מבחן Design Patterns

רון שמוטר | ת"ז 313163222 | מ"א 8071274

**יש אישור מהמרצה להגיש הסבר לגבי ה-Patterns בכל שאלה ללא מימוש**

חלק א':

1. מדובר כאן ב-State. Design Pattern זה מאפשר לאובייקט לשנות את ההתנהגות שלו כאשר המצב הפנימי שלו משתנה.

במקרה שמופיע בתרגיל יש לנו אובייקט "משחק" אשר יכול להכיל 4 מצבים: תקין (המצב ההתחלתי), מצב סכנה, מצב ניצחון (מצב מסיים ראשון), מצב הפסד (מצב מסיים שני). בנוסף, האובייקט "משחק" מחזיק את שני משתנים אשר משמשים לקביעת מצב משחק מתאים: כמות נקודות והשלב במשחק. בכל התקדמות במשחק יורדת כמות מסוימת של נקודות, וניתן להתקדם במשחק.

אפשרויות המעבר בין מצבים הן:

* 1. אם בעת התקדמות כמות הנקודות הגיעה ל-0 המשחק מסתיים בהפסד, ואז ניתן להתחיל את המשחק מחדש.
  2. אם בעת התקדמות הגענו לשלב מספר 3, אז המשחק מסתיים בניצחון, ואז ניתן להתחיל את המשחק מחדש.
  3. אם בעת התקדמות כמות הנקודות מתקרבת ל-0 (יורדת מתחת ל-10 למשל) אז המצב עובר למצב סכנה, ממנו עדיין אפשר להמשיך להתקדם במשחק.

1. מדובר כאן ב-Composite. Design Pattern זה מאפשר לשמור אובייקטים במבנה היררכי של עץ. כל אובייקט הוא מופיע בפני עצמו וניתן לשינוי, וגם ניתן לטפל באובייקט אשר מכיל בתוכו אובייקטים נוספים.

במקרה שמופיע בתרגיל העלים בעץ הם למעשה ערכים מספריים. פונקציה SUM תרוץ על כל הענף ותחשב את הסכום של הערכים שתחת הענף, במקרה של עלה יחזור הערך של העלה עצמו. הפונקציה הרקורסיבית לבדיקה האם כל הערכים זוגיים תרוץ על כל העץ מתוך התוכנית הראשית ותזמן את עצמה בכל פעם על כל אחד מהענפים בעץ. במקרה שחוזר מספר אי זוגי הפונקציה תחזיר False. אחרת תחזיר True.

1. מדובר כאן ב-Proxy. Proxy מאפשר לשלוט בגישה לאובייקט אחר, הוא מתאים כאשר אנחנו רוצים ליחצן חלק מהפעולות הקיימות עבור אובייקט אחד ופעולות אחרות עבור אובייקט אחר.

במקרה שבתרגיל אנחנו רוצים ליחצן רק את הפעולות "הצג מצב דלק" ו-"הצג מיקום" מתוך סט הפעולות הקיים באובייקט.

1. מדובר כאן ב-Strategy – ב-Pattern זה ניתן להגדיר משפחה של אלגוריתמים ולהשתמש בהם ככמוסות סגורות. בעת הצורך ניתן החליף את האלגוריתמים. במקרה זה הפעולה המדוברת היא פעולת המיון.
2. Chain Of Responsibilities – סט של אובייקטים המטפלים בבקשה אחת, כאשר כל אובייקט מבצע את החלק שלו ופועל לאחר שהאובייקט הקודם סיים את משימתו. בשאלה זו יש לוג בסיסי ביותר שעושה רק הדפסה, לוג אחד אחריו שמוסיף גם כתיבה לקובץ ולוג נוסף אחריו שמוסיף גם שליחת SMS. כאשר קורה אירוע של הלוג הכי חמור יבוצעו כל החלקים בשרשרת התלויות בתוכנית. הגדרת התלויות בפועל היא בתוכנית הראשית.
3. ?
4. *לא למדנו – אין צורך לממש*
5. ה-Design Pattern בתרגיל זה הוא Template Method. Pattern זה מאפשר ליצור שלד של אלגוריתם פעולה, ומפנה חלק מהצעדים לתתי מחלקות, כך תת-מחלקות יכולות להגדיר מחדש צעדים מסוימים של האלגוריתם, ללא שינוי המבנה הכללי.

בתרגיל זה יש מבנה כללי של אלגוריתם:

1. החלפת בגדי אימון
2. שתית מים
3. ביצוע תרגיל
4. מקלחת

במימוש יש לייצר מספר מחלקות "ביצוע תרגיל" אשר יורשות ממחלקת בסיס, כל פעם ניתן לייצר אלגוריתם אימון, אשר התרגיל בו הוא משתנה לפי סוג האימון.

1. *לא למדנו – אין צורך לממש*
2. ה-Design Pattern בתרגיל זה הוא Decorator. Pattern זה מאפשר הוספת אחריות נוספת לאובייקט קיים באופן דינאמי ומאפשר גמישות בפיתוח. בתרגיל המופיע בשאלה מייצרים חלון ואז מוסיפים לו יכולות כמו "תאורה תלת-ממדית", "מסגרת צבעונית", "שיקוף חלקי", "הצללה" ו-"רקע מהבהב".
3. בתרגיל זה ה-Design Pattern הוא Memento. Pattern זה מאפשר שמירת מצב של אובייקט וחזרה אליו בעת הצורך. מזכיר מאוד פעולות של UNDO או REDO. בתרגיל זה אנחנו שומרים את המצב של המטריצה ואז חוזרים אליו בעת הצורך.
4. בתרגיל זה ה-Design Pattern הוא Builder. Pattern זה מאפשר הגדרת אובייקט המורכב ממספר תכונות והרכבת אובייקטים מורכבים על בסיס אובייקטים פשוטים. לאחר מכן בתוכנית הראשית ניתן לייצר ישר אובייקטים מורכבים לפי סט אובייקטים שהוגדר מראש. ניתן גם להמשיך לייצר אובייקטים יותר מורכבים בהמשך חיי התוכנה, ולהשתמש בהם.

במקרה המתואר בתרגיל יש אובייקט מחשב בסיסי אשר מכיל "מארז", "לוח אם", "מעבד", "כרטיס גרפי", "זיכרון", ופעולה שעושה בדיקה. לאחר מכן נייצר מחלקות מחשבים מהסוגים הבאים: "מחשב משחקים", "מחשב משרדי" ו-"מחשב סלון".