הסבר על התוספות לתרגיל בית גדול 3 – יגל שפר – 315398404.

דרישות מערכת: את התוכנה יש להריץ על גרסאת Java se 15 לפחות. יש לבדוק את החיבור לאינטרנט לפני ובמהלך ההפעלה.

הפעלה: תקיית ההפעלה היא תקיית target בתוך הפרוייקט. כאשר מעתיקים את הקובץ יש להעתיק יחד איתו את כל התקייה. קובץ ההפעלה הוא javaEat.jar ואפשר להפעיל אותו בלחיצה כפולה או דרך cmd, ללא צורך בארגומנטים מיוחדים.

1. **CSS. הפרוייקט כולו מעוצב באמצעות מתודות setStyle. אין פה שימוש בסשן בילדר כלל. הסיבה המרכזית לזה היא שהתחלתי לבנות את הפרוייקט באמצעות קוד תוך כדי turturial שלימד איך משתמשים בFX בקוד. באמצעות הקוד יצרתי אובייקטים מובנים, כמו שניתן לראות בתרגיל , שהיה הרבה יותר מסורבל מבחינתי לייצר כל פעם מחדש בסשן בילדר. ולכן ומסיבה זו הפרוייקט מבוסס על טהרת הCSS. מבחינת מבניות החלון – יש window אחד שאליו ניתן לגשת ע"י ()App.getWindow, scene אחד ועליו יושב באופן קבוע BorderPane אחד, אליו ניתן לגשת ע"י ()App.getLayout. כל מעברי החלונות, לרבות פופאפס, מתבצעים ע"י חילוף איזור מרכז הlayout. סרגל הכלים יושב בחלק העליון, אם יש מצב היברידי כתובת הIP תופיע בחלק התחתון ואצל האדמין יופיע לוגו המסעדה מצד שמאל למטה.**

2. **MAVEN. זו תוספת מאילוץ. לא הסתדרתי עם הקומפילציה ויצירת הקובץ כשג'אווה בנוי ללא MAVEN, וMAVEN זו הדרך היחידה שבה הצלחתי לקמפל את הקובץ.**

3. **אוטומציה. לעובדים (טבחים ושליחים) יש תת מחלקה חדשה ובה שדה בוליאני הקובע האם הם כרגע עמוסים או לא (ירד – משכורת יחד עם מערכת קביעת מחירים ובדיקת רווחיות). שלבי האוטומציה מתוארים לפי השתלשלות זו:**

   א. **כאשר נוספת הזמנה חדשה היא נכנסת לתור ממתינה להכנה. אם באותו רגע יש טבח פנוי מתחיל תהליכון (ממומש באמצעות המחלקה Thread)בו הטבח "ישן" למשך זמן הכנת המנה. בזמן זה יהיה תפוס וישוחרר בסוף פרק הזמן. בסוף כל שינה כזו הטבח בודק אם יש עוד מנה שממתינה לו, ואם כן יתחיל להכין אותה.**

   ב. **כאשר טבח מסיים מנה הוא מעביר אותה לתור הממתין להחלטת AI לגבי חלוקה למשלוחים. ממומש כHASHMAP כאשר המפתח הוא איזור שילוח והערך הוא תור הזמנות. ההזמנה תמתין בתור להחלטה עד שיתפנה שליח מהאיזור אליו מיועדת המנה (מומשה בRESTAURANT מתודה הבודקת לפי שכונה לאיזה איזור משלוחים שייך הלקוח).**

   ג. **כשמתפנה שליח לאותו איזור מתקבלת החלטה, נוצרים משלוחים והמשלוחים עוברים לתור ממתינים לשילוח. השליח יוצא עם ההזמנה הראשונה מאותו תור. התור ממומש באופן דומה להמתנה לקבלת החלטה – HASHMAP כאשר המפתח הוא איזור והערך הוא תור משלוחים.**

