**Java - תרגיל 1 classes**

1. כתוב class בשם בTime המכילה את השדות הבאים: hours, minutes, secs ,כמו כן מגדירה ctors שונים לאתחול שדות האובייקט.

יש להגדיר לכל שדה מטודות get/set ו-מטודה toString () ובמחלקת התוכנית, בתוך ה-main יש להגדיר 2 אובייקטים ולהפעיל עליהם את המטודות השונות במחלקה.

1. כתוב את class Person המכילה את השדות הבאים: name, address, id, age ,כמו כן מגדירה ctors שונים לאתחול שדות האובייקט.

יש להגדיר לכל שדה באופן מתאים מטודות get/sets ו-מטודה toString () ובמחלקת התוכנית, בתוך ה-main יש להגדיר **מערך המכיל 3 אוביקטים** מסוג Person, להתחל כל מקום במערך עם אובייקט ולהפעיל עליהם את המטודות השונות.

1. כתוב את class Point המכילה את השדות הבאים: x ,y ‘מגדירה ctors שונים לאתחול שדות האובייקט.

יש להגדיר לכל שדה מטודות get/sets מתאימים ו-מטודה toString () , וt, את המטודות הבאות אשר מאפשרות חישוב מרחק בין 2 נקודות:

distance() - חישוב מרחק לנקודת 0,0

distance(int x, int y) - חישוב מרחק לנקודת x , y

distance(Point p) - חישוב מרחק לנקודת p כלומר לחשב את המרחק ל- p.x , p.y

ובמחלקת התוכנית, בתוך ה-main יש להגדיר 3 אובייקטים ולהפעיל עליהם את האופציות השונות במחלקה.

1. כתוב,את class Rectangle המהווה מלבן , ומכילה את השדות הבאים: x, y , width, height כמו כן מגדירה ctors שונים לאתחול שדות האובייקט.

יש להגדיר לכל שדה get/sets בהאתם לנכון מטודה toString() ואת הפונקציות הבאות:

int area() – מחזירה את שטח המלבן

int perimeter() – מחזירה את היקף המלבן

void resize(int percent) – מבצעת zooming למלבן באחוזים

void resize ( int w, int h) – משנה את גודל המלבן לפני רוחב וגובה חדשים

bool isEqual (Rectangle r ) – מחזירה האם שני המלבנים זהים מבחינת שטח

move ( int x, int y) void–משנה את קואורדינאטות המלבן

move ( Point p) void- משנה את קואורדינאטות המלבן לפי הנקודה המתקבלת

union( Rectangle r) Rectangle – מחזירה מלבן חדש, שהוא המלבן התוחם בין ה-this והמלבן המתקבל.

iIntersect( Rectangle r) bool – מחזירה אמת או שקר במידה ויש נקודות משותפות למלבנים

במחלקת התוכנית, בתוך ה-main יש להגדיר 2 מלבנים ולהפעיל עליהם את כל הפעולות השונות במחלקה.

1. יש לבצע Refactoring ל- class Rectangle , כך שבמקום הקואורדינאטות x , y נעשה שימוש בclass Point שבסעיף 3 . יש לשנות את כל המטודות בהתאם, ואת ה-main .

1. כתוב class Polygon המכילה מערך של אובייקטים מסוג Point , את גודל המערך נקבל ב-ctor , ובו האוביקט יותחל עם נתונים המתקבלים מהמקלדת בmain בזמן יצירת האובייקט..

יש להגדיר לכל שדה מטודות get/sets ו-מטודה toString () , פונקציה length () המחזירה את אורך הפוליגון ( יש להשתמש במטודות ממחלקה (Point, ומטודה המחזירה את כמות הנקודות בפוליגון.

במחלקת התוכנית, בתוך ה-main יש להגדיר אובייקט מסוג ה-class ולהפעיל עליו את המטודות השונות..

1. התרגיל הבא מורכב משני חלקים:

**חלק א'**

אפשר להסתכל על תמונה כ-grid של פיקסלים, לכן נגדיר class הנקרא RGBColor, שיכיל את השדות הבאים המתארים פיקסל במסך:

1. private int red,green, blue
2. כמו כן ctors:

RGBColor()

RGBColor(int r, int g, int b)

RGBColor(RGBColor other)

1. יש להוסיף מטודות get ו- set עבור כל שדה.
2. והמטודות :
   * String toString ()
   * boolean equals(RGBColor other)
   * void mix( RGBColor other) – מטודה המקבלת אובייקט מסוג RGBColor בשם other, ומשנה את ערכי השלישיה עליה הופעלה המטודה, להיות הצבע המתקבל מערבוב שני הצבעים. ערבוב הצבעים נעשה על ידי ממוצע של כל אחד מהמרכיבים. שימו לב שכאשר הממוצע הוא שבר, (למשל 112.5), יש לקחת את החלק השלם של הערך החדש (112).
   * double convertToGrayscale() - מטודה המחזירה את הערך האפור של שלישיית הצבע. הערל האפור נקבע כ- 30% מצבע האדום+ 59% מהצבע הירוק + 11% מהצבע הכחול.
   * void invert() - שיטה שמשנה את הצבע של השלישיה, על ידיד החלפה של כל אחד מערכי הרכיבים במשלים שלו ל- 255. ( למשל RGB של (0,1,2) יוחלפו ל- ) 255,254,253)

**חלק ב'**

בשאלה זו נייצג תמונה צבעונית בעזרת מערך דו-ממדי של אובייקטים מהמחלקה RGBColor. כל נקודה בתמונה היא פיקסל (pixel) המייצג את הצבע בקואורדינטה בודדת.

הייצוג נעשה בדרך המקובלת: בתמונה שיש בה n שורות ו- m עמודות,

השורות ממוספרות 0…n-1 מלמעלה למטה והעמודות ממוספרות 0…m-1 משמאל לימין.

עליכם לממש class RGBImage לפי הסעיפים להלן.

1. הגדרת התכונות של הclass כמערך דו מימדי של אוביקטים מהמחלקה הקודמת/
2. שלושה ctors כדלקמן:

* ctor היוצר תמונה שחורה חדשה בגודל לפי מספר השורות והעמודות

שהתקבלו כפרמטרים. אפשר להניח שמספרי השורות והעמודות חיוביים

( public RGBImage(int rows, int col

* ctor היוצר תמונה חדשה שזהה למערך של הפיקסלים שניתן לו כפרמטר.

אפשר להניח שהמערך pixels אינו null ושמספרי השורות והעמודות בו

חיוביים public RGBImage(RGBColor[][] pixels)

* ctor העתקה המקבל תמונה אחרת, ומעתיק את ערכיה. אפשר להניח שהפרמטר other שונה מ- null , ( public RGBImage(RGBImage other

כמו כן, יכיל את המטודות הבאות:

* public int getHeight ( ) – שיטה המחזירה את הגובה של התמונה בפיקסלים.
* public int getWidth ( )– שיטה המחזירה את הרוחב של התמונה בפיקסלים.
* מטודה המקבלת קואורדינטות public RGBColor getPixel (int row, int col)

בתמונה, ומחזירה את הפיקסל שנמצא בקואורדינטות אלו. אם הקואורדינטות מחוץ

לתמונה, יוחזר פיקסל שחור.

* public void setPixel (int row, int col, RGBColor pixel) - המטודה מקבלת קואורדינטות בתמונה ושלשת צבע (פיקסל), וקובעת פיקסל זה להיות בקואורדינטות

שהתקבלו כפרמטרים. אם הקואורדינטות הן מחוץ לגודל התמונה, לא עושים כלום.

* public boolean equals (RGBImage other) - מטודה המקבלת אוביקט נוסף מאותו טיפוס, ומחזירה האם התמונה שעליה הופעלה השיטה והתמונה שהתקבלה כפרמטר זהות.
* public void flipHorizontal ( ) – מטודה ההופכת את התמונה עליה הופעלה המטודה

סביב הציר האנכי . העמודה הראשונה הופכת להיות העמודה האחרונה, השניה

הופכת להיות השניה מהסוף וכד'.

* public void flipVertical ( ) – מטודה ההופכת את התמונה עליה הופעלה המטודה

סביב הציר האופקי . השורה הראשונה הופכת להיות השורה האחרונה, השניה

הופכת להיות השניה מהסוף וכד'.

