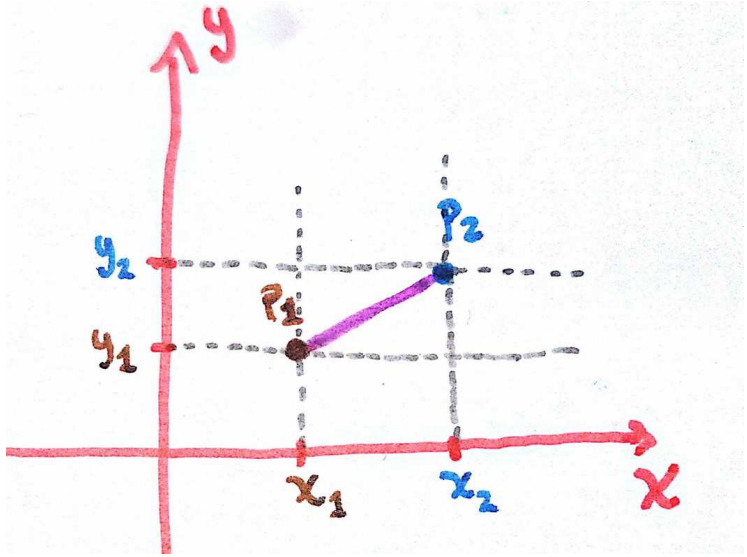
Objects and Classes

A simple exercise - Creating a 2D Point Class

- Following up on the code from last lecture
- The `Point` class
  - A `Point` has two coordinates: call them `x` and `y`
  - The coordinates should be **real** numbers (`double` type)
  - Write a class method that computes the distance to another `Point`



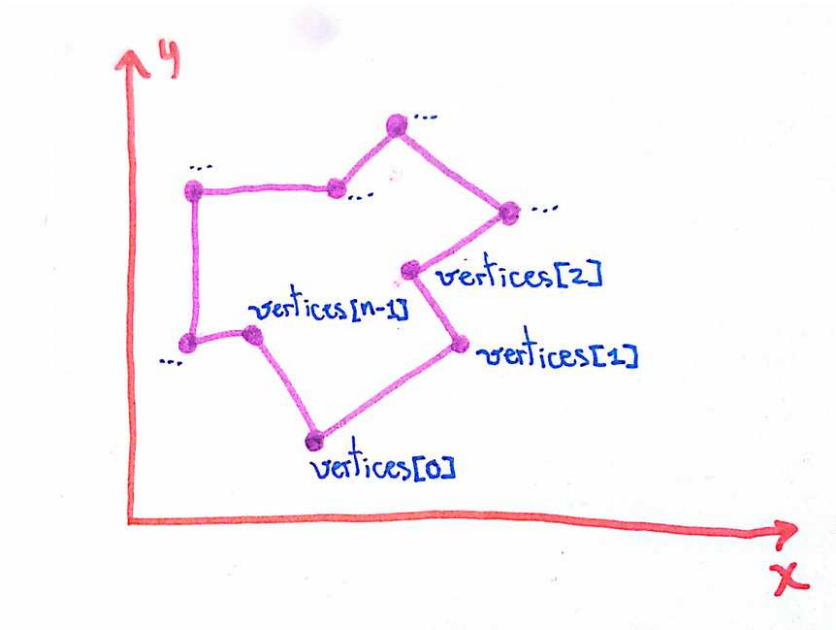
    - sqrt($(x_1 - x_2)^2 + (y_1 - y_2)^2$)

Another exercise - Creating a 2D Point Class, and using in a Polygon class

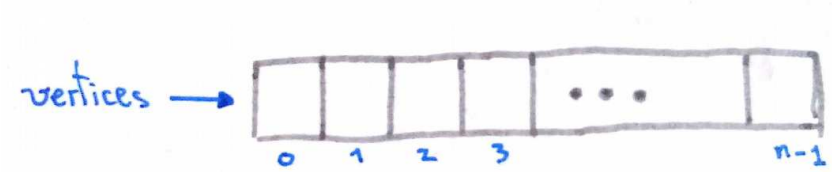
- The `Polygon` class
  - A `Polygon` has a list of `Point` objects; its vertices.
  - Write a class method that adds a new vertex to a polygon
  - Write a class method that returns `true` if the `Polygon` is **equilateral**
  - Write a class method that returns `true` if the `Polygon` is **regular**
  - Write a class method that returns the area of a polygon using the Shoelace Algorithm

