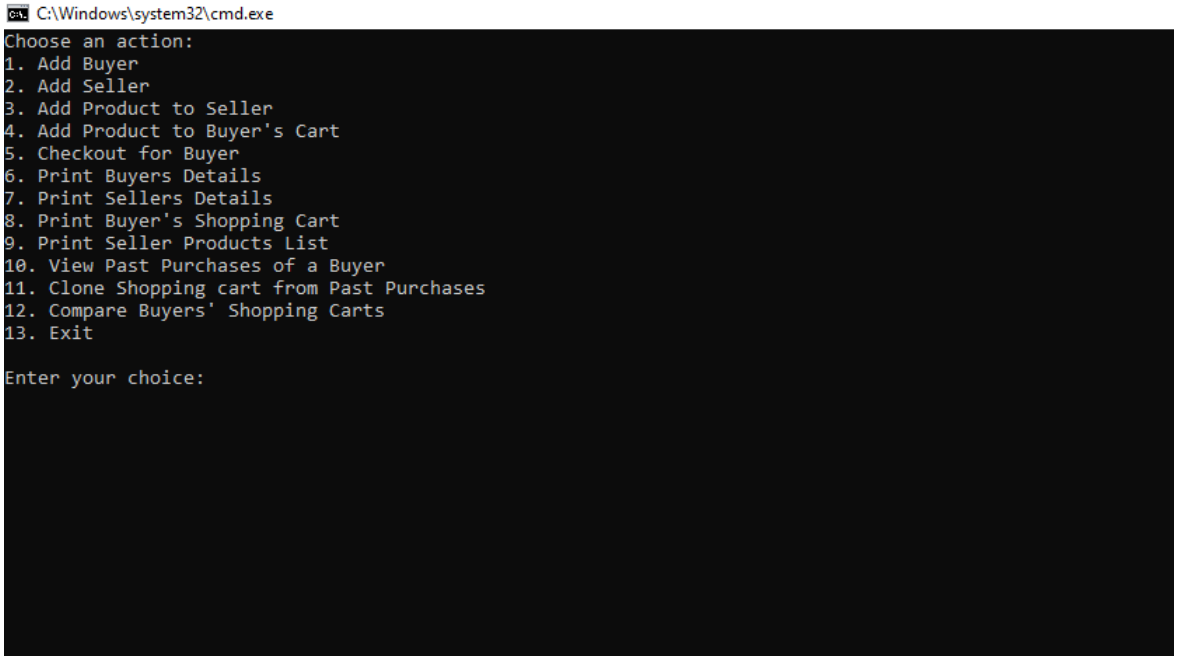
הסבר על החלק השלישי בעבודה:

הסבר על התפריט הראשי:



אופציה מספר 1: הוספת buyer לחנות

אופציה מספר 2: הוספת seller לחנות

אופציה מספר 3: הוספת מוצר חדש ל- seller.

אופציה מספר 4: הוספת מוצר לעגלת הקניות הנוכחית של buyer.

אופציה מספר 5: קנייה של עגלת הקניות הנוכחית של buyer.

אופציה מספר 6: הדפסת הפרטים של כל ה-buyers שבחנות.

אופציה מספר 7: הדפסת הפרטים של כל ה-sellers שבחנות

אופציה מספר 8: הדפסת עגלת הקניות הנוכחית של buyer מסוים.

אופציה מספר 9: הדפסת רשימת המוצרים ש-seller מסוים מוכר.

אופציה מספר 10: צפייה בהיסטוריית ההזמנות של buyer מסוים.

אופציה מספר 11: שכפול עגלת קניות מעגלה שכבר קיימת.

אופציה מספר 12: השוואה בין 2 עגלות קניות של 2 buyers מסוימים.

דרישה מספר 1:

בחלק זה המרנו את השימוש במערכים ב- List<> לכן כעת בכל מקום שהופיע מערך בעבר מופיעה כעת list.