* public void invertColors( ) – מטודה ההופכת את הצבעים של כל הפיקסלים

בתמונה על ידי החלפת כל צבע RGB במשלים לו ל- 255. לדוגמא: ערכי ה- RGB

(255,254,253) -ל יוחלפו (0,1,2) של

* public void rotateClockwise ( )– מטודה המסובבת את התמונה ב- 90 מעלות עם

כיוון השעון. שימו לב שהסיבוב יכול לשנות את מימדי התמונה.

לדוגמא, אם התמונה היא

( 0,0,0) (0,0,0) (0,0,0) (0,0,0)

( 1,1,1) (1,1,1) (1,1,1) (1,1,1)

( 2,2,2) (2,2,2) (2,2,2) (2,2,2)

אז לאחר הפעלת השיטה היא תהיה:

(2,2,2) (1,1,1) (0,0,0)

(2,2,2) (1,1,1) (0,0,0)

(2,2,2) (1,1,1) (0,0,0)

(2,2,2) (1,1,1) (0,0,0) 7

* public void rotateCounterClockwise ( )– מטודה המסובבת את התמונה ב- 90

מעלות נגד כיוון השעון. שימו לב שהסיבוב יכול לשנות את מימדי התמונה.

לדוגמא, אם התמונה היא

( 0,0,0) (0,0,0) (0,0,0) (0,0,0)

( 1,1,1) (1,1,1) (1,1,1) (1,1,1)

( 2,2,2) (2,2,2) (2,2,2) (2,2,2)

אז לאחר הפעלת המטודה היא תהיה:

( 0,0,0) (1,1,1) (2,2,2)

( 0,0,0) (1,1,1) (2,2,2)

( 0,0,0) (1,1,1) (2,2,2)

( 0,0,0) (1,1,1) (2,2,2)

* public void shiftCol (int offset) - מטודה המקבלת מספר שלם offset ,

ומזיזה את התמונה ימינה או שמאלה לפי הפרמטר שניתן. העמודה 0 עוברת להיות עמודה offset, עמודה 1 עוברת להיות עמודה offset+1 וכן הלאה. הפרמטר offset יכול להיות גם שלילי (או 0). העמודות שהוכנסו לעמודות שהוזזו צריכות להיות שחורות כולן.

אם ה- offset גדול ממספר העמודות, לא ייעשה כלום. אם ה- offset שווה למספר

העמודות, כל התמונה תהפוך לשחורה.

* public void shiftRow (int offset) - מטודה המקבלת מספר שלם offset , ומזיזה את התמונה למעלה או למטה לפי הפרמטר שניתן. השורה 0 עוברת להיות שורה offset,

שורה 1 עוברת להיות שורה offset+1 וכן הלאה. הפרמטר offset יכול להיות גם

שלילי (או 0). השורות שהוכנסו לשורות שהוזזו צריכות להיות שחורות כולן. אם ה-

offset גדול ממספר השורות, לא ייעשה כלום. אם ה- offset שווה למספר השורות, כל

התמונה תהפוך לשחורה.

* public double[][] toGrayscaleArray ( ) – מטודה המחזירה ייצוג אפור של התמונה.

הייצוג האפור של כל פיקסל מחושב כפי שהוגדר ב- API של RGBColor.

* public String toString()– שיטה המחזירה מחרוזת תווים המייצגת את התמונה.

המחרוזת צריכה להיות בדיוק בפורמט הבא:

כל שורה במערך נמצאת בשורה נפרדת כשבין הפיקסלים קיים רווח בודד. בסוף

שורה אין רווח.

כל פיקסל מוצג במחרוזת תווים בצורת שלשה של מספרים שלמים מופרדים

בפסיקים בתוך הסוגריים עגולים.

לדוגמה: ( 0,3,3) (1,2,3) (1,1,2) (1,0,1)

(1,3,4) (2,2,4) (2,1,3) (2,0,2)

( 2,3,5) (2,2,4) (2,1,3) (2,0,2)

* public RGBColor[][] toRGBColorArray()– מטודה המחזירה עותק של המערך של

הפיקסלים.