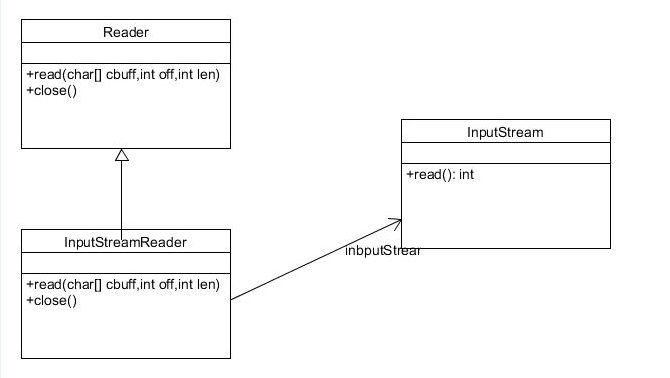
תכן ותכנות מונחה עצמים – תרגיל רטוב מספר 4:

מגיש: תומר גולני

ת.ז: 302658315

שאלה 1:

1. design pattern הממומש כאן הוא Adapter. והבעיה שהוא בא לפתור היא חוסר ההתאמה בין המפרט של InputStream ו – Reader
2. class diagram:



הסבר:

המפרט Reader לא תואם את המפרט של Inputstream , אך עדיין רוצים שתהיה אפשרות להשתמש במתודה read. לכן המחלקה היורשת Inputstreamreader דורסת את המתודה Read. כעת תתבצע הרכבה של InputStreamReader על ידי המחלקה InputStream בכך שמיוחזק רפרנס לאובייקט פנימי. זהו בדיוק מבנה שך adapter.

שאלה 2:

הסבר על התוכן והמימוש:

לצורך מימוש הצאט הגדרנו מספר רכיבים המגדרים את ממשק המשתמש:

1. user.java – קובץ ממש כל user בצאט. מחלקה זו מממת user בעזרת JPanel כל יוסר שונה בכך שיש לו מספר אחר. ה- Jpanel מורכב מ-2 Jlabes אחד עם תיבת הטקסט בה המשתמש כותב את הודעותיו והשני עם תיבת הצאט בה הוא רואה את כלל ההודעות שנשלחו על ידי כולם כולל אותו בכחול.

כמו כן labal של תיבת המשתמש משמש כ- Observer. כאשר המשתמש לוחץ על enter אז מתבצעת פונקצית actiionhandler אשר משנה את תיבת המצאט עם ההודעה החדשה.

1. Handler.java – מחלקה זו אחראית בעצמם לממש את הסובייקט בdesign patter של ה observer.

היא מכילה רשימה של כל ה- users במערכת הצאט ובהינתן שאחד המשתמשים לחץ על enter אז ה-Handler דואג להוסיף לכולם את האת ההודעה לתיבת הטקסט.

