תרגיל בית מס. 1 – Point2D

הנה תרגיל הבית הראשון בו נתכנת ב-Java. בתרגיל זה ננסה לממש מנשק נתון בשתי דרכים שונות. באתר נמצא יישום המשתמש במנשק: נבדוק שניתן להחליף בין המימושים מבלי לשנות את היישום. כמו כן, נוודא שההתנהגות של שתי המחלקות שניצור זהה, למרות ההבדל באלגוריתמים.

הפתרון, אם כן, כולל שתי מחלקות, שיש להגיש בצורת שני קובצי קוד (עם סיומת .java) הארוזים יחד בקובץ ZIP, אותו יש להעלות ל-moodle. שמו של קובץ זה חייב להיות בדיוק כדלהלן:

7_Ex1_*xxxxxxxxx*_*yyyyyyyyy*.zip

כאשר *xxxxxxxxx* ו-*yyyyyyyyy* הם מספרי הזהות (בני 9 ספרות כל אחד, גם אם מתחילים ב-0) של שני המגישים. אנא, הגישו בזוגות בלבד. ההגשה עד יום ראשון, 2 במאי.

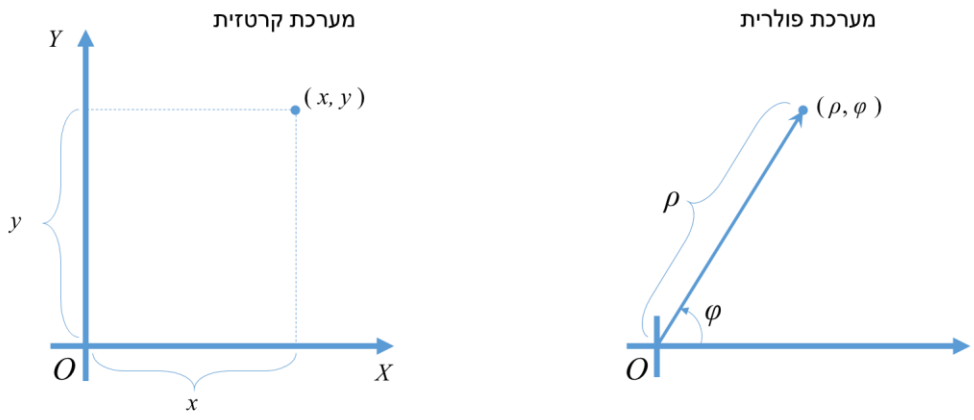
יש באתר קובץ Main.java שכולל מתודת ()main שבאמצעותה אפשר לוודא שהקוד שכתבתם עובד. אנא בדקו את הגשתכם לפני המשלוח, אך אל תכללו את קובץ הבדיקה או את הפלט ב-ZIP.

תיאור התרגיל:

המחלקות שניצור מייצגות נקודה במישור. המנשק, שעל שתי המחלקות לממש, נקרא Point2D (נקודה דו ממדית). הקובץ Point2D.java נמצא באתר.

מחלקה אחת נקראת Cartesian, והיא מייצגת את הנקודה בעזרת הקואורדינטות הקרטזיות שלה, x ו-y. בהתאם, השדות של מחלקה זו יהיו שני משתנים מטיפוס double ששמותיהם x ו-y, בהתאמה.

המחלקה השניה נקראת Polar, והיא מייצגת את הנקודה באמצעות קואורדינטות פולריות, הרדיוס-ווקטור ρ (נקרא רוֹ rho) והפאזה φ (נקרא פִי phi). הציור הבא מתאר כיצד גדלים אלה מגדירים נקודה במישור:

