רידמי תרגיל 3:

1. **שאלה 1 –** התפקידים והקשרים של המחלקות בקוד:
2. מחלקת Shell –
   1. תפקיד - מחלקה זו אחראית לממשק משתמש בצורת שורת פקודה, עבור אלגוריתם AsciiArt. היא מקבלת פקודות מהמשתמש, מוודאת את תקינותן ומפעילה את האלגוריתם בהתאם לנדרש.
   2. קשרים:
      * המחלקה משתמשת במופעים של מחלקת AsciiArtAlgorithm. בהינתן קבלת פקודה להרצת האלגוריתם, Shell יוצרת מופע של האלגוריתם, עם פרמטרים של תמונה, רזולוציה וסט תווים לריצה.
      * המחלקה משתמשת במופעים של מחלקות שמממשות את הממשק של AsciiOutput. כאשר Shell מקבלת פקודה להרצת האלגוריתם, היא משתמשת באובייקט שמממש את AsciiOutputכדי להוציא לפלט את תוצר האלגוריתם.
      * המחלקה משתמשת במתודות של המחלקה ImageUtils, על מנת לבצע ריפוד לתמונה שהמשתמש מבקש להשתמש בה.
      * המחלקה משתמשת בכל סוגי המחלקות שיורשות מהמחלקה IllegalCommandException, על מנת לטפל בשגיאות מתאימות שנובעות מקלט של המשתמש (פירוט על המחלקות בהמשך).
      * המחלקה משתמשת במחלקת KeyboardInput על מנת לקבל קלט מהמשתמש.
      * המחלקה מכילה מופע של מחלקת Image, על מנת להעביר אותה לאלגוריתם.
3. מחלקת AsciiArtAlgorithm –
   1. תפקיד – המחלקה אחראית למימוש אלגוריתם ה-AsciiArt.
   2. קשרים:
      * המחלקה משתמש בתודות של מחלקת ImageUtils על מנת לחלק את התמונה לתתי תמונות, ולחשב את בהירות כל תת תמונה.
      * המחלקה משתמשת במחלקת SubImageCharMatcher, על מנת להתאים כל תת תמונה לתו אסקי מתאים בסט התווים שקיבלה.
      * המחלקה מכילה מופע של מחלקת Image, עליו היא מבצעת את העיבוד המקדים ואת פעולת האלגוריתם.
4. מחלקת SubImageCharMatcher –
   1. תפקיד – המחלקה אחראית להתאים תו ascii לתת תמונה עם בהירות נתונה. מחלקה זו משתמשת את אלגוריתם Ascii Art', כדי להמיר תת תמונות בתווים.
   2. קשרים:
      * המחלקה משתמשת במחלקת CharConverter על מנת פירוק של תו ascii למערך בולאני, בו משתמשת על מנת לקבל את בהירות התו.
5. מחלקת ImageUtils –
   1. תפקיד – המחלקה מכילה מתודות שמשמשות לעיבוד תמונה, בדגש על ריפוד, חלוקה לתת תמונות ונרמול גוונים לאפור.
   2. קשרים:
      * המחלקה משתמשת במופעים של מחלקת Image, שעליהם נעשות הפעולות שבמחלקה.
6. מחלקת IllegalCommandException -
   1. תפקיד – המחלקה משמשת ליצירת exceptions עבור קלט לא חוקי של המשתמש במחלקת Shell.
   2. קשרים:
      * המחלקה יורשת ממחלקת Exception.
7. מחלקת InvalidResolutionBoundaryException –
   1. המחלקה משמשת לתיאור exceptions במצבים בהם המשתמש מבקש לשנות רזולוציה מעבר לגדלים המותרים.
   2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
8. מחלקת InvalidImageException –
   1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם קובץ התמונה שנשלח לתוכנית אינו מוביל לתמונה תקינה.
   2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
9. מחלקת IllegalCommandNameException –
   1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם המשתמש הכניס Shell פקודה לא תקינה.
   2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
10. מחלקת IllegalRunAlgoritrhmException –
    1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם ריצת האלגוריתם אינה חוקית, במקרה בו סט התווים ריק.
    2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
11. מחלקת IllegalRemoveException –
    1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם מתקבלים פרמטרים לא חוקיים עבור פקודת remove.
    2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
12. מחלקת IllegalAddException –
    1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם מתקבלים פרמטרים לא חוקיים עבור פקודת add.
    2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
13. מחלקת IllegalOutputException –
    1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם מתקבלים פרמטרים לא חוקיים עבור פקודת output.
    2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
14. מחלקת IllegalResolutionException –
    1. תפקיד – המחלקה משמשת לתיאור exceptions במצבים בהם מתקבלים פרמטרים לא חוקיים עבור פקודת res.
    2. קשרים – המחלקה יורשת ממחלקת IllegalCommandException.
15. שאלה 2 – מבני נתונים של java בהם השתמשנו בקוד:
16. במחלקת Shell – שימוש בTreeSet<Character> - השתמשנו בעץ זה כדי לשמור בShell את התווים שהמשתמש מעוניין להעביר לאלגוריתם. כך, בהינתן הסרה או הוספה של תווים הפעולה מתבצעת על מבנה נתונים זה, ובהינתן פקודת הרצה של האלגוריתם מועבר לאלגוריתם עותק תקין של קבוצת התווים.