לדוגמה, כעת ל- buyer יש את התכונות הבאות:

```
private List<Product> shopping_cart;
private List<Order> past_purchases;
```

רשימה של products שמייצגת עגלת קניות של ה-buyer.
רשימה של orders שמייצגת את היסטוריית ההזמנות של אותו ה-buyer. זאת רק דוגמה אחת, לשימוש ב-List. גם בחלקים נוספים בפרויקט המרנו את השימוש ב-arrays בשימוש ב- List<>.

דרישה מספר 2:

כעת המערכת שלנו תומכת בכך שהקלט לא יהיה מהטיפוס המבוקש. בכל מקרה כזה הצגנו הודעה מתאימה ללקוח.

עשינו זאת בעזרת שימוש ב- exceptions שמובנים בשפת c# ותפסנו אותם ב- mainProgram. דוגמאות ל- exceptions בהם השתמשנו הן: ArgumentNullException, ArgumentOutOfRangeException ועוד..

כעת המערכת שלנו תומכת ויודעת להתמודד עם מקרים בהם קלט שהוזן אינו תקין, ולהציג הודעה מתאימה למשתמש.

- אנחנו לא מאפשרים ששם של buyer/seller יכיל מספרים.
- אנחנו לא מאפשרים ששם של רחוב יכיל מספרים
- אנחנו לא מאפשרים שמספר בניין יהיה שלילי או שהוא יכיל אותיות (הרי אין בזה היגיון...)
- בנוסף אנחנו לא מאפשרים ששמות של עיר, רחוב יכילו תווים מיוחדים כמו @ או #.

דרישה מספר 3:

יצרנו `exception` חדשה משלנו שנקראת SingleItemOrderException – מטרת ה-`exception` הזאת היא לזרוק שגיאה במקרה שלקוח מבצע הזמנה שכוללת פריט אחד בלבד. את השגיאה הזו תפסנו ב- `mainProgram` והצגנו הודעה מתאימה ללקוח.

דרישה מספר 4:

מימשנו במחלקה EcommerceStore את האופרטור + (מימשנו 2 גרסאות שלו – אחת שמוסיפה buyer לחנות וגרסה שנייה שמוסיפה seller לחנות)

דרישה מספר 5:

במחלקה buyer מימשנו את האופרטור > שמבצע השוואה בין 2 buyers. הקריטריון של ההשוואה הוא CalculateTotalPrice() שזאת פונקציה שמחשבת את הסכום הכולל של עגלת הקניות. כלומר, כעת בלחיצה על אופציה מספר 12 נוכל להשוואת בין 2 עגלות קניות של buyers ולראות לאיזה buyer יש עגלה עם מחיר גבוה יותר (הערה: צריך לשים לב, שהשוואה צריכה להיעשות לפני שמפעילים את הפונקציה checkout בתפריט, כי פונקציה זו קונה את העגלה ומנקה אותה, לכן חשוב לבצע השוואה לפני שקונים את העגלה)

דרישה מספר 6:

כעת המחלה order היא ICloneable כלומר ניתן לשכפל הזמנה. מימשנו את האופציה הזו ע"י פונקציית ה-clone. כעת בלחיצה על אופציה מספר 11, נוכל לשכפל עגלה שכבר קיימת בהיסטוריית הרכישה של אותו buyer. נשים לב שבעת הלחיצה על אופציה מספר 11 נצטרך להזין order_id כלומר מספר הזמנה אותה אנו רוצים לשכפל עבור אותו buyer. לאחר שנזין ערך תקין, ההזמנה תשוכפל ותתווסף אוטומטית לרשימת ההזמנות של אותו buyer.

דרישה מספר 7:

כעת גם המחלקה seller היא IComparable<Seller>. את ההשוואה בין ה- sellers עשינו לפי מספר המוצרים שהם מוכרים (בפונקציית compareTo במחלקה seller). כעת גם מימשנו את זה שהדפסת המוכרים תהיה ממוינת לפי מספר הפריטים שהם מוכרים (זה התאפשר לנו בגלל ש- IComparable<Seller>). את זה מימשנו בפונקציה PrintSellersArrayDetails שנמצאת במחלקה EcommerceStore. בפונקציה זו מיינו את מערך ה-sellers ב- OrderByDescending ואז הדפסנו אותו.

וזהו 😊