```
cryp-11-Elliptic-Curve-Cryptography main ? ✕ cat ECCExample.java
import java.math.BigInteger;
import java.security.SecureRandom;

public class ECCExample {

    // Curve: y^2 = x^3 + ax + b (mod p)
    static BigInteger a = new BigInteger("2");
    static BigInteger b = new BigInteger("3");
    static BigInteger p = new BigInteger("97");   // small prime (demo only)

    // Base point G (must lie on curve)
    static ECPoint G = new ECPoint(
            new BigInteger("3"),
            new BigInteger("6")
    );

    static SecureRandom random = new SecureRandom();

    // ---------------- Point Class ----------------
    static class ECPoint {
        BigInteger x;
        BigInteger y;

        ECPoint(BigInteger x, BigInteger y) {
            this.x = x;
            this.y = y;
        }

        boolean isInfinity() {
            return x == null && y == null;
        }

        static ECPoint infinity() {
            return new ECPoint(null, null);
        }

        public String toString() {
            if (isInfinity()) return "Point at Infinity";
            return "(" + x + ", " + y + ")";
        }
    }

    // ---------------- Point Addition ----------------
    static ECPoint add(ECPoint P, ECPoint Q) {

        if (P.isInfinity()) return Q;
        if (Q.isInfinity()) return P;

        // If P == -Q → infinity
        if (P.x.equals(Q.x) &&
                P.y.equals(Q.y.negate().mod(p))) {
```

```
cryp-11-Elliptic-Curve-Cryptography main ? ❯ java ECCExample
Private Key: 3
Public Key: (80, 87)

Encrypted:
C1: (80, 10)
C2: (80, 87)

Decrypted Point: (80, 10)

cryp-11-Elliptic-Curve-Cryptography main ? ❯
```