**OOP- תרגיל בית 4**

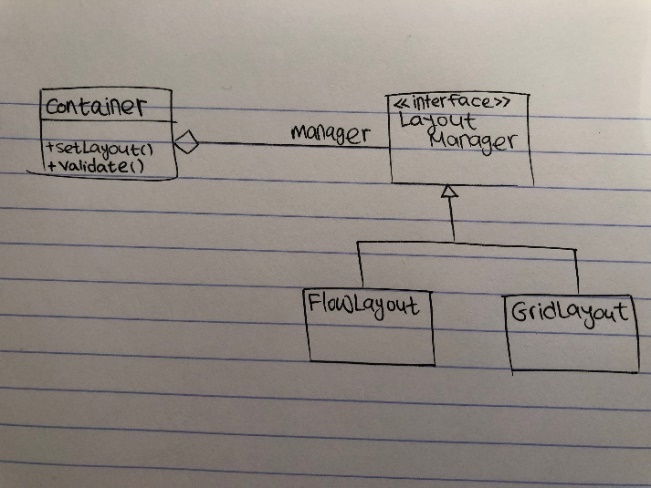
**מגישות:**

**ענבר קזס 203265947**

**לוטם מנסן 312142953**

שאלה 1

1. ה-design pattern הממומש כאן הוא **strategy**. הוא מגדיר משפחה של התנהגויות/אלגוריתמים כך שניתן להחליף ביניהם בקלות בזמן הריצה. במקרה זה יש לנו שתי אופציות ל-layout של רכיבים בתוך containers. ה-container מכיל מופע אחד של LayoutManager ויכול בזמן הריצה לשנות את הסוג שלו בעזרת מתודת setLayout(LayoutManager manager).
2. ניתן לראות בדיאגרמה את שתי המחלקות המממשות את הממשק LayoutManager, ואת ה- containers המכיל מופע יחיד manager אשר יכול להשתנות בזמן ריצה.



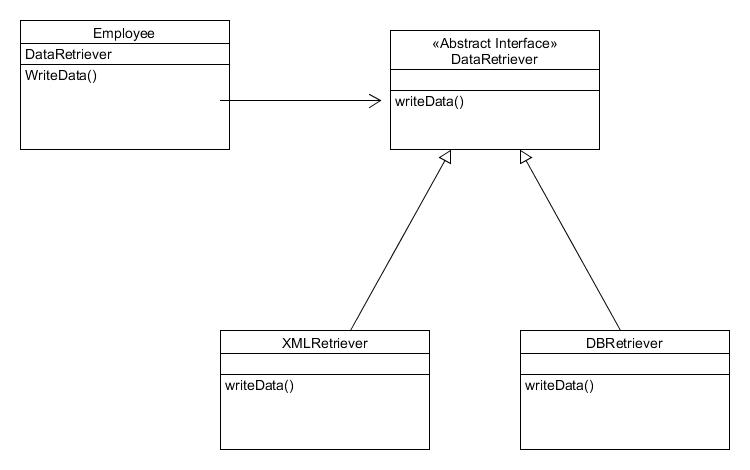
שאלה 4

1. העקרון המופר כאן הוא **עקרון** **הפתיחות/סגירות**. אם בעתיד נרצה לשמור את נתוני העובד בצורה שונה, נניח באמצעות קובץ טקסט רגיל, אז נצטרך להוסיף מתודה חדשה למחלקת Employee ונצטרך לשנות את השימוש במחלקה זו במקומות רבים בתוכנה.

הצעה לפתרון חלופי בעל תכן מוצלח יותר:

שימוש ב- Interface לכתיבת נתוני עובד, אשר יכיל מתודה בשם writeData. כל מחלקה שתרצה לשמור נתוני עובד למקום מסויים תממש ממשק זה. יממשו אותו מחלקות DBWriter, XMlWriter, txtWriter וכו'..

המחלקה Employee תחזיק שדה של אובייקט המממש את הממשק. כאשר תיקרא המתודה WriteData של Employee תקרא המתודה של הממשק. כעת אם נרצה להוסיף יכולת חדשה לשמירת המידע מהעובד ניצור מחלקה חדשה המממשת את הממשק ונחליף את האובייקט שה-employee מחזיק. להלן תרשים UML של הפתרון החדש:



1. הקשר בין עקרון האחריות היחידה לבין design pattern chain of responsibility-

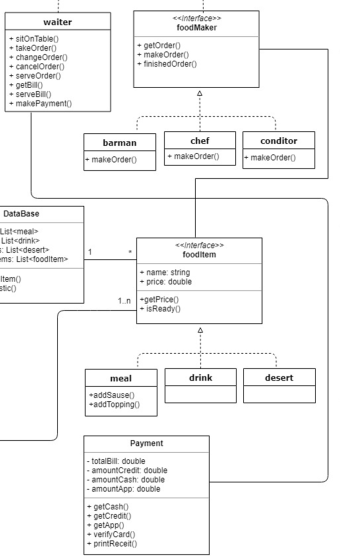
לפי העקרון לכל מחלקה יש אחריות יחידה, כלומר היא מבצעת שינויים בתחום יחיד בלבד.

ה-design pattern אומר שיש לבצע העברת אחריות ממחלקה למחלקות אחרות, ושכל מחלקה תטפל באספקט אחד בלבד של הבעיה המתגלגלת. כלומר ה-design pattern מהווה מימוש של העקרון.

1. דוגמא לעקרון creator מ-GRASP:

בתכנית המסעדה המחלקה waiter יצרה מופע של payement והשתמשה בו. זה תואם את התנאי שהמלצר משתמש בחשבון, וספציפית בתכנית כולה רק המלצר משתמש בו, ולכן הוא יוצר אותו.

מתוך class diagram:



דוגמא לעקרון Information expert:

בתוך foodmaker כתבנו פוקציית finishedOrder. הפונקציה מחכה למידע מכל נותני השירות ורק כשכולם סיימו להכין אותה היא מודיעה למלצר שהכל מוכן. זה ממלא את העקרון כי רק מחלקה זו יכולה לאסוף מידע מכל המחלקות היורשות ממנה ולדעת אם כל פריטי האוכל של ההזמנה מוכנים, כלומר יש לה הכי הרבה מידע לצורך ביצוע הבקשה.

מתוך class diagram:

