מבוא לחישוב - שיעור מספר 10

תכנות מונחה עצמים 2:

המחלקה הראשונה שלי **Point:**

```
public class Point {
   private int  _x,  _y ; // data member!!
   public Point (int x, int y) { _x = x;  _y = y}
   public int x() {return  _x;}
   public int y() {return  _y;}
   public void setX(int x) {_x = x;}
   public void setY(int y) { _y = y;}
} // class Point
```

מחלקה היא מבנה לוגי המאגד משתנים (תכונות) ופונקציות תחת שם אחד.

אובייקט הוא משתנה מהטיפוס של המחלקה.

משתנים השייכים למחלקה (_x, _y) מוכרים לכ הפונקציות השייכות אליה וערכי המשתנים קיימים כל עוד האובייקט קיים.

סוגי ההרשאה: public ו-private

private – אומר שכל המשתנים והפונקציות אשר יוגדרו אחריו הם פרטיים ויוכלו להשתמש בהם רק הפונקציות של המחלקה, ואילו public אומר שכל המשתנים והפונקציות אשר יוגדרו אחריו הם ציבוריים ויוכרו על-ידי כלל הפונקציות.

שימוש במחלקה כולל הרשאות:

```
class TestPoint {
   public static void main(String[] a) {
         Point p1 = new Point(2,3);  //constructor!
         p1._x = 5;  // ERROR (its private).
         p1.setX(5); // o.k. public
         boolean b = p1.equal(p2);  // true
```

לאחר שהגדרנו מחלקה ניתן להשתמש בה ממש כשם שמשתמשים בטיפוס של משתנה.

אופרטור . (נקודה) משמש לצורך גישה למשתנים או לפונקציות של אובייקט.

שיטות נוספות של נקודה: מרחק, הזזה, השוואה , toString:

```
class Point {
   …
   public double distance(Point p) {
          double dx = x()-p.x(),  dy = _y-p._y;  // x() , _x
          return Math.sqrt(dx*dx + dy*dy);
   }
   public void  rescale(Point p) {
          _x = _x + p.x();
          _y = _y + p.y();
   }
   public boolean   equal(Point p){
         boolean   ans =   false;
         if(_x==p.x() && _y==p.y()) ans = true;
         return ans;
  }
  public String toString(){
     String s =  "(" + _x + "," + _y+")";
     return s;
  }
} // class Point
```

פונקציות בונות - constructors

ה- constructor הוא שיטה שתפקידה לאתחל אובייקט חדש. שמו של constructor זהה לשם של המחלקה שבה הוא מוגדר. אסור שהשיטה תכלול את הפקודה return. אפילו לא return void.
יש לשים לב לכך שאם לא מגדירים במחלקה אף constructor אז קיים ה-Default Constructor, אשר קיים באופן אוטומטי בכל מחלקה שמגדירים ב-JAVA כל עוד לא הגדרנו בה constructor כלשהו. ה-Default Constructor הוא הפונקציה הבונה אשר מופעלת כאשר יוצרים אובייקט חדש מהמחלקה מבלי לשלוח ערכים בעת יצירתו, ובפעולתה מכניסה לכל אחד ממשתני המחלקה את ערך ברירת המחדל על פי טיפוס הערך שלו.

אופרטור new מחזיר את מצביע לאובייקט ולא אובייקט עצמו.

בנאי מעתיק copy constructor מיצר עותק מדויק של אובייקט - העתקה עמוקה.

```
public class Point {
   private int  _x,  _y ; // data member!!
   public Point (int x, int y) { _x = x;  _y = y} //constructor
   public Point(Point p){ _x = p.x(); _y = p.y();} // copy constructor
   . . .
} // class Point
```

```
class TestPoint {
   public static void main(String[] a) {
          Point p1 = new Point(2,3);  //constructor!
          Point p2 = new Point(p1);  //copy constructor! (new point)
          Point p3 = p1; // reference to the same point
          p1.setX(7); // p1 and p3 are changed! p2  does not change!
}
```

המחלקה השניה שלי Circle

המחלקה מייצגת מעגל במישור:
מידע: נקודה ורדיוס.
בנאי, copy constructor.

```
class Circle {
        private Point _center;
        private double _radius;
        public Circle(Point cen, double rad) {//constructor
                _center = new Point(cen);
                _radius = rad;
        }
        public Circle(Circle c) { //copy constructor!
                _center = new Point(c.center(), c.radius());
                _radius = radius;
        }
        public Point center(){return _center;}
        public double radius(){return _radius;}
        }
```

שיטות שונות:

שטח, היקף.
toString - מחזירה מחרוזת שמייצגת את האובייקט, תופעל כאשר נפעיל את הפונקציה System.out.print על אובייקט.
האם נקודה מוכלת

```
class Circle { …
   public double area() {
      return Math.PI*_radiud*_radius;
   }
   public double perimeter() {
      return 2*Math.PI*_radiud;
   }
   public String toString() {
      return "Circle: cen:"+_center+" rad: "+_darius;
   }
```

```
  public boolean contains(Point p) {
      Boolean ans = false;
      if (_center.distance(p) <= _radius)  ans = true;
      return ans;
  }
} // class Circle
```

