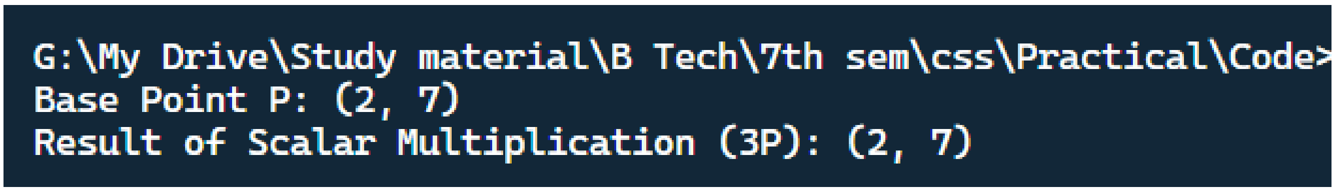
```
import java.math.BigInteger;

class EllipticCurvePoint {

    private BigInteger x;

    private BigInteger y;

    public EllipticCurvePoint(BigInteger x, BigInteger y) {

        this.x = x;

        this.y = y;

    }

    public static EllipticCurvePoint add(EllipticCurvePoint P, EllipticCurvePoint Q, BigInteger a,
BigInteger p) {

        BigInteger slope, x3, y3;

        if (P.equals(Q)) {

            slope =
(BigInteger.valueOf(3).multiply(P.x.pow(2)).add(a)).multiply(P.y.multiply(BigInteger.valueOf(2)).modIn
verse(p));

        } else {

            slope = (Q.y.subtract(P.y)).multiply(Q.x.subtract(P.x).modInverse(p));

        }

        x3 = slope.pow(2).subtract(P.x).subtract(Q.x).mod(p);

        y3 = slope.multiply(P.x.subtract(x3)).subtract(P.y).mod(p);

        return new EllipticCurvePoint(x3, y3);

    }

    public static EllipticCurvePoint scalarMultiply(EllipticCurvePoint P, BigInteger k, BigInteger a,
BigInteger p) {

        EllipticCurvePoint result = new EllipticCurvePoint(BigInteger.ZERO, BigInteger.ZERO);

        EllipticCurvePoint current = P;

        while (k.compareTo(BigInteger.ZERO) > 0) {

            if (k.testBit(0)) {

                result = add(result, current, a, p);

            }

            current = add(current, current, a, p);

            k = k.shiftRight(1);
        }
        return result;
    }
    @Override
    public String toString() {
        return "(" + x + ", " + y + ")";
    }
    public boolean equals(EllipticCurvePoint other) {
        return x.equals(other.x) && y.equals(other.y);
    }
}
public class Exp9 {
    public static void main(String[] args) {
        // Define the elliptic curve parameters
        BigInteger a = new BigInteger("1");
        BigInteger b = new BigInteger("6");
        BigInteger p = new BigInteger("11");
        // Define a base point P on the curve
        EllipticCurvePoint P = new EllipticCurvePoint(new BigInteger("2"), new BigInteger("7"));
        // Scalar multiplication: Compute 3P
        BigInteger scalar = new BigInteger("3");
        EllipticCurvePoint result = EllipticCurvePoint.scalarMultiply(P, scalar, a, p);
        System.out.println("Base Point P: " + P);
        System.out.println("Result of Scalar Multiplication (3P): " + result);
    }
}
```



```
G:\My Drive\Study material\B Tech\7th sem\css\Practical\Code>
Base Point P: (2, 7)
Result of Scalar Multiplication (3P): (2, 7)
```