♥ Java ♥

المحاضرة السادسة عشر الاثنين 27/11/2017

ملاحظات:

* لا يوجد في لغة Java هادم (الهادم مهمته تحرير مكان حجز الـ object في الذاكرة). ولكن يوجد خوارزمية اسمها (Garbage collector) تقوم بحذف الأماكن الغير مستخدمة والمحجوزة في الذاكرة من قبل البرنامج دون الحاجة لكتابتها أو استدعائها.

* في حال عدم كتابة باني داخل الهوف تولد اللغة باني افتراضي (ليس له وسطاء ويعطي المطيات الأعضاء قيم ابتدائية)
الباني الخالي والباني الافتراضي لهما نفس العمل ولكن الباني الخالي هو الذي أقوم بكتابته وأما الافتراضي هو الذي تولده اللغة وعادةً لا يتواجدان معاً.

* بمجرد وجود باني واحد على الأقل داخل الهوف لن تقوم اللغة بتوليد الباني الافتراضي.

مثال: أكتب صفاً يقوم بتمثيل نقطة في المستوي $\mathbb{R}^2$ (الاحداثيات الديكارتية) ثم يجز التالي:
1. ادخال ثلاث نقاط لا تقع على استقامة واحدة.
2. حساب مساحة المثلث المشكل من هذه النقاط.

اسم الملف: Point.java

```
class Point {
    double x;
    double y;
    Point(){ x=0; y=0;}
    Point (double x, double y){
        this.x = x;
        this.y = y;}
    double dist (double x, double y) {
        double R = Math.sqrt(Math.pow(this.x-x,2) + Math.pow
                   (this.y-y,2));
        return R;}
```

المطيات الأعضاء

الباني الخالي

باني آخر يقوم بإعطاء قيم مدخلة للمطيات الأعضاء

دالة لحساب البعد بين نقطتين والتي سعاملها هي مركبات النقطة الثانية

دالة أخرى لحساب المسافة بين نقطتين ووظيفتها حساب المسافة بين النقطتين

```
double dist (Point a, Point b){
    double R;
    R = Math.pow(a.x - b.x, 2) + Math.pow(a.y - b.y, 2);
    return (Math.sqrt(R));
        }
    void print(){
     Sestem.out.println("(" + x + "," + y + ")");
                }
        } //end of class
```

UsePoint.java :اسم الملف

```
class UsePoint{
 public static void main (String[]args){
  double x1, y1, x2, y2, x3, y3;
  do{ x1 = Stdin.readDouble();
      y1 = Stdin.readDouble();
      x2 = Stdin.readDouble();
      y2 = Stdin.readDouble();
      x3 = Stdin.readDouble();
      y3 = Stdin.readDouble();
  }while(((x2-x1)/(x3-x1)) == ((y2-y1)/(y3-y1)));
  Point P1, P2, P3;
    P1 = new Point(x1, y1);
    P2 = new Point(x2, y2);
    P3 = new Point(x3, y3);
   double d1, d2, d3;
   d1 = P1.dist(P2.x, P2.y);
   d2 = P2.dist(P3.x, P3.y);
   d3 = P2.dist(P1, P2);
```

ادخال ثلاث نقاط شرط لا يتحقق على استقامة واحدة

تعريف ثلاث اغراض من Point

بناء الاغراض باستخدام البانى

حساب بعد النقطة P1 عن النقطة P2

حساب بعد النقطة P2 عن النقطة P3

حساب بعد النقطة P1 عن النقطة P3

```
double P=d1+d2+d3;
    P=P/2;
double a=P-d1;
double b=P-d2;
double c=P-d3;
System.out.print(Math.sqrt(P*a*b*c));
} //end of main
} // end of class
```

ملاحظات :

* كل صف يوضع في ملف (لهي ملفين) اسم الملف يكون من اسم الصف
ممكن أن يكون اسم الملف لا يطابق اسم الصف إذا لم يكن هناك كلمة public قبل الصف ولكن يفضل أن يكون من اسم الصف ولكن في الحالة العامة

اسم الملف من اسم الصف العام

* لا يمكنني استدعاء دالة مكتوبة داخل صف في صف آخر دون وجود object من نوع صف الدالة.
* this : كلمة محجوزة وتعني هذا الصف أو اسم آخر للصف داخل الصف تستخدم أمام المعطيات الأعضاء أو الطرق الأعضاء وليس لها معنى خارج الصف يمكن اعتبارها مؤشر ضمني ولكن في لغة الـ Java لا تستخدم المؤشرات برمجياً وبالتالي نحن نتعامل مع this على أنها object داخل الصف
* قانون مساحة المثلث هو: $\sqrt{\frac{P}{2}(\frac{P}{2}-a)(\frac{P}{2}-b)(\frac{P}{2}-c)}$ حيث P هو المحيط أي $P=a+b+c$ حيث a و b و c أطوال أضلاع المثلث.
* عند كتابة دالة داخل الصف فليس هناك داعي لكتابة معطيات الصف كوسطاء للدالة (عند كتابة الدالة الأولى كان بعد نقطتين استخدمت وسطاء الدالة هي مكونات النقطة الثانية لأن لدينا النقطة الأولى التي يمثلها الصف).
* استطعت كتابة دالتين داخل الصف لهما نفس الاسم لاختلاف توقيعهما.
* عند استدعاء الدالة من خلال object فلا يوجد فرق بين أي object يتم استدعائها مع مراعاة عمل الدالة وهل يتعلق عملها بالـ object الذي تم استدعائها منه

التوقيع الحاضرة