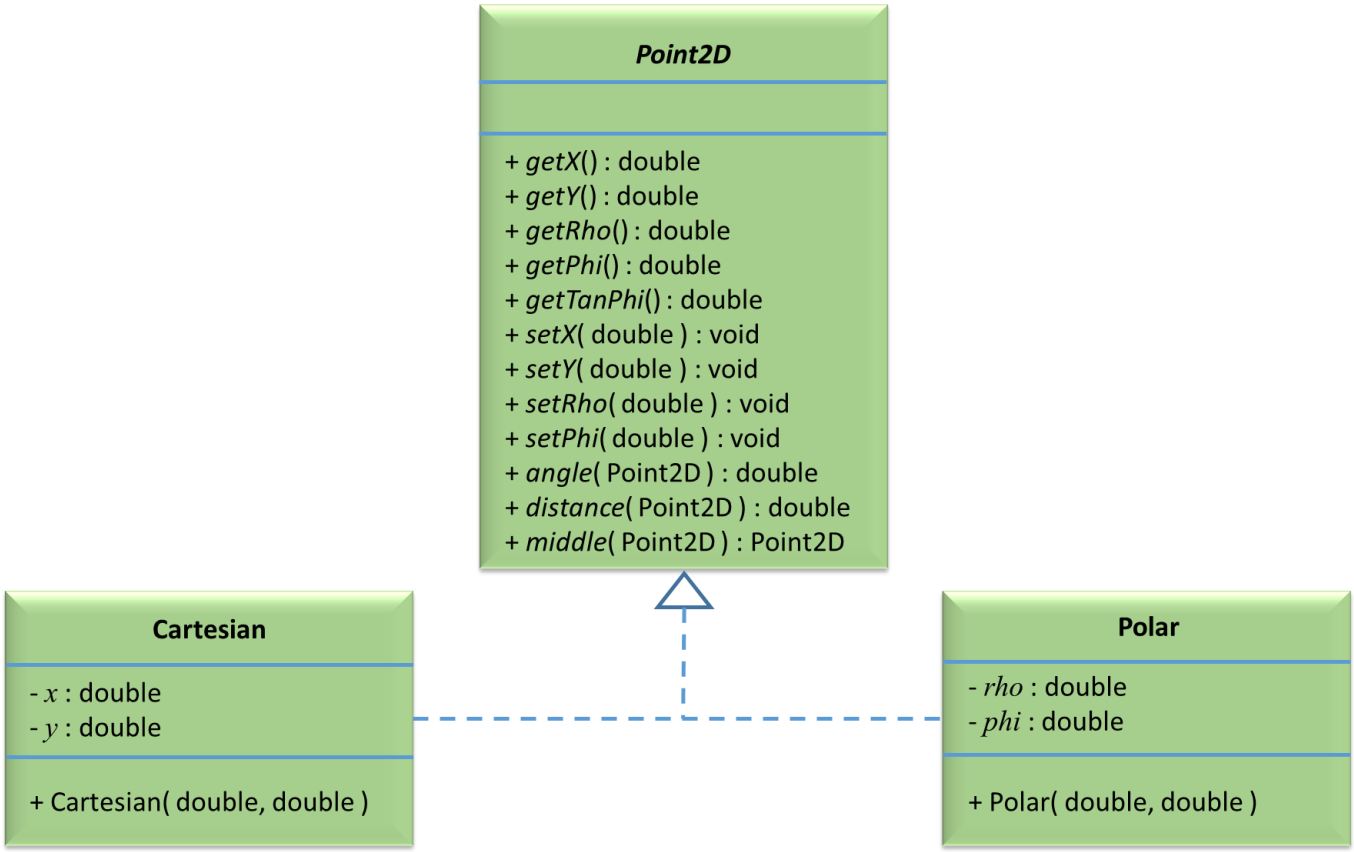


כלפי חוץ, שתי המחלקות מציגות את אותו המנשק: זה המתואר ב-Point2D. הוא כולל מתודות שמתייחסות לקואורדינטות הקרטזיות ואחרות המתייחסות לאלה הפולריות. שתי המחלקות יצטרכו לתרגם ממערכת אחת לשניה. לאלה שלא זוכרים (ולאלה שמעולם לא ידעו), הנה הנוסחאות המקשרות בין שתי מערכות צירים אלה:

$$\rho = \sqrt{x^2 + y^2} \qquad x = \rho \cos \varphi$$

$$\varphi = \tan^{-1}\left(\frac{y}{x}\right) \qquad y = \rho \sin \varphi$$

על כל אחת מהמחלקות לממש את **כל המתודות** שבמנשק. כמו כן, יש לממש בכל אחת מהן בנאי המקבל את הקואורדינטות שמתאימות לה: ב-()Cartesian על הבנאי לקבל x ו-y, ב-()Polar עליו לקבל ρ ו-φ.



באתר יש Main.java המכיל תכנית בדיקה לקוד שכתבתם. התכנית מצפה לפרמטר: c, p או m.
"c" גורם להרצה עם מופעים מהמחלקה Cartesian, "p" פועל עם מופעים של Polar,
ו-"m" משתמש בהם בערבוביה.

(את הפרמטר רושמים בחלון Run→Run Configuration→Arguments→Program Arguments.)

