OOP 4

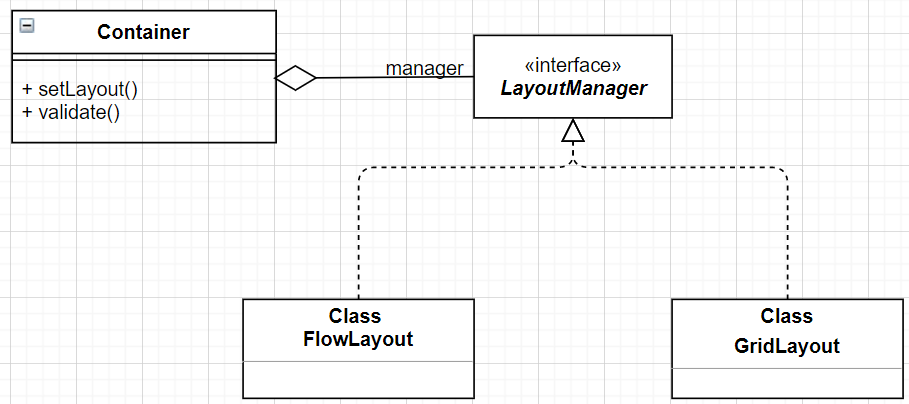
יואב בנימין

תז

**שאלה 1**

סעיף א

ה design pattern שממומש בשאלה הוא מסוג Strategy. הסיבה לכך היא שהבעיה המוצגת היא החלפת ה- Layout על ידי המשתמש בזמן הריצה בצורה גמישה.  
נשים לב כי יש לנו 2 מחלקות אשר מממשות את אותו הממשק והן קשורות אחת לשניה ונבדלות בהתנהגות שלהן.

סעיף ב

**שאלה 2**

סעיף א

**שאלה 3**

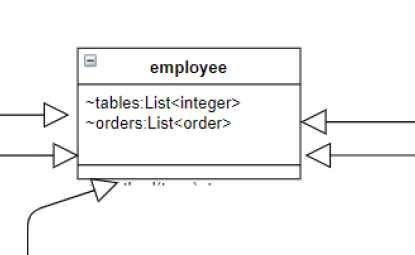
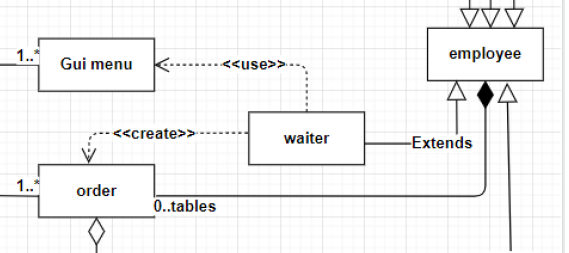
**שאלה 4**

בתרגיל בית 3 הגדרנו מחלקת employee אשר ממנה יורשים כלל משתמשי המערכת במסעדה.  
מחלקה זו מחזיקה את כל המידע על ההזמנות הקיימות במערכת ומצבן, המידע על כלל השולחנות במסעדה ומצבם.

מסיבה זו מחלקת employee מקיימת את עיקרון ה Information Expert בתכן של התוכנה שעשינו.

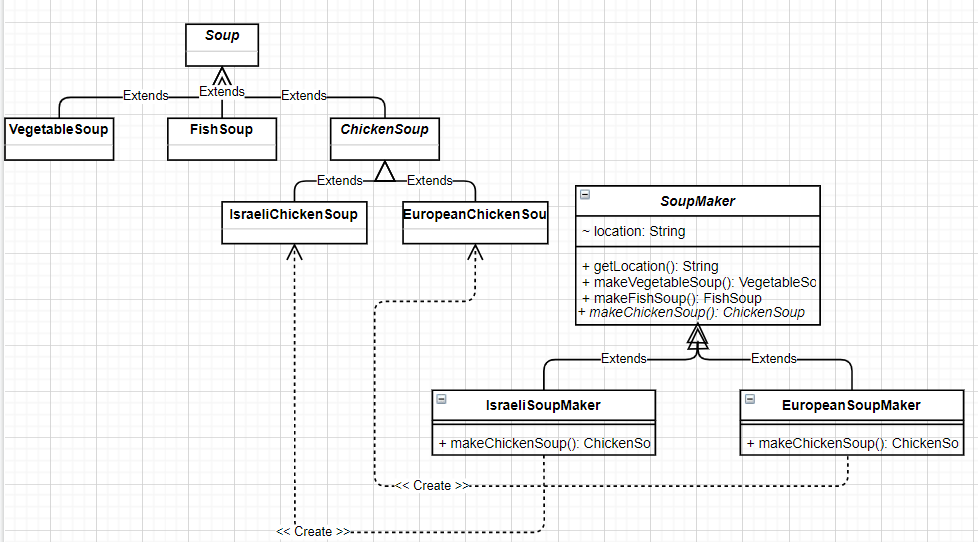
מחלקת waiter שיורשת ממחלקת employee היא זאת אשר יוצרת הזמנות חדשות, מטפלת בסטטאוס של ההזמנות ומעדכנת אותן.  
מסיבה זו מחלקת waiter מקיימת את עיקרון ה creator בתכן של התוכנה שלנו.

ניתן לראות זאת מהתרשימים הבאים:

****

**שאלה 5**

סעיף א

ה design pattern שמממוש כאן הוא factory. בעיית התכן שאנו פותרים בעזרת design pattern זה הוא הפרדת יצירת האובייקטים השונים שלנו והקונפיגורציה שלהם. בעזרת מימוש זה אנחנו מורידים את רמת הצימוד של התכן שלנו, ומאפשרים רמת אבסטרקציה של יצירת אובייקטים.

סעיף ב

מחלקת Calendar מממשת את ה design pattern מסוג singleton.

ניתן לראות כי הקריאה הראשונה היא למתודה של getInstance , זהו מימוש של singleton כמו שלמדנו בהרצאה. מימוש זה הגיוני עבור אובייקט מסוג לוח שנה אשר כדאי שיהיה מופע בודד שלו שכל רכיבי התכנית יוכלו להשתמש בו.