**שאלה מספר 1 (70 נקודות)**

האתר "אנו מטיילים" הוא אתר לחובבי טבע המספק מסלולי טיולים ומידע על אתרים שונים ברחבי הארץ.

באתר נשמר מידע על נקודות ענין (PointofInterest). לכל נקודת ענין יש מיקום גאוגרפי (Location), שם מזהה (יחיד) ותיאור תמציתי. מיקום גיאוגרפי של נקודה מתייחס למיקומה על המפה והוא תלוי בשני מספרים (x ו-y). נקודת ענין יכולה להיות מסוג נקודת שירות (ServicePoint) המספקת שירותים (כמו הסעדה, מכירה או לינה) או נקודת פעילות (ActivityPoint) בה ניתן להנות מפעילויות כמו טיפוס, פעילות לילדים , צפייה בנוף וכדומה.

לכל נקודת שירות יש דרוג כוכבים (1-5) המציין את רמת השירות. לכל נקודת ענין יש דרגת קושי (1-6), דרגת ענין (1-5) וחיווי אם הפעילות כרוכה בתשלום או לא.

באתר מוגדרים אזורים (Areas). כל נקודה משויכת לאזור מסוים. כמו כן מוגדרים באזור שבילי הליכה בין נקודות הענין השונות. ניתן לדעת אם קיים שביל הליכה בין שתי נקודות ואם קיים, מה אורכו.

האתר מציע מסלולי הליכה: מסלול מחבר מספר נקודות ענין שונות באזור מסוים. המשתמשים המתחזקים את האתר יכולים להגדיר נקודות ענין, לשייך אותן לאזור, להגדיר ביניהן שבילים, ליצור מסלולי הליכה, להוסיף להם או להוריד מהם נקודות ענין .

**חלק א' (15 נקודות)**

ציירו תרשים מחלקות של המערכת. יש לשמור על עקרון הכימוס כראוי. כללו בתרשים מצייני גישה. כללו בתכנון רק את התכונות והפעולות המצוינות לעיל. אין צורך להסביר את התכנון.

**חלק ב' (55 נקודות)**

קוד המחלקה Location מומש והוא מספק את הממשק הבא:

|  |  |  |
| --- | --- | --- |
| 1 | בנאי המקבל את ערכי x ו-y. | **public** Location(**double** x,**double** y) |
| 2 | בנאי העתקה | **public** Location(Location location) |
| 3 |  | **public** **void** setLocation(**double** x,**double** y) |
| 4 |  | **public** **void** setX |
| 5 |  | **public** **void** setY(**double** y) |
| 6 |  | **public** **double** getX() |
| 7 |  | **public** **double** getY() |

כמו כן מומש קוד המחלקה Area עבור אזור. המחלקה מממשת את הממשק Iterator<PointofInterest> ומספקת את הממשק המובא בעמוד הבא.

|  |  |  |
| --- | --- | --- |
| 1 | בנאי המקבל את מספר נקודות הענין שהאזור מכיל. | **public** Area (**int** size) |
| 2 | מתודה המחזירה איטרטור (של נקודות ענין) Iterator<PointofInterest> | **public** Iterator<PointofInterest> iterator() |
| 3 | מתודה להוספת נקודה p לאזור. | **public** **boolean** addPointofInterest(PointofInterest p) |
| 4 | מתודה להגדרת מרחק בין נקודות p1 ו-p2. אם אחת הנקודות או שתיהן אינן נמצאות באזור המתודה תחזיר false. אחרת הנקודה תוסף לאזור ויוחזר true. | **public** **boolean** setDistance(PointofInterest p1, PointofInterest p2, **double** distance) |
| 5 | מתודה המקבלת שתי נקודות ענין ומחזירה distance.  distance = -1 אם אחת מהנקודות לא שייכת לאזור או אם לא קיים שביל הליכה בין p1 לבין p2.  0 < = distance אם קיים שביל הליכה בין הנקודות באורך distance (אפס פירושו ששתי הנקודות נמצאות באותו Location). | **public** **double** getDistance(PointofInterest p1, PointofInterest p2) |

1. **(18 נקודות)** יש לממש את המחלקות PointofInterest, ServicePoint ו -ActivityPoint. לכל נקודה יש שם, מיקום (מטיפוס Location) ותיאור (מטיפוס מחרוזת).

למחלקה ServicePoint יש דרוג של מספר כוכבים (1-5).

למחלקה ActivityPoint יש דרוג קושי (1-6(, דרוג ענין (1-5( ומשתנה המציין אם הפעילות כרוכה בתשלום או לא. יש לממש מתודות set ו-get .

1. **(12 נקודות)** יש לממש מחלקה בשם AreaUtil הכוללת מתודה סטטית אחת בעלת החתימה הבאה:

**public** **static** ArrayList<PointofInterest> areaPoints (Area area)

המתודה מקבלת מופע מטיפוס Area ומחזירה ArrayList של כל נקודות ענין (PointofInterest) השייכות לו.

