1. `==` compares identity while `.equals` compares values, with String's this is the difference between comparing "`a`" to "`a`" (`.equals`, true) and `a`(ref "a") to `b`(ref "a") (`==`, false). `==` is useful when you want to know if two "objects" (variables/references to objects) are the same, a direct example being in a linked list when locating a node.
2. a `NullPointerException` occurs when you try to call methods or preform actions on uninitialized (null) objects. The simplest way to avoid a `NullPointerException` is to use an if statement checking if the variable == null. It's also important to assign default values to all attributes, thus warding off a whole class of `NullPointerException`'s
3. You can instantiate a Scanner that reads from the standard input, and allows you to quite easily get the next int. ex:
    (we enter '4' at the console, and the following code eventually is invoked as part of some method)

```
    intscanner = new Scanner(System.in);
    System.out.println(intscanner.nextInt()); //(in this case, it
    would print '4')
```

4. Separated presentation design allows the underling data methods to be changed at any time, and allows the UI to concentrate solely on the presentation and formatting of data, rather than on the manipulation of data. with the airport program we can test any class and method without having to test the UI, and we can test the UI by supplying mock data/ classes.
5. encapsulation ensures that underlying data structures, routines, and logic, can be replaced wholesale with no modification to any code dependent on it. Information hiding allows the underlying data to be represented in the best possible manner (or really any manner) for the problem, while methods return data in a sane and consistent manner.
6. explanations:
    - TwoDimensionalPoint (constructor the first) - calls this with null as it's last argument (eg, the next point will be null)
    - TwoDimensionalPoint (constructor the second) - assigns x,y and the next point in a Polygon
    - addPointToPolygon - loops through the polygon until it reaches the end (nextInPolygon == null or nextInPolygon == this), set's the last point's nextInPolygon attribute to be the point we are adding, and the point we are adding nextInPolygon to the calling point
    - removePointFromPolygon - if we are removing the calling point from the polygon we loop until we reach the point before the calling point then we set its next point to be the calling points next point(removing the calling point) and return that point. otherwise we loop until we either reach null or the next point is the point we want to remove, if the next point isn't null we set the next point to be the point after the next point (removing the point we want to remove), we then return the calling point (even if the next point was null)
    - lowestPointInPolygon - we loop through the polygon until the next point is either null or we have found the lowest point in the polygon (done == true). if the current point's y value is less than the lowest y value we have seen it becomes the new lowest point. if the current point equals the calling point we have looped the whole polygon, and are done. finally we return the lowest point.
    - perimeterOfPolygon - we loop through the polygon until we reach the calling point (or null), and calculate the perimeter as sqrt(dx*dx + dy*dy) where dx,dy is the difference between the current and next points). we then return the calculated perimeter
    - printPoint - prints the calling points x and y values to the console, formatted as "(calling point x, calling point y)"

- printPolygon - loops through the polygon until it reaches the calling point (or null) and calls printPoint() for each point.

encapsulation:

- a Polygon class would take control of adding, removing, summing(perimetering?), finding the lowest point, and printing the points it contains, while the TwoDimensionalPoint class would be responsible for stringifying itself and keeping a reference to the next point (much like a node in a linked list). the Polygon class would keep a reference to the root point. All attributes of both classes wold be private. TwoDimensionalPoint would have setters/getters for all attributes (x,y could reasonably be changed or read at any point, and nextInPolygon needs to be gotten or set at any point), Polygon would have a getter for the root point, you don't want something else telling you what the root point is.