תרגול חזרה לבחינה - מונחה עצמים 2025

תכנון מערכת

שאלה 1

אתם נדרשים לממש מערכת לניהול הזמנות במסעדה, שבה ניתן לבצע הזמנות באמצעות טלפון, אתר אינטרנט, או הזמנה במקום. הפונקציונליות הנדרשת כוללת הוספת פריטי תפריט להזמנה, חישוב סך המחיר של ההזמנה, ותשלום באמצעי תשלום לבחירת הלקוח (מזומן, כרטיס אשראי, או ארנק דיגיטלי). בנוסף, יש לאפשר ניהול דוחות על ההכנסות היומיות של המסעדה.

בתשובתכם אתם נדרשים:

1. לממש את המערכת ברמת החתימות (מחלקות, שדות, פונקציות).
2. להציג את השימוש בתבניות עיצוב במידת הצורך, וכן להסביר כיצד החתימות בקוד מצביעות על כך שמדובר בתבניות בהן בחרתם.
3. במידה והחתימה מצביעה על שימוש בתבנית עיצוב יש לממש כיצד תבנית זו באה לידי ביטוי.

שאלה 2

אתם נדרשים לממש מערכת לניהול אירועי ספורט, שבה ניתן לנהל אירועים, משתתפים, ותוצאות. הפונקציונליות הנדרשת כוללת יצירת אירועים חדשים (כגון מרתון, טורניר כדורסל, תחרות שחייה), רישום משתתפים לאירועים, ניהול תוצאות האירועים, וחישוב הזוכים בהתאם לכללי האירוע. בנוסף, יש לאפשר הפקת דוח בסיום כל אירוע הכולל את שמות הזוכים, מספר המשתתפים, והכנסות האירוע. מערכת זו יכולה לנהל מספר אירועים במקביל, ולכלול חישוב של דוח מסכם לכלל האירועים שבוצעו במערכת.

בתשובתכם אתם נדרשים:

4. לממש את המערכת ברמת החתימות (מחלקות, שדות, פונקציות).
5. להציג את השימוש בתבניות עיצוב במידת הצורך, וכן להסביר כיצד החתימות בקוד מצביעות על כך שמדובר בתבניות בהן בחרתם.
6. במידה והחתימה מצביעה על שימוש בתבנית עיצוב יש לממש כיצד תבנית זו באה לידי ביטוי.

שאלה 3:

בתשובה לשאלה יש לזהות את התבניות עיצוב הרלוונטיות ולענות על הסעיפים למטה:
עליכם לתכנן מערכת לניהול פלטפורמה לסטרימינג של תכני וידאו, שבה יש צורך לטפל ביכולות הבאות:

1. **מודל מנוי גמיש**:
   - כל משתמש מתחיל עם מנוי בסיסי (Basic Plan).
   - ניתן להוסיף למנוי תכונות נוספות לפי בחירה, כגון תמיכה ברזולוציית 4K, אפשרות להורדת תכנים לצפייה אופליין, או הוספת מכשירים במקביל.
   - התוספות יכולות להשפיע על מחיר המנוי, על איכות הצפייה, ועל משך החוזה.
2. **התראות ועדכונים אוטומטיים**:
   - מנויים צריכים לקבל הודעות כשעולה תוכן חדש, כשהמנוי מתקרב לסיום, או בעת שינוי במדיניות המנויים.
   - המערכת צריכה להיות מסוגלת להתריע באופן דינמי (מייל, הודעת SMS, או התראה בתוך האפליקציה) בהתאם להעדפות המשתמש.
3. **תהליך חידוש מנוי**:
   - אחת לכמה זמן (למשל, בסוף חודש) המערכת מפעילה תהליך חידוש.
   - תהליך זה כולל שלבים עוקבים כגון בדיקת סטטוס החשבון, מתן הטבות (Promotions) במידת הצורך, חיוב כספי, עדכון פרטי המנוי במערכת ושליחת אישור חידוש ללקוח.
   - רצוי לאפשר וריאציות שונות בתהליך עבור סוגי מנויים שונים (מנוי בסיסי, מנוי פרימיום, מנוי בארגון עסקי) או בעת קמפיינים מיוחדים.

דרישות למימוש:

1. ליצור תרשים (למשל UML) למערכת, באופן שברור מתוכו איזה תבנית עיצוב השתמשתם והיכן זה נכנס. יש לתכנן את המערכת באופן שמאפשר:
   a. **הוספה קלה של מאפיינים ותכונות נוספות** למנוי.
   b. **עדכון גורמים רלוונטיים באופן אוטומטי** כאשר מתרחשים שינויים (כגון הוספת תוכן חדש או שינוי במחיר).
   c. **תהליך חידוש מנוי כללי** הניתן להתאמה עבור מנויים שונים או מבצעים שונים.
2. להציג את השימוש בתבניות עיצוב במידת הצורך, וכן להסביר כיצד ניתן לראות שמדובר בתבניות בהן בחרתם.

שאלות חזרה כלליות על החומר

שאלה 1

```
public class Main {
    public static void main(String[] args) {
        Parent p = new Child();
        System.out.println(p.getMessage());
    }
}
class Parent {
    String getMessage() {
        return "Hello from Parent!";
    }
}
class Child extends Parent {
    String getMessage() {
        return "Hello from Child!";
    }
}
```

שאלה 2

השלם את הקוד למימוש תבנית עיצוב סינגלטון

```
public class Singleton {
```

```
    – Singleton instance;

    – Singleton() {
        ________
    }

    public static Singleton getInstance() {
        ________
    }
}
```

שאלה 3

צור מפעל ליצירת צורות

```
abstract class Shape {
    abstract void draw();
}

class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a Circle");
    }
}

class Rectangle extends Shape {
    @Override
    void draw() {
```

```
        System.out.println("Drawing a Rectangle");
    }
}


class ShapeFactory {
    // Complete the method to create and return a Shape based on the type
    public static Shape getShape(String type) {
        // Add your implementation here
    }
}
```

שאלה 4

Which of the following accurately describes the effects of using the `final` keyword in Java?

A. A `final` method cannot be overridden by subclasses.
B. A `final` class cannot be subclassed (inherited).
C. A `final` variable cannot be reassigned once initialized.
D. All of the above.

שאלה 5

האם התכנית הזו תרוץ? מה יהיה הפלט?

```
public class Main {
    public static void main(String[] args) {
        try {
            int result = divide(10, 0);
            System.out.println("Result: " + result);
        } catch (NullPointerException e) {
```

```
            System.out.println("ArithmeticException caught: " + e.getMessage());
        }
    }
    public static int divide(int a, int b) {
        return a / b;
    }
}
```

שאלה 6

(יכולות להיות כמה תשובות נכונות)

What is polymorphism in Java?
א. The ability of a class to inherit from multiple classes
ב. The ability to define multiple methods with the same name but different parameters
ג. The ability of an object to take on many forms
ד. The ability of Java to handle different data types in the same variable

שאלה 7

מה יהיה הפלט הבא?

```
public class Main {
public static void main(String[] args) {
 String str1 = "Hello";
String str2 = new String("Hello");
 System.out.println(str1 == str2);
 System.out.println(str1.equals(str2));
}
 }
```

שאלה 8

What is the difference between overloading and overriding in Java?
א. Overloading refers to defining multiple methods with the same name but different parameters, while overriding refers to redefining a method in a subclass
ב. Overloading occurs only in inheritance, while overriding can occur in any class
ג. Overloading and overriding are synonyms in Java
ד. Overloading requires `@Override`, while overriding does not

שאלה 9

What does the `static` keyword mean in Java?
א. It allows a method or variable to belong to the class rather than an instance of the class
ב. It ensures a method cannot be overridden
ג. It is used to make a method or variable immutable
ד. It ensures that only one instance of a class can be created

שאלה 10

הסבר את ההבדל בין comperator לבין comperable.

מתי נשתמש במה?

שאלה 11

איזה הבדל ברמת הקוד יש בין שגיאות מסוג Exception לשגיאות מסוג runtimeException?

שאלה 12

במידה וארצה להגדיר מחלקה גנרית של רשימה דינמית, אך כל אובייקט ברשימה זו חייב להיות סוג של shape, כיצד ניתן להגדיר זאת?

שאלה 13

מה ההבדל במימוש של סינגלטון בין java לפייתון וממה הוא נובע?