אם מימשתם הכל כראוי, התכנית אמורה לרוץ ללא שגיאה, ויתר על כן, לתת בדיוק את אותן תוצאות בכל המקרים. אם יש שגיאות, **אין לשנות את תכנית הבדיקה בשום אופן**. יש לשנות רק את הקוד שכתבתם אתם, עד שהשגיאות נפתרות.

כמה רמזים לגבי המימוש:

1. כשמגדירים למחלקה מנשק למימוש, eclipse מיד מציין שגיאה אם המימוש אינו כולל את כל המתודות של המנשק (מה שקרוב לודאי המצב אם המחלקה חדשה וריקה). לחיצה כפולה על סימן השגיאה האדום בשולי הקוד פותח חלון של אפשרויות תיקון אוטומטיות. אחת מהן (לרוב הראשונה) היא להגדיר את כל המתודות הדרושות באופן ריק. אתם מוזמנים להשתמש באופציה זו תחילה, ואחר כך לעבור על כל אחת מהמתודות הריקות שנוצרו ולמלא אותן בתוכן מתאים. אבל אל תשאירו מתודות ללא מימוש! כל אחת מהן צריכה תוכן.

2. אם תזדקקו לפונקציות מתמטיות שונות (למשל sqrt(), sin(), cos(), $\tan^{-1}$()), כולן מוגדרות מראש כמתודות סטטיות במחלקה Math. המשמעות של העובדה שהן סטטיות הוא שניתן לקרוא להן ישירות מבלי ליצור מופע של המחלקה Math לפני כן, למשל כך:

```
x = Math.cos( alpha );
```

במחלקה Math מוגדר גם הקבוע PI (=....3.14159).
אין ב-Math פונקציה של העלאה בריבוע: זאת אפשר להשיג על ידי הכפלת הגורם בעצמו.
אל תשתמשו במתודות exp() או pow(), מאחר ואלה משתמשות בלוגריתמים לביצוע החישוב, מה שעושה אותו יותר יקר ופחות מדויק.

3. במתודות של ה-setters (כמו setX() או setRho()) יש לדאוג שקביעת הערך של אחת הקואורדינטות לא תשפיע על האחרת. למשל, המימוש של setX() במחלקה Polar צריך להיות כזה שהערך שמוחזר על ידי getY() לא ישתנה עקב ביצועו. התיעוד בתוך הקוד של Point2D אומר שעל setX() להזיז את הנקודה ימינה או שמאלה, כך שקואורדינטת ה-x שלה תהיה שווה לערך הנתון למתודה, אך ללא שנוי ב-y. בדומה, המתודה setRho() ב-Cartesian צריכה לדאוג שהערך שמוחזר על ידי getPhi() לא ישתנה. שוב, התיעוד בתוך הקוד של Point2D אומר שעל setRho() להאריך או לקצר את הרדיוס-וקטור לנקודה הקיימת, כך שאורכו של יהיה הערך שניתן למתודה, אך הכוון של הרדיוס-וקטור צריך להשאר כשהיה. על setY() ב-Polar ושל setPhi() ב-Cartesian להתנהג כך אף הן.

4. בחישוב הזוית בין (הרדיוס-ווקטורים של) שתי נקודות במתודה angle() יש לחשב את הזוית בכוון הפוך ממחוגי השעון מהנקודה this (זו שעל המופע שלה המתודה נקראת) לנקודה השניה, המוצבעת על ידי הפרמטר. (אם p ו-q הם שתי נקודות, הקריאה p.angle(q) תחזיר את הערך הנגדי של הערך שיוחזר על ידי q.angle(p).

5. לעומת זאת, המתודה distance() מחזירה תמיד את המרחק החיובי בין שתי הנקודות, כך שתפקידן סימטרי. (אם p ו-q הם שתי נקודות, הקריאות p.distance(q) ו-q.distance(p) יחזירו את אותו הערך.)
כיון שהרדיוס-ווקטורים של שתי נקודות יחד עם וקטור המרחק ביניהן מהווה משולש, המימוש של המתודה distance() ב-Polar יכול להשתמש במשפט הקוסינוסים מטריגונומטריה:

$$c^2 = a^2 + b^2 - 2ab\cos\gamma$$

כאשר a ו-b הם הרדיוס-וקטורים, γ הזוית שביניהם, ו-c המרחק המבוקש.

בהצלחה!