**Design Pattern**

במערכת שלנו השתמשנו בשני סוגים של Design Pattern חוץ מ-MVC ו-Singelton :

1. **The Observed Pattern:**

במסך ניהול השאלות המנהל יכול למחוק שאלה, להוסיף שאלה, לעדכן שאלה או לצפות בשאלה. לשם כך, הוא מתבקש להזין מספר השאלה שהוא רוצה לבצע עלייה אחת מהאפשריות שיש לו. אחרי בחירת מספר השאלה הנתונים של האפשריות מופיעות (אם הוא רוצה למחוק או לעכן או להוסיף שאלה) מקובץ JSON, כלומר הנתונים נשלפים מקובץ JSON ונכנסים ל-ArrayList של השאלות שנמצאת ב-SysData, מבצעים על השאלות את השינויים שהמשתמש רוצה לעשות, ואחר כך שומרים את השינויים עוד פעם בקובץ ה-JSON.

בעצם מחכים למספר השאלה שהמשתמש יקליד כדי לבצע עליו שינויים, לוקחים את הנתונים מקובץ JSON. ל-ArrayList שב-SysData ומעדכנים השינויים ומחזירים את ה-ArrayList המעודכנים לקובץ JSON.

**מיקום בקוד:** ב-SysData: Model.SysData.loadQuestion/Model.SysData.WriteQuestion .

1. **The Template Pattern:**

Knight ו- King ו- Queen מחלקות שיורשות ממחלקת Piece. בנוסף לכך, שלושת החלקים במשחק ממשים אותה פונקציה StrightMove ממחלקה Piece בנוסף לפונקציות שמאפיינות כל מחלקה בנפרד, כדי למנוע כפליות במימוש אותה פונקציה השתמשנו ב-Template Pattern ובו הגדרנו מחלקת ה-Piece שתהיה מחלקה אבסטרקטית ובה הפונקציה StrightMove היא פונקציה אבסטרקטית. המחלקות Knight ו- King ו- Queen שיורשות מהמחקלה Piece יורשות את הפונקציה גם.

**מיקום בקוד:** Model.Piece מחלקת האב (שיורשים ממנה) והמחלקות שיורשות ממנה: Model.Knight, Model.King, Model.Queen.