Using `Point` in `Polygon`

- The `Polygon` class

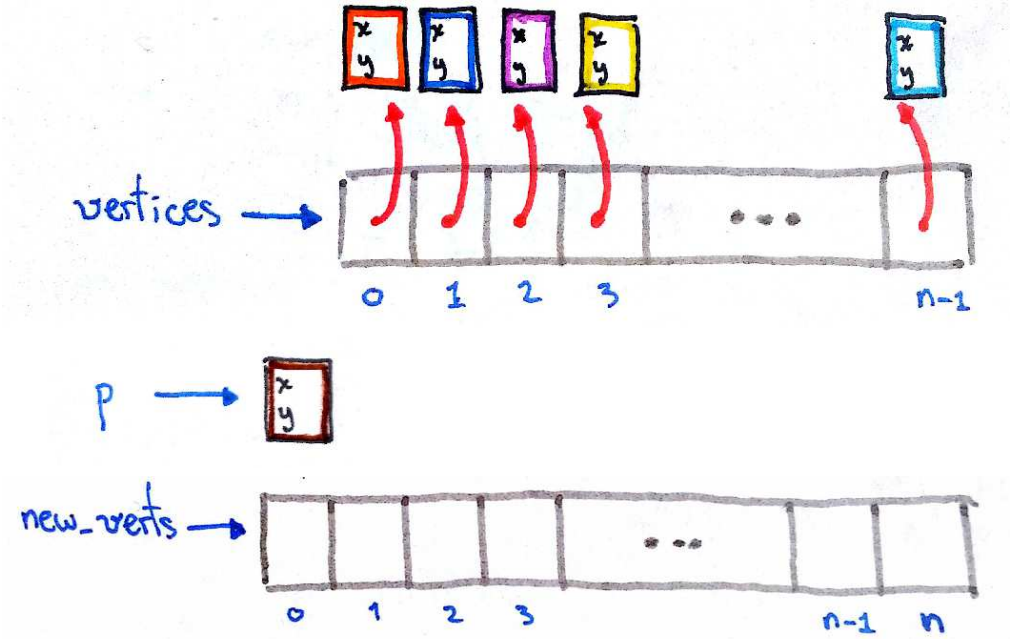


  - A `Polygon` has a list of `Point` objects; its vertices.



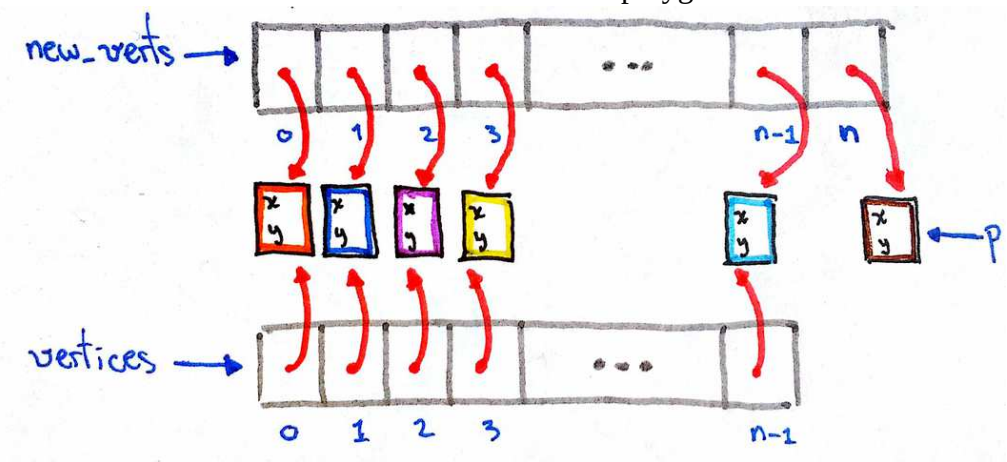
Adding a new `Point` to `vertices`

- The `Polygon` class
  - A `Polygon` has a list of `Point` objects; its vertices.
  - Write a class method that adds a new vertex to a polygon