   ד. **השליח מפעיל תהליכון משלו בו הוא "ישן" פעמיים את זמן ההגעה ליעד – הלוך וחזור. בין לבין מבצע התהליכון את מתודת DELIVER המאשרת שהמשלוח הגיע ליעדו. כאשר יגיע השליח חזרה, יחזור להיות פנוי ויבדוק אם יש הזמנה שממתינה להחלטת AI או משלוח הממתין לשילוח.**

   ה. **ההזמנות מנוטרות לפי 2 גורמים – לכל הזמנה יש שדה זמן מדוייק (SpecificTime הכולל תאריך ושעה) לכל אחד מהשלבים שעברה – יצירה, הכנה, המתנה להחלטה, המתנה לשילוח, הגעה ליעד. מלבד זאת לכל הזמנה יש סטטוס Enum המתאר את הסטטוס העדכני שלה.**

   ו. **בתהליכי כלל הטבחים, השליחים וההזמנות ניתן לצפות על ידי כניסה למערכת CRM בממשק המנהל. הCrmLayout הוא חלון המסוננרן בזמן אמת על ידי תהליכון חיצוני SyncThread, כלומר תהליכון שמשנה מבחוץ את האובייקטים בתוך החלון באינטרוולים של שניה אחת. מהCRM ניתן לראות איזה שליח או טבח עמוס עכשיו וכמה זמן הוא כבר בסטטוס שלו, כמו כן ניתן לראות את סטטוס ההזמנה וכמה זמן עבר מאז שנוצרה.**

   ז. **הערה: תחת מתכונת זו, פעולות הוספה של משלוחים, הפעלת AI או סימון שילוח הם מיותרים לחלוטין. האופציות קיימות בממשק, אבל עדיף לתת למערכת לעבוד בלי התערבות מבחוץ.**

4. **תצורת שרת – לקוח.**

א. בחלון הראשון שנפתח קיימת בחירה בין 3 אופציות. הפעלת שרת בלבד, הפעלת לקוח בלבד (דורש הכנסת כתובת IP של שרת לשדה המיועד) והפעלה היברידית, המומלצת לצורך בדיקת התרגיל (אאב יש לכם כמה מחשבים פנויים עם תקשורת בינהם). במידה ובוחרים היברידי או לקוח בלבד נפתח חלון הLOGIN. אחרת יפתח חלון שאומר שהשרת רץ ברקע ומה הכתובת שלו.
*חשוב לשים לב: הפעלת מצב לקוח בלבד ללא הזנת שרת אליו יש להתחבר תביא לסגירת התוכנית. אין פעולות שניתן לבצע על לקוח ללא שרת.

ב. התקשורת עובדת על הפורטים 5656 עבור השרת הראשי ו5657 עבור שרת הCRM. בהפעלה ראשונית, על מנת שהתוכנה תוכל לעבוד, יש לאשר לחומת האש לפתוח את הפורטים (או לפתוח ידני אם איננכם משתמשים במערכת הפעלה של windows).

ג. הפרוטוקולים מבוססי טקסט, באופן שבו מצד הלקוח אין כלל שימוש באובייקטים פנימיים (מלבד ENUMS). כל פרוטוקול הוא מחרוזת יחידה הבנויה על פי הכללים הבאים:
הלקוח שולח לשרת מחרוזת מופרדת בתו /, כאשר הארגומנט הראשון במחרוזת הוא שם הפקודה שברצון הלקוח להפעיל. בצד השרת יש מילון עבור כל הפקודות האפשריות כאשר המפתח הוא הפקודה והערך הוא מתודת למבדה שאותה הפקודה מפעילה (בדומה למה שעיצבתם לנו עם הקובץ CSV). השרת מבצע את הפקודה וכותב ללקוח בחזרה מחרוזת תגובה.
מחרוזת התגובה עשויה להיות רשימה, סטטוס+ID או סיבת כישלון (עבור מתודות בוליאניות) או אובייקט מלא. הלקוח קורא את התגובה כטקסט, מפצל אותה לפי התו הרצוי (יש תו מיועד לכל סוג הפרדה וכמה חריגים) ומזין נתונים לממשק המשתמש כטקסט בלבד, ללא שום פיתוח לאובייקטים.

ד. עבור כל משתמש שמתחבר לשרת נפתח תהליכון צדדי המיועד לטיפול בפקודות אותו לקוח. התהליכון מת כאשר נקטע החיבור מול הלקוח.

ה. מחלקת הלקוח יושבת בסרגל הכלים ToolsBar. אני רואה את זה כניצול העובדה שרכיבי FX עובדים בעצמם כתהליכונים מקבילים ואם יש רכיב שאני יודע שיהיה זמין תמיד עבור הלקוח, אפשר גם להושיב עליו סוקט ומשתני קריאה וכתיבה במקום להפעיל תהליכון נפרד.

ו. ברקע פועל מנקה תהליכונים ConnectionCheckThread שבודק כל 4 שניות את סטטוס הפעילות של תהליכוני DealWith. תהליכון כזה מפסיק פעילות ברגע שנשבר החיבור ללקוח. במידה והתהליכון לא פעיל, המנקה יהרוג אותו ויסיר אותו מרשימת התהליכונים הפעילים.

5. בנוסף לשרת, הסרגל כלים מכיל בתוכו 2 מחסניות. מחסנית אחת לחלונות ומחסנית שניה לשמות החלונות. בכל מעבר של חלון לחלון חדש המחסנית דוחפת את החלון הקודם ואת השם הקודם בהתאמה. בעת לחיצה על כפתור "back" המסך טוען מחדש את החלונית העליונה במחסנית (pop) ושדה שם החלון המופיע בסוף הסרגל טוען את השם העליון במחסנית שלו.
בעת לחיצה על כפתור logout מופיע מחדש מסך הlogin הראשוני והמחסניות מאופסות. מלבד זאת מכיל הסרגל כפתור יציאה (אם מופעל שרת כפתור היציאה גם שומר מידע), מזעור ושמירת מידע.

6. יש צלילים מגניבים כאלה כשמתקבלת הודעת הצלחה או שגיאה. לחלון popup יש פרמטר האם הבשורות טובות או רעות ולפי זה נקבע הצליל. עד כאן דברים שהוספתי לבונוס יצירתיות כפיצוי על זה שיש לי חוש עיצובי גרוע.

7. חיפוש מרובה משתנים. האופן שבו מוצגות למנהל רשימות של אובייקטים הוא באמצעות toString שנשלחו אליו דרך השרת. כלומר – כל תצוגה של אובייקט למנהל מכילה את כל המשתנים שיש בו כמחרוזת. לדוגמא, כאשר משתמש צריך לקבל את כל

8. ניצלתי את זה כבר בהתחלה עוד לפני שראיתי שיש בונוס בנדון בכדי לייצר 2 אובייקטים משלי – FilteredBox שהוא בעצם ComboBox מסונן, FilteredList שהוא בעצם ListView מסונן. האובייקטים בנויים כך שלכל רשימה או תיבה מוצמד מלמעלה שדה טקסט. על השדה מופעל listener שבכל הקלדה ממיין את הרשימה לפי המחרוזות שמכילות את המחרוזת המוקלדת בשדה.

במסך הצגת הנתונים, אליו הבונוס מתייחס, כל נתוני האובייקט הם חלק מהמחרוזות המוצגות, ולכן ניתן לבצע את הסינון על פי כל אחד ואחד מהם. בנוסף, היות ושם הלקוח והאזור שלו מופיעים במידע על ההזמנה, ניתן לחפש הזמנה לפיהם ואין צורך בחלון נפרד עבור מבני הנתונים הזמנה לפי לקוח והזמנה לפי איזור.
הערה חשובה: נכון שמבחינת סיבוכיות היה עדיף לעשות מילון לכל פרמטר ולבדוק לפי פרמטרים. אבל יש את עניין הנוחות באי הצורך לבחור פרמטר, והעובדה שהחיפוש מתבצע ישירות על הטקסט הופכת את החיפוש לנוח יותר. זו החלטה שקיבלתי באופן מודע, להאט את הסיבוכיות עבור הנוחות.

9. סרטון רץ בלולאה בכניסה כלקוח.

מסתבר שCSS לא תומך בJAVAFX בטעינת קבצי GIF. הוא כן תומך בתמונות. אז יצרתי Util שתפקידו להריץ 21 תמונות מתחלפות בלולאה כשמתחברים ממשתמש. הUtil פועל כתהליכון נפרד, בדומה לתהליכון הסנכרון בCRM, ומשנה מבחוץ את הרקע של מסך המשתמש באינטרוולים של 0.8 שניות.

10. ממשק רספונסיבי. דבר שגיליתי שאני חייב לעשות ברגע האחרון כשהתחלתי להריץ על מחשבים שונים ולראות שהGUI יוצא איום ונורא (כשאובייקטים חוצים גבול של BORDERPANE הם גורמים לכפתורים ושדות טקסט מעבר לגבול להפסיק לתפקד).
הרספונסיביות מוגדרת על ידי לקיחת פרמטרים מהמסך בתחילת ההרצה, חלוקה שלהם בפרמטרים עליהם עבדתי בהתחלה ושמירת התוצה למשתנים סטטיים אליהם ניתן יהיה להגיע בקלות מכל המחלקות.
כל מתודות הsetStyle הועברו לString.format והפרמטרים שקיבלו קודם הוכפלו ביחס הרלוונטי לפני שהוכנסו למחרוזת. זו היתה עבודת נמלים של כמה שעות אבל כרגע הממשק רספונסיבי לחלוטין.

11. אינדוקטיביות. כל השדות בהם אני משתמש הם שדות יורשים מהאובייקטים המקוריים שלהם על פי הצורך שלי באותו שדה.
קיימים מספר סוגי שדות – שדות בחירה, כאשר מופעלת אופציית חיפוש אם יש הרבה משתנים, שמסומנים באדום כאשר הם ריקים. שדות בחירת זמן – מסומנים באדום עבור תאריך לידה אם התאריך היה לפני פחות מ18 שנה (המסעדה לא מעסיקה או מוכרת לאנשים מתחת ל18), ועבור שדות בחירת תאריך משלוח יסומנו באדום אם התאריך כבר עבר.
שדות טקסט – בודקים אם הטקסט בתוכם תקין. יש מספר סוגים של שדות טקסט:
   א. שדה טקסט רגיל: בודק שאין תווים העשויים להפריע לתקשורת עם השרת.
   ב. שדה מייל: בודק שהמייל כתוב בפורמט תקין.
   ג. שדה מספר רגיל: בודק שהשדה הוא מספר. השדה מוודא גם שהמספר חיובי, מפני שאין שדות מספר שיכולים לקבל מספר שלילי בתוכנה.
   ד. שדה שבר: כמו שדה מספר, אבל מאפשר נקודה באמצע.
   ה. שדה טלפון – יורש ממספר – בודק שמספר הטלפון תקין.
   ו. שדה ת"ז לאדם – יורש ממספר – בודק שמספר הת"ז תקין ע"פ ספרת ביקורת.
   ז. שדה סיסמא – נותן אידיקציה לגבי מורכבות הסיסמא (מחושב לסיביות) – עד 64 סיביות השדה יהיה אדום, בין 64 ל96 כתום, בין 96 ל128 צהוב, ומעל 128 יהיה ירוק. לא תתאפשר סיסמא עם פחות מ64 סיביות.
בכל מקרה ערך לא חוקי לשדה יקבל הודעת שגיאה, בין אם מבדיקה אינדוקטיבית ובין אם לאחר תשובה מהשרת. כדאי לציין שבכל אובייקט כזה קיימת מתודת checkStyle. הסיבה לקיום המתודה היא שלאחר ההגדרה אני עושה לכל אובייקט setStyle כדי לקבוע פרמטרים כמו מיקום, גודל וכו (CSS כבר אמרנו). בכל פעם שבה מריצים את המתודה הזו כלל הפרמטרים הישנים שהוגדרו בה בהגדרה הראשונית, לרבות צבע, נמחקים. ולכן צריך להפעיל טריגר שיקבע את הצבע מיד לאחר ביצוע הפעולה. מכיוון שאני עצלן ולא רציתי להתחיל לעבור על כל המתודות עם עוד תוספת, פשוט יצרתי מתודה checkStyle שעושה setStyle ניוצרת טריגר חדש לקביעת צבע.
לא הגדרתי אינדוקטיביות ברשימות. במידה ויש בכך צורך אכן תיזרק BlankFieldException, אבל הרשימה לא תצבע באדום.

חריגות:
קיימת היררכיית חריגות המחולקת ל3 קטגוריות:
א. InputException. חריגות אלו מתקבלות כאשר המנהל או הלקוח מנסה להזין ערכים לא חוקיים ברמת המערכת. היורשות מחריגה זו הן:
   1. BlankFieldException נזרקת כאשר שדה או רשימת קלט ריקים. חריגה זו אינדיקטיבית.
   2. FaildToAdd/RemoveException נזרקות כאשר מתודות ההוספה או ההסרה בRestaurant מחזירות תשובה שלילית מכל סיבה שהיא (לדוגמא אם מספר התז כבר קיים במערכת). חריגה זו אינה אינדיקטיבית.
   3. IllegalCharacterException נזרקת כאשר שדה טקסט מכיל תווים העשויים להפריע לפרוטוקול השרת, לדוגמא תווים בהם משתמש השרת להפרדה בין ארגומנטים. תווים אסורים: רווח, סלאש (רגיל והפוך)ת מקף, סימן קריאה, דולר, אחוז ופסיק. חריגה זו אינדיקטיבית.
   4. InvalidEmailException נזרקת כאשר פורמט המייל אינו username@domain או כאשר הדומיין לא לפי הפורמט name.regionFolder
   5. InvalidIDException נזרקת כאשר תעודת זהות אינה תקינה (נבדק לפי ספרת ביקורת). חריגה זו אינדיקטיבית.
   6. InvalidPhoneException נזרקת כאשר מספר הטלפון לא תקין. מספר טלפון בישראל יכול להכיל 9 או 10 ספרות כאשר 2 או 3 ראשונות הן קידומות (קבוע) והיתר הן ספרות. חריגה זו אינדיקטיבית.
   7. ToLateException נזרקת כאשר נוצרת הזמנה לתאריך שכבר עבר. היות ויש אוטומציה המנהל לא אמור להוסיף הזמנה, כך שאין סיבה שהיא בפועל תזרק. חריגה זו אינדיקטיבית.
   8. WrongAreaException נזרקת כאשר הלקוח, השליח או המשלוח לא משוייכים כולם לאותו האיזור. האיזור נקבע לפי השכונה בה גר הלקוח ואין אפשרות להוסיף או לערוך משלוח המכיל הזמנה עבורה תוך שיוך לאיזור משלוחים אחר.
   חריגה זו אינה אינדיקטיבית.

