתבניות העיצוב הללו מספקות למערכת ניהול הטיסות. איך כל תבנית תורמת לגמישות, לניתנות להרחבה, ולתחזוקה של הקוד?

2. עקרונות תכנות מונחה עצמים:

1. הפרדת ממשק ומימוש: הסבר כיצד השימוש בתבניות עיצוב מדגים את עקרון הפרדת הממשק מהמימוש. דוגמה לכך היא השימוש ב-Strategy Pattern להגדרת ממשקי אלגוריתם חיפוש שונים, ומימושם בצורה גמישה.
2. אינקפסולציה: פרט כיצד תבניות העיצוב משפרות את האינקפסולציה במערכת. (לדוגמה, איך ה-Composite Pattern מאפשרת ניהול היררכי של חברות תעופה וטיסות מבלי לחשוף את פרטי המימוש הפנימיים למשתמש.)
3. פולימורפיזם: הסבר את השימוש בפולימורפיזם במערכת ואיך הדבר קשור לתבניות העיצוב בהן השתמשתם. בנוסף, תארו איך המערכת מצליחה לטפל בסוגים שונים של אובייקטים (למשל, סוגים שונים של מנגנוני חיפוש או סוגים שונים של מנויים) באופן גמיש?

שאלה 2 – עיצוב חופשי (30 נק'):

עליכם לפתח מערכת הכוללת מימוש **קוד מלא**, תוך שימוש בירושה, מחלקות אבסטרקטיות, ממשקים, עקרונות OOP ותבניות עיצוב. בתשובתכם יש להגיש קוד מלא ומסמך המפרט באילו כלים השתמשתם וכיצד כל כלי או מנגנון שכזה תורם לקוד שלכם. רמז: ישנן 5 תבניות עיצוב המסתתרות בתיאור הבעיה.

תכננו מערכת הרשמה לקורסים שבכל רגע נתון מכילה לכל היותר 100 פעילים (מרצים, מתרגלים או סטודנטים).
במערכת זו מרצים ומתרגלים יכולים להגדיר קורסים מסוגים שונים (סמינר, בחירה ,חובה) הכוללים את שם הקורס ומספר הקורס. נדגיש כי אם קורס הוגדר פעם אחת הוא לא יוגדר מחדש והמערכת תפנה לקורס שכבר הוגדר.
סטודנטים יכולים לבחור קורס ולהוסיפו לסל הקניות שלהם בלבד ויש בקורס מקום. במידה ואין, ניתן לבקש התראה כאשר מתפנה מקום. מערכת זאת נועדה לשימוש משתמשי קצה ולכן נחשוף עבורם ממשק פשוט ככל הניתן.

חלק ב' – שאלה תכנותית המבוססת על התרגולים (30 נק')

שאלה 3 (30 נק'):

עבור חתיכות הקוד הבאות, ציינו האם מתקיימת שגיאת קומפילציה, שגיאה בזמן ריצה (Run Time Error), או שגיאה לוגית. הסבירו את סיבת השגיאה עבור כל סעיף. אם מדובר בשגיאה לוגית יש לכתוב את הפלט וכיצד ניתן לתקן את השגיאה, בקצרה. כל סעיף משקלו 10 נק'.

1) בשאלה זאת, נרצה שתי רשימות, האחת באותיות גדולות והשנייה באותיות קטנות.
נניח וקיימת מחלקה Person, עם המתודות שבקוד:

```
public static void main(String[] args) {
    List<Person> uppercaseLettersList = new ArrayList<>();
    List<Person> lowercaseLettersList = new ArrayList<>();
    lowercaseLettersList.add(new Person("alice"));
    lowercaseLettersList.add(new Person("bob"));
    for (int i = 0; i < lowercaseLettersList.size(); i++) {
        uppercaseLettersList.add(lowercaseLettersList.get(i));

uppercaseLettersList.get(i).setName(uppercaseLettersList.get(i).getName().to
UpperCase());
    }
    // Printing the first list
    System.out.println("Upper case List:");
    for (Person person : uppercaseLettersList) {
        System.out.println(person.getName());
        }
    // Printing the second list
    System.out.println("Lower case List:");
    for (Person person : lowercaseLettersList) {
        System.out.println(person.getName());
    }
}
```

2) נניח וקיימת מחלקה Position, עם המתודות שבקוד:

```
public static void main(String[] args) {
    int n = 10;
    Position[][] positions = new Position[n][];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            positions[i][j] = new Position(i, j);
        }
    }
    for (Position[] position_row: positions) {
        for (Position position: position_row) {
            System.out.print(position);
        }
        System.out.println();
    }
}
```

3) באמצעות שינוי אחד בקוד, הסבירו כיצד ניתן לקבל התנהגות פולימורפית

```
class Animal {

   final static String sound = "Animal sound";

   void makeSound() {

      System.out.println(sound);

   }

}

class Dog extends Animal {

   final static String sound = "Bark";

   void makeSound() {

      System.out.println(sound);

   }

}

public class Main {

   public static void main(String[] args) {

      Animal animal = new Dog();

      animal.makeSound();

   }

}
```