1. **(5 נקודות)** בזמן יצירת מסלול (ביצוע פעולות של הוספת נקודות ענין או הורדתן) עלולות להתעורר חריגות (למשל כאשר מבקשים להוסיף למסלול שתי נקודות ענין עוקבות שאין ביניהן שביל הליכה). ממשו מחלקת חריגות בשם PointException המרחיבה את RuntimeException ומאפשרת יצירת חריגה עם הודעה (שתכלול הסבר על סוג החריגה).
2. **(20 נקודות)** יש לממש את המחלקה .Path המחלקה אמורה לשמור מידע על מסלול מסודר של נקודות ענין (כלומר יש נקודה ראשונה, שניה וכן הלאה) עם הממשק המוגדר להלן:

|  |  |  |
| --- | --- | --- |
| 1 | בנאי המקבל מופע מטיפוס Area המציין את האזור בו מוגדר המסלול. | **public** Path (Area area) |
| 2 | מתודה המוסיפה את הנקודה point בסוף המסלול. | **public** **void** addPoint(PointofInterest point) |
| 3 | מתודה המוסיפה את הנקודה point במקום position במסלול (position החל מ-0). | **public** **void** addPoint(PointofInterest point, int position) |
| 4 | המתודה toString | **public** **void** toString() |

הערות למימוש:

* ניתן להוסיף נקודה למסלול רק אם היא שייכת לאזור (למופע area) ואם היא איננה עדיין על המסלול.
* אם position שווה או גדול מאורך המסלול, מוסיפים את הנקודה לסוף המסלול.
* ניתן להוסיף נקודה לסוף המסלול רק אם קיים שביל הליכה בינה ובין הנקודה האחרונה במסלול.
* ניתן להוסיף נקודה למסלול במקום position רק אם קיים שביל הליכה בינה ובין הנקודה הנמצאת במקום position – 1 ובינה ובין הנקודה במקום position.
* במידה ולא ניתן להוסיף את הנקודה על המסלול יש לזרוק חריגה עם הודעה מתאימה:
  + אם הנקודה לא נמצאת באזור - "Point out of area".
  + אם הנקודה כבר קיימת במסלול - "Point is already on the path".
  + אם לא קיים שביל – "There is no route".
* המתודה toString מחזירה מחרוזת הכוללת את שמות הנקודות במסלול ואת אורכו. לדוגמא עבור מסלול הכולל 3 נקודות (נקודה ראשונה ששמה "Carmel View" המרוחקת ב-1 ק"מ מהנקודה השניה ששמה "The Best Food" ונמצאת 3 ק"מ מהנקודה האחרונה ששמה "Our Park") תוחזר המחרוזת הבאה:

Carmel View – The Best Food – Our Park"

" Length =4

להלן מתודת main המשתמשת עם מחלקות התכנית ועם הפלט שלה:

**public** **static** **void** main (String args[]) {

Location l1 = **new** Location(12,50);

Location l2 = **new** Location(120,30);

Location l3 = **new** Location(132,10);

Location l4 = **new** Location(112,250);

Location l5 = **new** Location(10,5);

PointofInterest p1= **new** PointofActivity

("Carmel View",l1,"Point of view", 1,4,**false** );

PointofInterest p2= **new** PointofActivity

("Carmel Park",l2,"Children park", 1,3,**true** );

PointofInterest p3= **new** PointofActivity

("The mountain",l3,"Climbing", 5,5,**true** );

PointofInterest p4= **new** PointofService

("Fast food",l4,"Fast food resturant", 1 );

PointofInterest p5= **new** PointofService

("The Best",l5,"Chef resturant", 4 );

Area north = **new** Area(10);

north.addPointofInterest(p1);

north.addPointofInterest(p2);

north.addPointofInterest(p3);

north.addPointofInterest(p4);

north.setDistance(p1, p2, 2);

north.setDistance(p2, p3, 1);

north.setDistance(p1, p3, 1.5);

north.setDistance(p3, p4, 2);

Path path2 = **new** Path(north);

Path path1 = **new** Path(north);

path1.addPoint(p4);

System.***out***.println("Path 1:\n"+path1);

**try** {

System.***out***.println("Trying to add p5");

path2.addPoint(p5);

}

**catch** (PointException e) {

System.***out***.println(e);

}

**try** {

System.***out***.println("Trying to add p1 and p1");

path2.addPoint(p1);

path2.addPoint(p1);

}

**catch** (PointException e) {

System.***out***.println(e);

}

**try** {

System.***out***.println("Trying to add p2 and p4 after p1");

path2.addPoint(p2);

path2.addPoint(p4,1);

}

**catch** (PointException e) {

System.***out***.println(e);

}

path2.addPoint(p3,1);

System.***out***.println("Path 2:\n"+path2);

}

}

הפלט שיתקבל הוא:

Path 1:

Fast food

length = 0.0

Trying to add p5

PointException: Point out of area

Trying to add p1 and p1

PointException: Point is already on the path

Trying to add p2 and p4 after p1

PointException: There is no route

Path 2:

Carmel View - The mountain - Carmel Park

length = 2.5