תרגיל 3 –קורס ג'אווה ואינטרנט

פרטי הגשה

שם: אלון בן הרוש

ת.ז: 312533698

מייל: alonbh5@gmail.com

קישור לרפוזטורי ב GITHUB : https://github.com/alonbh5/SuperDuperMarket-Web-Tomcat

הערות: **לא מומשו בונוסים**

קישור לאתר : http://localhost:8080/WebAppSDM

הסבר כללי על המערכת

הפרויקט מורכב מ-3 מודלים עיקריים :

- **engine** –ה"מנוע" של המערכת, מכילה את כל הלוגיה שלה ותפקידה לנהל את המערכת בהתאם לפקודות ה-UI, מכיל 3 תיקיות עיקריות:

  1. תיקיית **engine** שמכילה את מצבור המחלקות שמרכיבות את המערכת (חניות, הזמנות, מוצרים וכו'). בתוכה נמצאות המחלקה **MainSystem** והמחלקה **SuperDuperMarketSystem** שהן היחידה שחשופה (public) ורק איתה מתנהלת מערכות ה-UI השונות.
  2. תיקיית **exceptions** שמכילה את מצבור החריגות שהמערכת עלולה לזרוק לUI.
  3. תיקיית **generatedClasses** שנוצרה ע"י JAXB ומכילה את כל המחלקות שקבצי ה-XML עושים בה שימוש.

- **mediatorClasses** – עוזרת לתקשורת נוחה בין המערכת ל-UI, מכילה מחלקות שמייצגות אובייקטים במערכת, מבלי לחשוף את האובייקטים של מערכת המנוע עצמם. כל המחלקות שם מכילות שדות (public final) וכן public c'tor. כאשר ה-UI מבקש מידע מסוים על אובייקטים במערכת, הוא יקבל אותם בצורה של אובייקטים ממחלקה זו, וכן עבור פונקציות מסוימות המערכת מבקשת מידע בדמות אובייקטים ממודול זה.

- **WebAppSDM** –UI מסוג Web App שנבנה באמצעות JS,Servlet,Css,jQuery,Html, בתוכה מכילה מספר סרבלטים שמתקשרים עם הדפדפן, בתיקייה ה-web נמצאת תיקיית Pages בה נמצאים הקבצים הסטטיים.