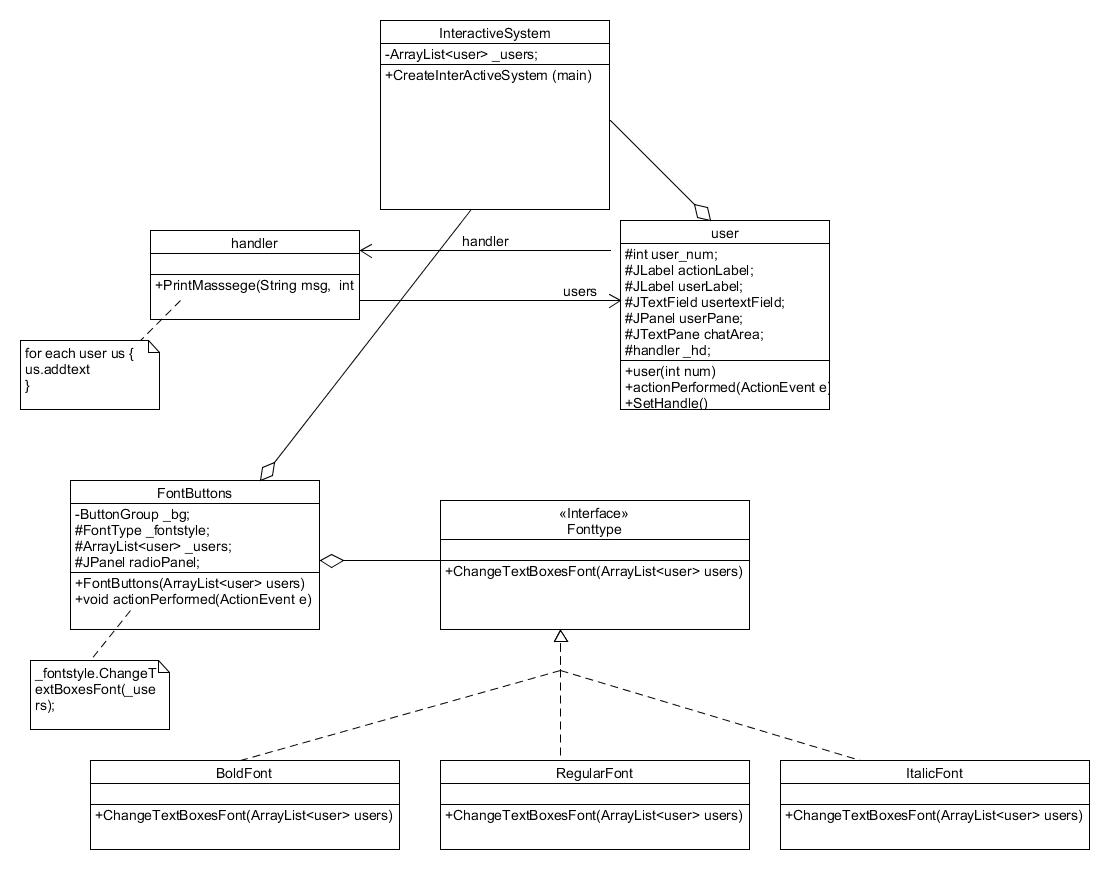
כמו כן עבור תיבת הטקסט של כותב ההודעה , ההודעה תתיצבע בכחול.

1. FontType.Java - מחלקה הממשמת את ה- design pattern strategy עבור שינוי במערכת הצאט.

המחלקה FontType היא אבסטרקטית והיא מכילה שלוש מחלקות יורשות אשר כל אחת מממשת הכנסת Font שונה שלמערכת. (רגיל,מובלט,מיוחד)

1. FontButtons.java – מחלקה למימוש כפתורי רדיו במערכת אשר באמצעותם ניתן לבחור את ה-Font הרצוי במערכת. היא מחזיקה שדה שהוא ממשק ה – FontType ובאמצעותו נקבע הפונט במערכת.
2. InterActiveSystem.java – הקובץ הראשי למימוש המערכת, בקובץ זה בונים בונים את ה- Jframe עם שלוש users ותוספת של label של כפתורי הרדיו.

תצוגה של ה- design pattern:



שאלה 3:

תיעוד חיצוני + הסבר על ה- design pattern שנבחר:

ה-Design Pattern המתאים למימוש התוכנית הוא מסוג **composite**

סוג זה של design מסדר את האובייקטים המרכיבים אותו במבנה של עץ כדי לבטא את ההיררכיה בין שלם לחלקיו. בכך ניתן תהייחס באופן אחיד לאובייקט פשוט ולאובייקט מורכב.

במקרה שלנו , כל אובייקט "מורכב" מייצג ביטוי אריטתמי. (חיבור,חיסור,כפל,חילוק,מינוס) והאובייקט הפשוט (העלה בעץ) הוא מסוג Number. (int או Double) כל ביטוי אריתמטי בעת יצירתו יכול להכיל ביטויי אריתמטיים נוספים ובצורה זו נוצר מבנה של עץ, כאשר העלים הם מספרים מסוג Number. בכך התאמנו את הדרישות ל –Composite.

כאשר אחת המתודות של ביטוי אריתמטי תיקרא(eval או toString) המתודה תיקרא למתודות המתאימות של בניו, עד העלים.

את Expression ממימשתי באמצעות מחלקה אבסטרקטית בשם זה, אשר מכילה את שתי המתודות שנדרשו בצורה אבסטרקטית. בצורה זו מובטח שכל exression שירחיב אותו יהיה חייב לממש את המתודות. ואנו בודקים ע"י checkrep שהן אינן מחזירות null.