שאלה 14

ציין 2 יתרונות עיקריים של java על פני פייתון, ו2 יתרונות של פייתון על java

שאלה 15

כיצד אוכל להדפיס אובייקט מורכב בjava? כיצד בפייתון?

מה הסיבה שלולא אגדיר זאת בעצמי תודפס הכתובת של האובייקט בזיכרון?

<u>שאלה 16</u>

נתונה מחלקת Movie:

```
public class Movie{
   private final double rating;
   private final String name;
   private final int year;
   // Constructor
   public Movie(String name, double rating, int year) {
      this.name = name;
      this.rating = rating;
      this.year = year;
   }

   // Getter methods for accessing private data
   public double getRating() { return rating; }
   public String getName()   {  return name; }
   public int getYear()      {  return year;  }
}
```

עליכם לספק אפשרות למיין אובייקטים של המחלקה לפי **rating**, לפי **name** ולפי **year**.

הדרכה: אין צורך במימוש פונקציית מיון, ניתן לשמור את אוסף של סרטים ב- ArrayList להשתמש במתודה sort של מחלקת ArrayList:

```
list.sort(Compreator comp)
```

דוגמה לבדיקה:

```
public static void main(String[] args) {
   ArrayList<Movie> list = new ArrayList<Movie>();
   list.add(new Movie("Force Awakens", 8.3, 2015));
   list.add(new Movie("Star Wars", 8.7, 1977));
   list.add(new Movie("Empire Strikes Back", 8.7, 1980));
   list.add(new Movie("Return of the Jedi", 8.4, 1983));
   ......
```

עליכם גם יש להשלים את קוד הבדיקה

```
   ......
}
```

פתרון - דרך 1

```
public class RatingComperator implements Comperator<Movie>{

public int compare(Movie m1, mMovie m2){

retuen Integer.compare(m1.getRating, m2.getRating)

}

public class NameComperator implements Comperator<Movie>{
```

```
public int compare(Movie m1, mMovie m2){

retuen String.compare(m1.getName, m2.getName)

}
```

פתרון - דרך 2

```
Comperator rating = (m1, m2) -> Integer.compare(m1.getRating,
m2.getRating)

Comperator name= (m1, m2) -> String.compare(m1.getName, m2.getName)

Comperator yaer= (m1, m2) -> Integer.compare(m1.getYear, m2.getYear)
```

עבור כל אחד מהסעיפים ציין האם זה יש שגיאת קומפילציה, שגיאה בזמן ריצה (Run Time Error), או שגיאה לוגית. הסבירו את סיבת השגיאה עבור כל סעיף. אם יש שגיאה לוגית יש לכתוב את הפלט.

יש להניח שכל המחלקות מהמטלה הראשונה הן כפי שכתבתם אותן.

A.

המחלקה ConcretePiece הינה מחלקה מופשטת (Abstarct Class).

```
1. public final abstract class ConcretePiece implements Piece {
2. …
3. }
```

B.

בדומה לסעיף הקודם, המחלקה ConcretePiece הינה מחלקה מופשטת (Abstarct Class):

```
1. ConcretePiece concretePiece = new ConcretePiece();
```

c.

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().toUpperCase());
  }

// 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());
    }
  }
}
```

D.

```
    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();
    }
  }
```

E.

עבור כל אחד מהשדות (Data Members) במחלקה GameBoard המייצגת לוח משחק כלשהוא,
יש לציין איך נכון מבחינת תכנות מונחה עצמים שעל השדות להיות:

private / public, final, static

על המתכנת לאפשר הרצת כמה משחקים במקביל,
ולא לאפשר לשחקן להתחלף במהלך המשחק עם שחקן אחר.

יש לנמק בקצרה את סיבת הבחירה עבור כל בחירה. (נימוק טוב יכול לזכות בנקודות).

```
public class GameBoard implements PlayableBoard {

    // A macro constant
    int BOARD_SIZE = 11;

    // Represents the pieces on the board at any time, null for an empty square
    Piece[][] piecesOnBoard;

    // The two players currently play the game
    Player attacker;
    Player defender;

    // Boolean flags, indicate the game end and who's currently playing
    boolean gameFinished;
```

```
    boolean firstPlayerTurn;

    // Stack to store move history (Move is a class for a certain move)
    Stack<Move> moveHistory;

    …
}
```