9. NotNumberException נזרקת כאשר בשדה בו אמור להיות מספר מוזנת מחרוזת שאינה מספר.

ב. LegalActionException. חריגות אלו נזרקות כאשר המנהל או הלקוח מנסים להזין ערכים חוקיים ברמת המערכת, אך מסיפור מנהליות נאסרו. היורשות מחריגה זו הן:
1. ConvertToExpressException נזרקת כאשר נוצרת משלוח רגיל עם הזמנה אחת. לא אינדיקטיבי.
2. EmptyBufferException נוצר כאשר מוחזרת מהשרת ללקוח רשימה ריקה. לא אינדיקטיבי.
3. IllegalCustomerException נזרקת כאשר לקוח מהרשימה השחורה מנסה לבצע הזמנה. לא אינדיקטיבי.
4. LieExeption נזרקת כאשר מנהל מנסה להזין ערכים שליליים בשדות שדורשים מספר, לדוגמא בזמני הכנה או בערכים תזונתיים לרכיבים. אינדיקטיבי.
5. LowPasswordException נזרקת כאשר הסיסמא חלשה מדי (פחות מ64 סיביות). אינדיקטיבי.
6. NoComponentsException נזרקת כאשר נמחק רכיב ממנה והיא נשארת כתוצאה מכך ללא רכיבים. לא אינדיקטיבי.
7. SensitiveException נזרקת כאשר מוזמנת הזמנה עבור לקוח שאלרגי לאחד הרכיבים בהזמנה. לא אינדיקטיבי.
8. ToYoungException נזרקת כאשר קטין מתחת לגיל 18 מנסה לבצע הזמנה או להתקבל לעבודה כטבח או שליח. אינדיקטיבי.

ג. InternalErrorException. חריגות אלו נזרקות כתוצאה מתקלה מערכתית. היורשות מחריגה זו הן:
1. CommandNotFoundException נזרקת כאשר קליינט שולח לשרת פקודה שלא נמצאת במילון הפקודות. במצב זה תוצב הודעה שגיאה לפיה פקודה ככה וככה לא נמצאה.
2. ReaderConnectionException נזרקת כאשר קליינט או שרת לא מצליחים לקבל הודעה שנשלחת אליהם.
3. במחלקה Restaurant נעשה שימוש בIOException כאשר בעלייה של השרת הנתונים נשמרו בפורמט לא תקין או כאשר אין קובץ Rest.ser והמחשב לא מצליח לייצר כזה.
4. DefaultException. מסיבה כלשהי JavaFX זורק שגיאות שאין לתהליכונים מסוג Thread תמיכה בו. השגיאות הללו לא מייצגות את המתרחש בתוכנה ובפועל התהליכונים פועלים ללא שום תקלה. כדי שלא יזרקו ברגע שגיאות מיותרות שיעצרו את פעולת התוכנה, לתהליכונים מקבילים הוגדרה מערכת זריקת שגיאות דיפולטיבית שמבצעת כלום.

כלים חדשים בUTILS:

ClientReader extends BufferedReader הוא אובייקט לקריאת נתונים שבודק אם הערכים שמגיעים אליו תקינים וזורק הודעת שגיאה בהתאם לפני שהנתונים מוזרמים לתוכנית.

Encryption הוא כלי להצפנה ופיצוח הצפנה עבור סיסמאות.

Gender קיבל פורמט מעודכן ועכשיו יש הכרה במסעדה גם במגדרים נוספים מלבד גברים ונשים.

OrderStatus הוא ENUM המכיל את כל המצבים בהם הזמנה יכולה להיות.

RunGif extends Thread הוא התהליכון שמריץ את התמונות ברקע של מסך הלקוח.

ServerAction הוא ממשק שמאפשר לשמור לתוכו מתודות שמקבלות ארגומנטים (עבור תהליכוני DealWith)

SpecificTime הוא אובייקט שמשלב תאריך ושעה.

TimeDistance הוא אובייקט שמכיל הפרש בין SpecificTime 2.

בשכבת MODEL נוספו Account המכיל שם משתמש וסיסמא לצורך התחברות וWorker המכיל את המצב שבו עובד (טבח או איש משלוחים) נמצא.

בשכבת CRM מופיעים כל תהליכוני הCRM מלבד התצוגה שנמצאת לבקשתכם בView.

בשכבת application נמצאים תהליכוני השרת והאפליקציה.

בשכבת View נמצאים מסכי המנהל, מסך הלקוח, מסך ההתחברות ובחירת סוג הפעלה ומסך הפופאפ.

בשכבה View.Nodes נמצאים אובייקטי עזר גארפיים שבניתי על מנת להשיג את כל הפונקציות שהצגתי למעלה.

עד כאן דף ההסבר, מקווה שלא חפרתי 😊