שאלה 4:

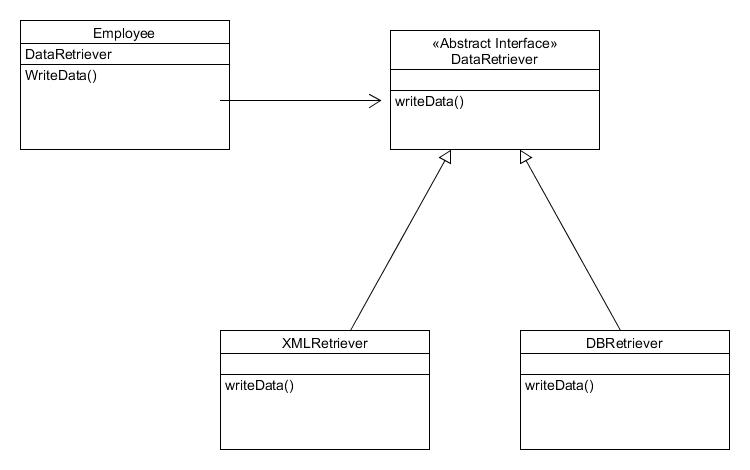
1. העקרון המופר כאן הוא עקרון **הפתיחות\סגירות**.

הסבר:

אם בעתיד נרצה לשמור את נתוני העובד בצורה שונה, נניח קובץ טקסט רגיל, אז נצטרך להוסיף מתודה חדשה למחלקת employee ונצטרך לשנות את השימוש במחלקה זו במקומות רבים בתוכנה.

הצעה לפתרון מוצלח יותר:

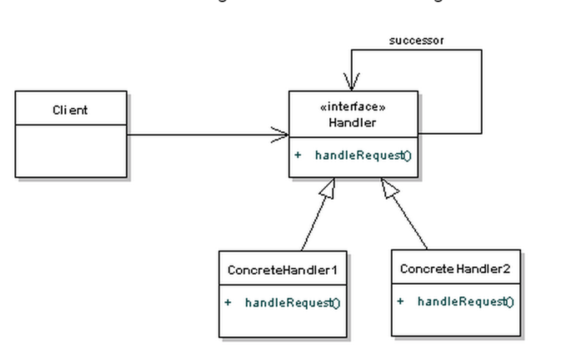
* שימוש ב- Interface לשמירות נתונים, אשר יכיל מתודה בשם WriteData. כל מחלקה אשר תרצה לשמור נתונים על העובד תממש ממשק זה. כלומר ימומש בה ממשק DBWriter ו XMlWriter ו txtWriter ... המחלקה employee תחזיק שדה של אובייקט המממש את הממשק. כאשר תיקרא המתודה WriteData של Employee תקרא המתודה של הממשק.
* כעת אם נרצה להוסיף יכולת חדשה לשמירת המידע מהעובד כל שיש לעשות זה ליצור מחלקה חדשה המממשת את הממשק ולהחליף את האובייקט שהemployee מחזיק.
* תרשים uml:



סעיף ב:

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

העקרון אומר שלכל מחלקה יש אחריות יחידה,וה- design pattern אומר שיש לבצע העברת אחריות ממחלקות למחלקות אחרות:



ה- design pattern הוא בדיוק מימוש של העקרון.

1. דוגמא לביטוי עקרון ה- creator:

יצירת אובייקט Item על ידי אחת המחלקות מטבח\דוכן שתייה\קינוחים מתוך מחלקת ItemDescription המועברת מ- Menu.

תנאים 1 ו4 מתקיימים:

"מופעים של Y מכילים או "מקבצים" באופן כלשהו מופעים של X":

Department מקבץ את כל מופעי Item

"למופעים של Y יש את המידע המשמש לאתחול מופעים של X":

הידע ליצירת Item מתוך ItemDescription צריך להיות אצל המחלקה המתאימה.

בתרשים:

Department

Item

ItemDescription

+createItem(ItemDescription)

+getReadyItems()

* דוגמא ל information expert:

מחלקת ה PaymentSystem אשר אחראית על חישוב המחיר הכללי של ההזמנה והדפסת החשבון. למחלקה הזו יש את הגישה לכלל המידע ולכן על פי העקרון הנלמד בתרגול האחריות על חישוב המחיר מוטלת עליה. למחלקות אחרות כגון Order אין את המידע המלא למרות שיש להן תפקיד בהוצאת החשבון ולכן האחריות הזו לא נמצאת אצליהן.