הסבר על המחלקות העיקריות

מודול WebAppSDM

- תיקיית **Servlets** שמכילה 7 סרבלטים :
  - AllAreasServlet – דואג לקבל ולשלוח את כל האזורים שטעונים במערכת, השליחה נשלחת ע"י שיטת PULL
  - GetUserTypeServlet – סרבלט שדואג לשלוח למבקש JSON שמכיל את הפרטים הנוכחים של השרת על הלקוח (שם,מס' סידורי,אזור נוכחי, וסוג משתמש) ע"י כך המערכת יודעת בעת עליית עמודים 2,3 איזה אפשריות להציג למשתמש (למשל רק לקוח יכול לבצע הזמנה, ורק מוכר מקבל התראות)
  - GetZoneInfoServlet – הסרבלט העיקרי לקבלת מידע מהשרת, וכן לבקשת עדכון מהשרת, בעת פנייה אליו מעמיסים עליו פרמטר קבוע "infoType" שמציין איזה בקשה אנחנו רוצים לבצע, למשל על מנת לייצר הזמנה נשלח בקשת POST עם פרמטר הבקשה "createOrder", ועל מנת לקבל את כל המוצרים באזור נשלח בקשת Get עם פרמטר בקשה "items" , הסרבלט ידע ע"י תכונות המשתמש שנמצאים על ה-Session איזה מידע רלוונטי לשלוח לו בדמות JSON.
  - LocalZoneServlet – סרבלט שמקבל בקשת משתמש להיכנס לאזור מסוים (מעמוד 2) , כאשר הוא מבקש זאת הוא מטעין על ה-Session את שם האזור הנוכחי בו נמצא המשתמש, ומפנה אותו לעמוד 3 (שכעת בעת העלייה יודע איזה מידע לבקש מהשרת מבלי לשים לב שהשרת יודע איזה מידע אזורי לשלוח לו).
  - LoginServlet – סרבלט שמפנה מעמוד 1 לעמוד 2, כאשר עושה זאת , הוא מעמיס על ה-Session את כל הפרטים הרלוונטים ושולח למנוע המערכת לייצר משתמש חדש (לקוח או מוכר בהתאם לבקשה)
  - UploadXML – במידה והמשתמש הוא מוכר, ניתן לפנות אל הסרלבט הזה על מנת לעלות קובץ XML חדש, הסרבלט ידע בהקשר המשתמש ששלח לו את הבקשה לפנות למנוע ולטעון את הקובץ ע"י אותו משתמש – במידה ויקבל חריגה בעקבות בעיה עם הקובץ הוא ישלח אותה בדמות JSON על התשובה.
  - UserListServlet - סרבלט עבור עמוד 2, דואג לקבל ולשלוח רשימת משתתפים נוכחית.
- תיקיית **web**
  - תיקיית Common עם קבועים שהסרבלטים וקבצי ה-JS משתמשים בהם (שמות בקשות, תכונות וכו')
  - תיקיית Pages עם כל העמודים הסטטיים – HTML,CSS,JS (מסודרים לפי סוג לקוח), רוב עבודת קבצי ה-JS היא :
    1. לייצר את העמוד בהקשר של המשתמש (לקוח או מוכר).
    2. לשלוח בקשות AJAX לסרבלטים כאשר המשתמש לוחץ על אחד הכפתורים.
    3. לטעון עמודי HTML סטטיים ולמלא אותם במידע שהתקבל מהסרבלטים.
    4. לבצע פעולות חוזרות כמו בדיקת התראות, בדיקת אזורים ומשתמשים מחובירם.

מודול engine

- תיקיית **engien** עבור ניהול המערכת, הסבר קצר על רוב המחלקות בה:

- MainSystem – שדרוג המערכת הקודמת, כעת מחלקה זאת מכילה 2 אוספים – לקוחות ומוכרים, יכולה לשלוח את ה- SuperDuperMarketSystem לפי אזור (שכן הם נמצאים אצל המוכרים)
- FileHandler – משומש ע"י המערכת לגישה וקבלה של קובץ ה - XML.
- LoadXmlTask – Task שטוען למערכת את כל הפרטים מה-XML, בהתחלה ה-UI קוראת למטודה UploadInfoFromXML עם נתיב הקובץ, שם המנוע יוצר את הTASK ושולח חזרה לUI (למטודה bindTaskToUIComponents) את הTASK על מנת לחווט את כל המידע הרלוונטי (מקשר את אזור ההודעות להודעות של ה-TASK, וכן את מד ההתקדמות לTAS) וכן להגדיר מה לעשות במידה והTASK הצליח ומה לעשות במידה ולא.
- Item – מייצג פריט בסיסי, בעל מסדר סידורי, שם ואופן קנייה (enum).
- Person – מייצג אדם, יש לו **מספר סידורי** ייחודי ושם
- Customer – מרחיב את Person, ובנוסף יש לו מפה של היסטורית הזמנות שביצע, רשימה של פידבקים שנתן,וארנק.
- Seller – מרחיב את Person, יש לו אוסף SuperDuperMarketSystem אליה נוסף אוביקט כזה כאשר הוא מעלה קובץ (באמצעות מטודת העלה אצלו),כמו כן מכיל ארנק, רשימת פידבקים שקיבל, רשימת התראות שיקבל ורשימת חניות שיש לו.
- ProductInSystem – מייצג פריט שמור במערכת , מכיל הפנייה לפריט הבסיס היחיד **item** (וכך מקבל את כל פרטיו), בנוסף מכיל נתונים שמתעדנים ע"י המערכת כמו **מספר חניות שמוכרות** את הפריט במערכת, מספר **פעמים שנמכר** במערכת, וכן נכון לחלק זה של התרגיל **הפנייה לחנות שמוכרת את הפריט במחיר הזול ביותר** (מתעדכן ביצירת המערכת מקובץ הXML, וכאשר מתבצע שינוי כלשהו על הפריט). זאת במטרה לייעל את ההזמנה הדינאמית בבונוס התרגיל (ניסיתי להימנע כרגע מלחפש כל פעם את הפריט הזול ביותר, עלול להשתנות בהתאם להמשך התרגיל).
- ProductInStore - מייצג פריט שמור בחנות , מכיל הפנייה לפריט הבסיס היחיד **item** (וכך מקבל את כל פרטיו), כמו כן שם נמצא המחיר שלו **בחנות**, כמות **הפעמים שהמוצר נמכר וכן מצביע לחנות** שמוכרת אותו.
- ProductInOrder – מייצג פריט בתוך הזמנה , מכיל הפנייה לפריט סופי מסוג **ProductInStore** (וכך מקבל את כל פרטיו), כמו כן כאן נמצאים הכמות שנקנה בה בהזמנה והמחיר של הפריט X כמות, בנוסף ערך בוליאני האם הגיע מהנחה או לא.
- Store – מייצגת חנות במערכת, בעלת **מספר סידורי** ייחודיי, שם מיקום, רווח ממשלוחים וPPK. בנוסף מתוך מחשבה על יעילות חיפוש מהירה יש לה Map של היסטורית הזמנות מסוג **<Order, מס' הזמנה>**, וכן Map של מוצרים בחנות מסוג **<ProductInStore, מס' מוצר>** וסט של הנחות שהחנות מציעה. מממשת את הממשק Coordinatable,
- Order – מייצג הזמנה במערכת, מכילה **מספר סידורי** ייחודי, את שאר פרטי ההזמנה כמו לקוח, תאריך וסכומי קנייה , כמו כן מכילה **סט של חניות** שהשתתפו בהזמנה, **סט של מוצרי ProductInOrder** (שכן אין חשיבות לסדר

בהזמנה ואין טעם לשמור מספר חניות/מוצרים זהים) בנוסף היא מממשת את הממשקים **Coordinatable**.

- SuperDuperMarketSystem – המערכת הראשית איתה עובדת ה-UI, בעקבות שינויי התרגיל הפכה למערכת אזורית, יש לה מספר פונקציות עזר private אבל הרוב כן חשופות ל-UI, מכילה שדה בוליאני "נעול" שיישתנה רק לאחר שיטען קובץ XML תקין, השדות האחרים שלה כולם מפות מתוך מחשבה שלכל פריט במערכת יש מסדר סידורי ייחודי וכך יהיה ניתן לגשת אליו ב-$\theta(1)$.
  - **<מס' מוצר, ProductInSystem>** עבור מוצרי המערכת השונים.
  - **<מס' חנות, Store>** עבור החניות השונות.
  - **<מס' הזמנה, Order>** עבור מוצרי המערכת השונים.
  - **< Point,Coordinatable>** עבור ייצוג מפה להמשך, כרגע מכיל רק חניות.
- Discount - מייצג הנחה, להנחה יש שם , סוג הנחה (eNum), ProductYouBuy וסט של ProductYouGet.
- ProductYouBuy – מייצג מוצר שאתה צריך לקנות על מנת לקבל הנחה, יש לו בפנים את מוצר הבסיס Item, וכמות אותה צריך לקנות
- ProductYouGet – מייצג מוצר שאתה יכול לקבל מההנחה, יש לו בפנים את מוצר הבסיס Item, מחיר לפריט אחד בהנחה וכמות של מוצרים אותה ניתן לקבל.
- Wallet – מייצג ארנק, מכיל יתרה נוכחית ורשימת תנועות בחשבון
- Transaction - מייצג תנועת חשבון שמכילה את כל המידע הרלוונטי
- FeedBack - מייצג פידבק, מכיל פרטים על השולח, המקבל, פרטי הזמנה ממנה נשלח הפידבק.