Adding a new `Point` to `vertices`

- The `Polygon` class
  - A `Polygon` has a list of `Point` objects; its vertices.
  - Write a class method that adds a new vertex to a polygon



Completing the `Polygon` class

- The `Polygon` class
  - A `Polygon` has a list of `Point` objects; its vertices.
  - Write a class method that adds a new vertex to a polygon
  - Write a class method that returns `true` if the `Polygon` is **equilateral**
  - Write a class method that returns `true` if the `Polygon` is **regular**
  - Write a class method that returns the area of a polygon using the Shoelace Algorithm

Yet another example, a social network

Suppose you're in the year 2004 and had this great idea...

- of a website where people could spend countless hours
- uploading personal information, writing messages to people that you see daily
- and you realise that you could sell this information!

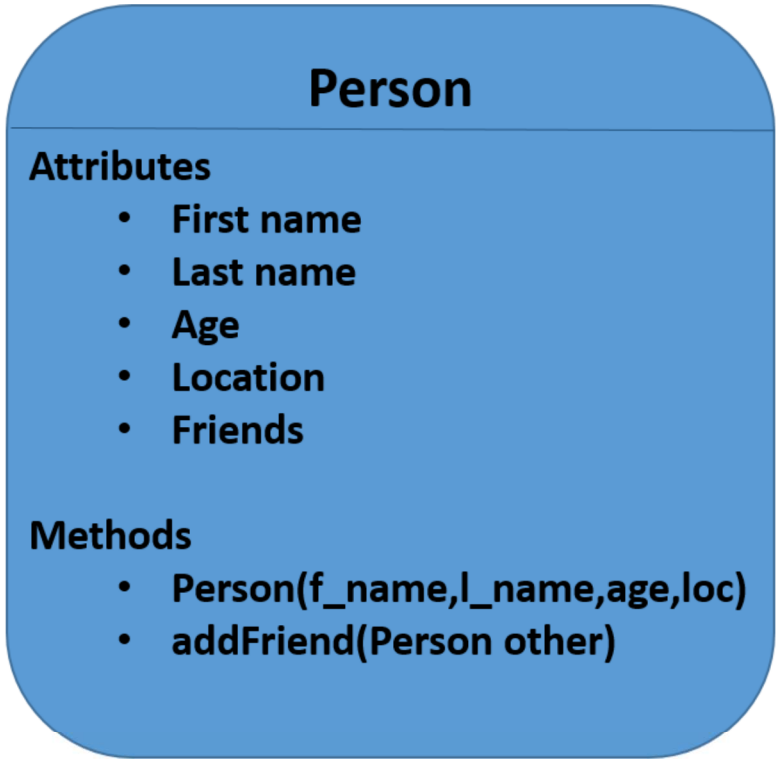
You decide you want to build the basic data infrastructure of your website first

