**Design pattern- מבחן סיום**

**9,7 לא למדנו**

**שלום רועי,**

**באישור איתי נדרשתי רק לזהות את הpattern ולא לממש, כיוון שאיני עוסקת בתכנות בתפקידי. השתדלתי לפרט כמה שיותר!**

**בכל שאלה הpattern מודגש כך.**

1. מדובר במימוש של pattern state, כאשר הcontext יהיה משחק אשר יחזיק אובייקט מסוג מצב. יהיו לנו 4 מצבים-

חללית בסכנה, חללית תקינה, הפסד וניצחון. חללית במצב תקין ובמצב סכנה תחזיק מספר המציין את כמות הפגיעות שנותרו (מתחיל מ50 ויורד עם כל פגיעה), ומספר נוסף המציין את כמות הנקודות שהיא צברה.

נניח שמעבר שלב מצריך צבירה של 10 נקודות באותו השלב(וכמובן לא למות) ונניח שמצב סכנה מוגדר להיות פחות מ10 פגיעות שנותרו לחללית. בנוסף יהיה מספר שמציין מהו השלב הנוכחי בו החללית נמצאת.

בכל מצב יהיו הפעולות הבאות: פגיעה, צבירת נקודה, תחילת משחק מחדש.

חשוב לציין שבמעבר שלבים אנחנו נשארים באותו המצב ומה שמשתנה הוא מספר השלב, וכמות הנקודות הצבורה שמתאפסת. מספר הפגיעות נשמר במעבר בין שלב.

1. מדובר במימוש של Composite, כאשר כל צומת בעץ אשר יש לה בנים היא component שמחזיקה את רשימת הבנים שלה, וכל עלה הוא אובייקט יורש מcomponent אך רשימת הילדים שלו היא null, כי הוא לא יכול להכיל בנים.

בפונקציית הסכום כל קומפוננטה תרוץ על כל ילדיה, תפעיל את פונקציית הסכום עבורם ותסכום את ערכי ההחזר, ותחזיר את ערך זה.

בפונקציית הזוגיות כל קומפוננטה תרוץ על כל ילדיה כל עוד כול כולם זוגיים ותחזיר ערך בוליאני.

1. מדובר במימוש של הדפוסים Proxy,façade יחד. נחלק למספר ממשקים ע"פ תחומי אחריות-

* interface ששולט במהירות- יממש את פונקציות האטה והאצה
* Interface ששולט בפניות- יממש פונקציית פניה שמאלה וימינה
* Interface בקרה- יממש הצגה של מצב דלק והצגת מיקום.

ניצור facade שיממש את כל הממשקים האלו ויחזיק אובייקט מחלקת לוגיקה שמבצעת את כל הפעולות הללו בפועל. במימוש הפונקציות האלו הוא יקרא לפונקציה המתאימה מתוך מחלקת הלוגיקה (בדומה לDAO שכתב לDB בפועל).

מעל הfacade ניצור מחלקת proxy שמחזיקה אובייקט private מסוג facade, הproxy ייחצן רק את הפעולות מיקום,דלק ובפועל יפעיל את הפעולות הרלוונטיות מתוך הfacade.

1. מדובר על מימוש של מספר patterns. ראשית, נבנה מבנה נתונים כנדרש, והוא יממש ממשק visitable (ניתן לבצע בו שינויים). בנוסף נממש ממשק visitor (השינויים עצמם), וניצור 2 visitors אשר כל אחד מהם מקבל מערך ומבצע בו פעולה אחרת- האחד ימיין את המערך במיון מהיר, השני יכפיל את אברי המערך ב2.

נחזור למבנה נתונים. הוא יחזיק פונקציה userCommand שיוצרת visitor כללי ובהתאם לקלט מהמשתמש מאתחלת אותו להיות הvisitor הרלוונטי וקוראת לפעולה שבו- זה מימוש של strategy. לאחר ביצוע השינוי נבצע הדפסה של איברי המערך- נרוץ עליהם אחד אחד בלולאה ונשלח לconsole.writeline שכפול של האיבר הנוכחי. לצורך זה נצטרך לממש פונקציית clone לint- מקבלת ערך ומחזירה new int. זה מימוש של prototype.

1. מדובר על מימוש של Chain of responsibility. ניצור מחלקה אבסטרקטית מסוג logType שמגדירה מה השלב הבא שלה בשרשרת, ומייחצנת פונקציה של handle – כלומר איך היא מתמודדת עם הלוג. כל סוג לוג יממש את הhandle בצורה שונה- יבדוק האם השגיאה היא ברמתו, במידה וכן יטפל בה (בהתאם להודעת שגיאה שהוא אמור להציג) ובמידה ולא יעביר אותה הלאה.
2. מדובר במימוש של Flyweight אשר מאפשר לנו לא לבזבז מקום,
3. **לא נלמד**
4. מדובר במימוש של .Templateניצור מחלקה אבסטרקטית של אימון, המכילה 4 פונקציות אבסטרקטיות ופרטיות– החלפת בגדים, שתיית מים, ביצוע תרגיל, מקלחת. בנוסף תהיה למחלקה פונקציית אימון משותפת לכל המופעים שלמעשה מריצה את כלל הפונקציות האבסטרקטיות על פי המימוש שלהם.

