שאלה 3 :

במימוש הלוח האלקטרוני השתמשנו ב-design patterns הבאים :

- Singelton – השתמשנו בדפוס זה עבור המחלקה ColorGenerator ממנה התבקשנו ליצור עצם אחד בלבד. בדומה לנלמד בתרגול, בחרנו להשתמש בדפוס זה במימוש lazy initialization מכיוון שהעדפנו שאובייקט ייווצר רק לאחר הקריאה הראשונה לבנאי ולא באופן אוטומטי בתחילת התוכנית. בנוסף מכיוון שאין צורך לאפשר הורשה הגדרנו את הבנאי כפרטי.
- Observer – בשאלה אנו נדרשים לשנות בסדר מסוים (ארבע אופציות נתונות) את צבעם של הלוחות בהתאם לשינוי בצבעו של האובייקט ColoeGenerator. החלטנו להשתמש בדפוס מכיוון שזהו מקרה קלאסי בו ישנם סובייקט (ColorGenerator) המשנה את מצבו באופן תדיר, וישנם אובייקטים משקיפים (Panels) המגיבים לעדכון במצבו של הסובייקט.
לכן המחלקה ColorGenerator יורשת מהמחלקה הנתונה ע"י Java – Observable ,
והמחלקה Panel מממשת את הממשק הנתון ע"י Java – Observer .
המימוש שלנו עבור ColorGenerator כלל מימוש רשימה של אובייקטים צופים, ודריסה של המתודות addObserver(Observer) , deleteObserver(Observer) , notifyObservers() .
המימוש עבור Panel כלל את מימוש הפונקציה update(Observable,Object) .
- Strategy – בשאלה נתונות לנו 4 אופציות בהן ניתן לאפשר למשתמש בלוח המודעות לבחור את סדר עדכון צבעם של הלוחות. השתמשנו בדפוס זה בשביל לבצע אנקפסולציה לארבעת האלגוריתמים הנ"ל שדרכם אנו קובעים את סדר העדכון ומיד מעדכנים את הלוחות ואת צבעם. המימוש בו בחרנו זהה למימוש שהוצג בתרגול : ארבעת האלגוריתמים יורשים מהממשק שהגדרנו NotifiyObserversAlgorithm המגדיר פעולה אחת בודדת notify .
האלגוריתמים מומשו בתור מחלקות נפרדות היורשות מממשק זה ולמחלקה ColorGenerator קיים שדה שמתעדכן בהתאם להחלטת המשתמש והוא מכיל את האלגוריתם המתאים.