שימוש במחלקות Point, Circle

```
class TestCP {
    public static void main(String[] a) {
        Point p1=new Point(5,6),   p2=new Point(1,3);
        double d = p1.distance(p2); // 5.0
        Circle c1 = new Circle (p1,4);
        boolean in = c1.contains(p2); // false
    }
}
```

מילה שמורה this

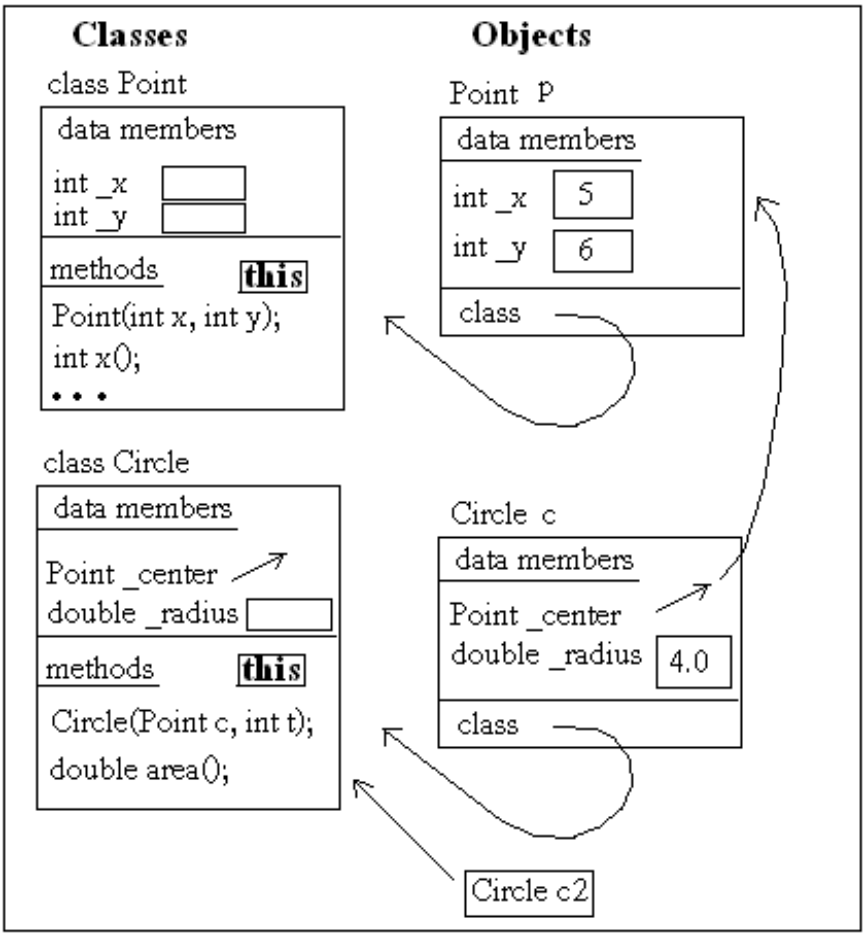
- מילה שמורה, מייצגת מצביע לעצם המפתח.
- כאשר מפעילים שיטה (על אובייקט), המצביע לאובייקט מועבר לשיטה (בצורה נסתרת) וכך השיטה מודעת לכל משתני העצם של האובייקט, מצביע זה נקרה this.
- טווח ההכרה של this רק בגבולות הקוד של המחלקה.
- המצביע this הינו קבוע ולא ניתן לשנות את ההצבעה שלו.

שיכון אובייקטים מזיכרון: this

מכל מחלקה אפשר ליצור אובייקטים רבים.

this מצביע על עצם עליו הפעלנו את השיטה

```
Point p=new Point(5,6);
Circle c=new Circle(p,4);
Circle c2;
```



Method Overloading - שיטות חופפות

בתוך אותה מחלקה ניתן לכתוב מספר methods בשם זהה, שמה שמבדיל ביניהן הוא מספר הפרמטרים ו/או סוגם. כדי שניתן יהיה לכתוב מספר מתודות בשם זהה חייב להיות שוני או במספר הפרמטרים או בטיפוס הערך שלהם. מתודה ששמה זהה לשמה של מתודה אחרת שכבר קיימת במחלקה נקראת בשם: overloaded method.

המחלקה אוסף של נקודות PointContainer

יכולות של אוסף (Container):

יצירה: אתחול.

הוספה, בדיקת תקינות, גדילה אוטומטית.

מספר איברים, איבר במקום.

שוויון לוגי.

שיכפול.

```
public class PointContainer {
        // *** data members ***
        public static final int INIT_SIZE=10;  // init size.
        private int _sp=0;          //   "stack pointer"
        private Point[] _points;  //   the point array
        /*** Constructors: creates a empty set ***/
        public PointContainer(){
                _sp=0;
                _points = new Point[INIT_SIZE];
        }
public int size() {return _sp;}
public void add (Point p){
   if (p != null){
   if(_sp==_points.length) rescale(10); // RESCALE
  _points[_sp] = new Point(p);  // deep copy semantic.
  _sp++;
}
public Point at(int p){
  if (p>=0 && p<size()) return _points[p];
  return null;  // no need for else!
}
/************** private methodes *************/
private void rescale(int t) {
   Point[] tmp = new Point[_sp+t];
   for(int i=0;i<_sp;i++) tmp[i] = _points[i];
    _points = tmp;
   }
```