בנוסף, מבנה נתונים זה משמש להצגת התווים כאשר מתקבלת פקודת chars. שיקולי הבחירה במבנה נתונים זה:

* + - מבנה הנתונים מונע כפילויות של תווים.
    - מבנה הנתונים מאפשר הצגה ממוינת של התווים לפי ערך הascii שלהם, מכיוון שהוא ממומש על ידי עץ אדום-שחור. כך, בקבלת פקודה להדפסת התווים הפעולה תתבצע בO(n), מעבר על כל התווים, ולא נצטרך למיין אותם.
    - הסרה והוספה של ערכים מבצעת בזמן של log(n). אמנם קיימים מבני נתונים שיכולים לבצע פעולות אלו בצורה יעילה יותר, אך הדרישה למיון של התווים כשנדרשת הצגתם יעילה באופן יחסי בהשוואה למבני נתונים אלו. על כן בחרנו במבנה זה.

1. במחלקת SubImageCharMatcher – שימוש ב- TreeMap<double, PriorityQueue<Character> - השתמשנו בעץ זה על מנת להתאים בהירות נתונה לתו בצורה יעילה. התאמה זו מתבצעת ע"י מציאת בהירות התו הקרובה ביותר אל בהירות הנתונה, לכן בחרנו להשתמש במבנה נתונים מסוג : TreeMap שמאפשר מציאת מפתחות קרובים ביותר בערכם לערך נתון בO(log(n)).

בנוסף השתמשנו בvalue ב PriorityQueue של תווים, על מנת לשמור ביחד תווים בעלי ערך בהירות זהה. בחרנו במבנה נתונים זה כדי לשלוף התו בעל ערך הascii המינימלי (בהתאם לדרישות התרגיל), בהינתן בהירות נתונה, ב-O(1) מתוך התור.

*בשורה התחתונה, מבנה נתונים זה מאפשר לשלוף מפתח שקרוב לערך נתון בצורה די יעילה (O(log(n)) ואחר מכן לקבל את התו המינימלי שמתאים למפתח בO(1), שכן הוא נמצא בראש התור.*

1. *במחלקת* SubImageCharMatcher *– שימוש בטבלת גיבוב סטטית: HashMap<Character, double> - כאשר מופע של המחלקה מחשב בהירות של תו נתון, הוא מכניס את הערכים לטבלה זו. כך, כאשר מופעים נוספים מעוניינים לקבל את הבהירות של התו פעם נוספת, הם אינם נדרשים לחישוב שלה פעם נוספת ומסוגלים לשלוף אותו מהטבלה בזמן קבוע בקירוב. מכיוון שהשימוש העיקרי של מבנה הנתונים הוא מציאה והוספה של ערכים, בחרנו בטבלת גיבוב שבה פעולות אלו מתבצעות ב-O(1) בקירוב.*
2. *במחלקת* SubImageCharMatcher *– שימוש בקבוצה: HashSet<character> - טבלת גיבוב זו משמשת מופע של המחלקה בכדי לשמור על קבוצת התווים שקיבלה מהאלגוריתם. בחרנו בטבלת גיבוב על מנת למנוע כפילויות, וכדי שפעולות כמו הוספה והסרה של תווים מהקבוצה יבוצעו ב-O(1) בקירוב.*
3. *על מנת לייצר ולטפל בחריגות שנובעות מפעולות לא חוקיות של המשתמש מול מחלקת Shell, תחילה יצרנו מחלקת אב שנקרית IllegalommandException שיורשת ממחלקת Exception, וממנה יורשות מחלקות שמייצגות חריגות ספצפיות (ראו פירוט למחלקות בתשובה לשאלה 1). כך, כל טיפול בפקודה מסוימת זורק חריגה מתאימה עבורו בלבד, והחריגה נתפסת בפונקציית run כחריגת IllegalCommandException, ומטופלת בהתאם לדרישות. במקביל, לכל מחלקת חריגה הודעה משלה, והשימוש במחלקת האב במתודת run מאפשר פולימורפיזם וביטוי לכל חריגה. חשוב לציין כי במקרה בו מתקבלת בShell חריגת IOEXCEPTION בעקבות שגיאה בפתיחת קובץ תמונה, החריגה נתפסת בפונקציה שיוצרת את התמונה, והטיפול בה הוא זריקה של שגיאת InvalidImageException שתיתפס בפונקציית run.*