ניצור מחלקה עבור כל סוג אימון, שיורשת מהמחלקה האבסטרקטית שיצרנו. נתאים את מימוש הפונקציה לסוג האימון (למשל כמות מים ששותים, משך זמן המקלחת, סוג הבגדים וכמובן האימון עצמו).

כדי לבצע אימון ניצור מופע של אחת ממחלקות האימון ונריץ את פונקציית האימון.

1. לא נלמד
2. מדובר במימוש של Decorator. נממש ממשק לחלון שייחצן פונקציה של הצגת פרטי החלון. ניצור מחלקה אבסטרקטית של חלון כללי, אשר מחזיק מאפיינים בסיסיים על החלון (אורך, רוחב).

ניצור מספר מחלקות שיורשות ממחלקה זו, אחת מהן תהיה החלון הראשוני שלנו שמימוש הפונקציה בו תהיה הדפסה של אורך החלון. יתר המחלקות יהיו התוספות של החלון (מחלקה כל תוספת) ובהתאם ההדפסה תיג את הקישוט הרלוונטי ותקרא לפונקציית ההדפסה בשנית עם אותו המשתנה.

המימוש יהיה כשניצור מופע של חלון ריק, וכל פעם שנרצה להוסיף קישוט ניצור מופע של חלון עם הקישוט החדש ונשלח את החלון האחרון שיש לנו עד כה. בסוף , נצברים לנו הרבה חלונות.

1. מדובר על מימוש של Memento. ניצור מחלקת "לוח" ובה מטריצה של שלמים בגודל 5X5 שבכולה יש ערכי null פרט לנקודה המרכזית בה קיים הערך 1(3,3). למחלקה יש גם 2 משתנים, (X,Y) שמצביעים על המיקום הנוכחי של האובייקט במטריצה, ומשתנה המעיד על ערך האובייקט.

במחלקה מייצגת את המשחק ניצור מופע של מחלקת הלוח הפעיל, ומופע נוסף שהוא הלוח השמור (בהתחלה יאותחל לnull). נתחזק switch case שמבצע בדיקה לקלט מהמשתמש, שיקבל את הערכים הרלוונטיים. ערכים 1-4 יבצעו שינוי אינדקסים כמתואר בשאלה, וידפיסו את המטריצה החדשה. ערכים +,- יבצעו הגדלה או הקטנה של ערך האובייקט, וידפיסו את המטריצה גם הם. S ישמור בלוח השמור את הלוח הקיים וימשיך את המשחק בלוח הקיים, וs ישווה את הלוח הקיים ללוח השמור וידפיס אותו. שמירת הלוח באובייקט נוסף הוא למעשה מימוש memento.

1. מדובר על מימוש של דפוס Builder. נממש מחלקת "מחשב" שמכילה רשימה של רכיבי המחשב ומייחצנת פונקציה אחת לביצוע כל אחת מהפעולות הבאות: בניית מארז, הוספת לוח אם, הוספת מעבד, הוספת כ.גראפי, הוסף RAM ובדיקות. כל מתודה כזו תוסיף רכיב לרשימת הרכיבים של המחשב.

נממש מחלקה אבסטרקטית לבניית מחשב אשר מכילה אוביידט protected מסוג מחשב (כלשהו) ופונקציות זהות לפונקציות שפורטו, אך הן protected. מחלקה שתממש ממשק זה מייחצנת רק "בניית מחשב" ו"קבלת מחשב", כאשר בניית מחשב מריצה את כל הפונקציות הprotected לפי הסדר וקבלת מחשב מחזירה את אובייקט המחשב במידה ומאותחל.

ניצור מספר "בנאים" על פי סוגי המחשבים שאנחנו רוצים לאפשר ליצור, כאשר כל אחד מהם יורש מהמחלקה לבניית מחשב ושולח סוגים שונים של רכיבים לאותן המתודות, זאת במטרה לייצר מחשב מסוג ספציפי.

**שאלות בונוס:**

2. אפשר לעבור על הרשימה פעם אחת ובכל פעם לייצא את האיבר הנוכחי למשתמש בצד, למיין את ערך המחרוזת לפי סדר לקסיקוגרפי, ולייחצן את התוצאה לHASHMAP- המפתח הוא הסדר הלקסיקוגרפי והערך הוא כמות הפעמים שהסדר הזה הופיע. כל פעם שנחשב סדר נבדוק האם הוא קיים- אם כן נגדיל את הערך ב1 ואם לא נוסיף אותו עם הערך 1.

לאחר שעברנו על המערך יהיה לנו מיפוי מדוייק של כמה פעמים כל מילה מופיעה (בסדר כלשהו) במערך.