* Not all ways of creating objects are equal, some are better including encapsulation
* Lecture 6 Example 1 class:
  + First create a point class with constructors, fields, and method called distanceTo() that calculates the distance between two points
  + Then create a triangle class that utilizes point classes, constructors, fields, and methods that calculate the perimeter using point instance methods
* When you change coordinates of points, make sure these coordinate values are not shared by another triangle. This can be a problem when you change one triangle's points, which inadvertently changes the points of another triangle

* When working with triangles as classes, it's important to check that the three points that define a triangle aren't colinear(because you can't make a triangle from just a line of three points). Check this by checking if the area is less than 0.0000001;

* Principle of encapsulation: don't expose your internals to the rest of the world, or else your internals of a class may be altered unexpectedly
  + Separate exposed behavior from internal behavior.
* Steps:

1. Make all fields private(for now in the class) by amend the field declaration with 'private' access modifier
2. Create getters and setters
   1. In order to get to these private fields, use methods that act as getters and setters
   2. Analogy: instead of every peasant having access to the king, have delegates who act as a liaison that talk to peasants and report to the king. The delegates are getters and setters.
   3. If you don't provide setter and getter methods in addition to making fields private, **then anything can change the fields and this can screw things up!**
   4. Benefits of getters/setters:
      1. Setters can validate new values to prevent misuse
         1. If illegal/improper values are put into a setter, you can create a runtime exception error that looks like this:

If(points.length != 3) {

throw new RunTimeException("Your Error message");

}

1. You can also create another method that checks if a certain value is equal to an object's field like so:
   1. boolean equals(Point other) [

return((this.x==other.x)&&(this.y==other.y));  
]

1. How to make getters/setters:
   1. Give type T and property P
      1. Getter:
         1. Public T getP() { return P;}
      2. Setter:
         1. Public void setP(T value){this.P = value;}

* Derived properties
  + Properties that are a combination or transformation of existing fields
  + Ex: sides of class triangle are fields, while perimeters are derived properties
  + This is necessary b/c it minimizes the number of fields of a class that you have to take care of, but rather you just create these derived properties off of what is absolutely necessary
  + You use setters/getters as well
* Sometimes you may want to use getters/setters internally, such that methods of a class have to use getters and setters
* Static class fields creates a global variable that can be accessed anywhere
  + Avoid this if you don’t want your variable to be accessed globallyu