Github link: https://github.com/arniki/ECC

`__init__`: This is the constructor method for the class, which is called when you create a new instance of the class. It initializes the point with the given values for x, y, a, b, and p. These values are stored as parameters of the object:

- `x (int)`: The x coordinate of the point.
- `y (int)`: The y coordinate of the point.
- `a (int)`: The a parameter of the curve.
- `b (int)`: The b parameter of the curve.
- `p (int)`: The prime modulus of the curve.

`add`: This method takes in another `EllipticCurvePoint` object called `other` and returns a new `EllipticCurvePoint` that is the result of adding the two points together on the curve. The curve is defined by the `a` and `b` parameters of the object, and all calculations are done modulo `p`.

There are two cases to consider:

1. If the two points being added are the same (i.e. `self.x == other.x and self.y == other.y`), then the slope of the line connecting the two points is not defined. In this case, we use a special formula to calculate the slope.
2. If the two points are not the same, then we can use the usual formula for the slope of a line to calculate the slope of the line connecting the two points.

Once we have the slope, we can use it to calculate the `x` and `y` coordinates of the new point.

`mul`: This method takes in a `scalar` value called scalar and returns a new `EllipticCurvePoint` that is the result of multiplying the given point by the scalar on the curve. It does this by adding the point to itself scalar-1 times.

`order`: This method returns an integer that represents the order of the point, which is the number of times the point must be added to itself to get the point at infinity (a special point on the curve that is represented by None for the `x` and `y` values). The method does this by adding the point to itself repeatedly until the `x` and `y` values of the result are None. It keeps a count of the number of additions performed, and returns this count when the point at infinity is reached.