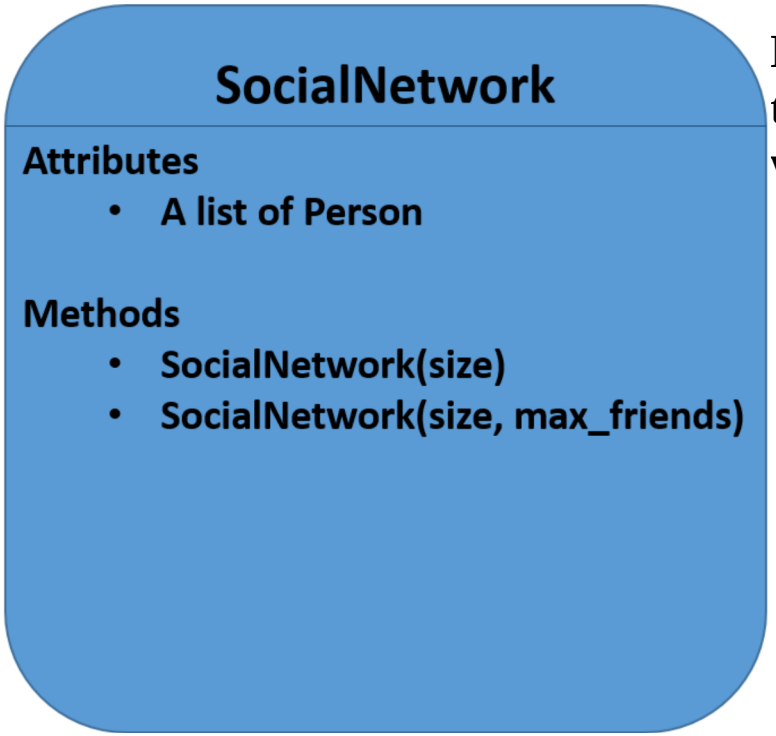
- A social network consists of people, each individual being a `Person`
- Each `Person` has a name, location, age and a list of friends

The first step on your road to success, fame and $$$

You decide you want to build the basic data infrastructure of your website first

- A social network consists of people, each individual being a `Person`
- Each `Person` has a name, location, age and a list of friends



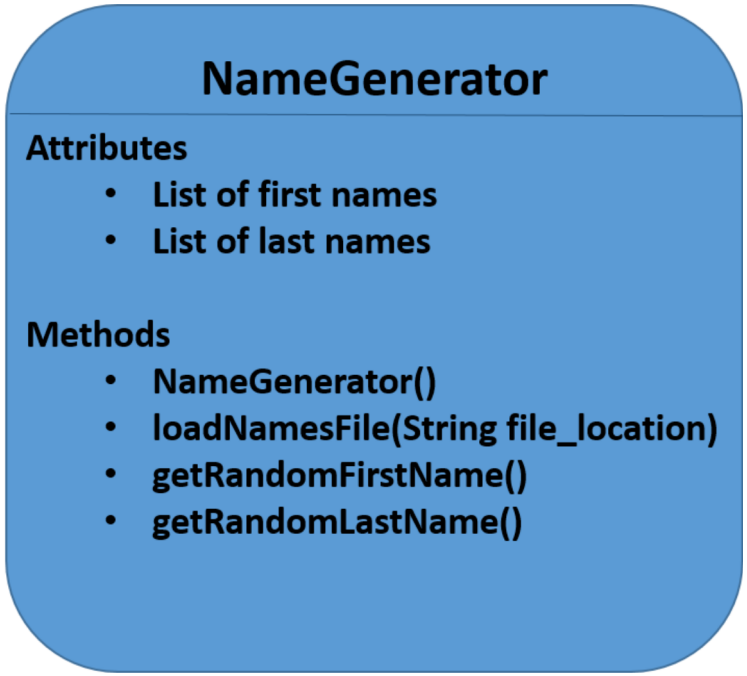


Now you want to test your software with fake data

You want to test your software with randomly generated people

To do this, you would like to generate a social network with randomly generated `Person` objects, each one with randomly assigned firends

- When constructing a `SocialNetwork` you would decide on the size of the network and the maximum number of friends
- Take a look at the NameGenerator class
- We want to add two methods: getRandomFirstName, and getRandomLastName



Now that you have a simulated social network, you are ready for pitching your idea

You want to generate silly statistics that would attract investors to your new startup

- Find the person with the largest number of friends
- Find the 2nd person with the largest number of friends
- Find the N-th person with the largest number of friends
- Find the total number of connections in your social network
- Find if you have isolated groups in your social network (see a4)
- Find the maximum **degree of separation**

Resources

- Classes and Objects:
  http://docs.oracle.com/javase/tutorial/java/javaOO/
- The Shoelace Algorithm:
  http://en.wikipedia.org/wiki/Shoelace_formula
- Suggested reading:
  How to think like a Computer Scientist, Chapter 11