- תיקיית **exceptions** עבור החריגות שהמערכת עלולה לזרוק לUI. בחרתי לסמן את כל החריגות כ-Checked על מנת לזכור לטפל בחריגות ברמת ה-UI, רשימת החריגות בקצרה, רובם נזרקים בקבלת קובץ ה-XML:
  - DuplicateItemIDException – התגלו 2 פריטים עם אותו מסדר סידורי
  - DuplicateItemInStoreException – חנות מוכרת 2 פריטים זהים
  - DuplicatePointOnGridException – חנות/הזמנה מנסה לגשת למיקום תפוס
  - DuplicateStoreInSystemException – התגלו 2 חניות עם אותו מסדר סידורי
  - ItemIsNotSoldAtAllException -יש מוצר במערכת שלא נמכר בכלל
  - NegativePriceException – התקבל מחיר שלילי או אפס.
  - NoValidXMLException – אין קובץ XML תקין במערכת, אן שהמערכת נעולה.
  - PointOutOfGridException – התקבלה נקודה מחוץ לטווח המוגדר
  - StoreDoesNotSellItemException – יש חנות שלא מוצר שום פריט
  - StoreItemNotInSystemException – חנות מנסה למכור מוצר שלא במערכת
  - WrongPayingMethodException – התקבלה אופן קנייה לא תקין (בקובץ)
  - CustomerNotInSystemException-התקבל לקוח שלא נמצא במערכת
  - DuplicateCustomerInSystemException–התקבל לקוח עם אותו ID פעמיים.
  - ItemIsTheOnlyOneInStoreException–מנסים למחוק מוצר אחרון בחנות.
  - IllegalOfferException–התקבל סוג הנחה לא תקין.

- NegativeQuantityException– התקבלה כמות לא תקינה בהנחה.
- NoOffersInDiscountException– התקבלה הנחה ללא הצעות.
- OrderIsNotForThisCustomerException– מנסים להוסיף הזמנה להיסטורית הזמנות של לקוח כאשר ההזמנה לא בוצעה על ידי אותו לקוח.

מודול mediatorClasses

כפי שתואר לעיל, מודול זה מחזיק מספר מחלקות שמייצגות את האובייקטים השונים של המערכת בגרסה פשטנית יותר, המחלקות מייצגות פריטים, חניות והזמנות כאשר רוב השדות בהם הם פרימיטיביים או אוסף של מחלקות אחרות מתיקייה זו, כמו כן כולם public final ובעלי public ctor. יש 2 מטרות עיקריות עבור מחלקות אלו:

1. שליחה מה-UI למנוע המערכת – חלק מהמטודות במערכת המנוע דורשות לקבל אובייקטים ממודול זה, למשל ביצירת הזמנה על מערכת ה-UI לשלוח **אוסף של פריטים** רצויים להזמנה, המערכת תדע לקבל את המידע ולהמיר אותו לפריטים הרלוונטיים אצלה, וזאת מבלי לחשוף ל-UI את הפריטים עצמם ובכך שומרת על כך שתפקיד ניהול המידע נשאר רק לה. כמו כן תדע להוציא חריגות מתאימות במידע וקלט לא תקין הגיע.
2. שליחה מהמנוע ל-UI – עבור כל מטודות קבלת המידע של המערכת (getters למיניהם), המערכת תייצר עותק שמייצג את המידע הרלוונטי ותשלח אותו ל-UI לעשות בה כרצונה, בכך נמנע מצב שה-UI תוכל לפנות למידע רגיש במערכת – שכן היא מקבלת רק העתק פשטני.