AIM:- WAP to implement Digital Signature scheme using RSA.

```
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

public class RSADigitalSignatureWithPrimes {

    static BigInteger modExp(BigInteger base, BigInteger exp, BigInteger mod) {
        return base.modPow(exp, mod);
    }

    static boolean isPrime(BigInteger n) {
        return n.isProbablePrime(20);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        SecureRandom rand = new SecureRandom();

        System.out.print("Enter prime number p: ");
        BigInteger p = sc.nextBigInteger();
        if (!isPrime(p)) {
            System.out.println("p is not prime. Exiting...");
            return;
        }

        System.out.print("Enter prime number q: ");
        BigInteger q = sc.nextBigInteger();
        if (!isPrime(q)) {
            System.out.println("q is not prime. Exiting...");
            return;
        }

        BigInteger n = p.multiply(q);
        BigInteger phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Dynamically choose e and compute d
        BigInteger e, d;
        while (true) {
            e = new BigInteger(phi.bitLength(), rand);
            if (e.compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0 &&
phi.gcd(e).equals(BigInteger.ONE)) {
                d = e.modInverse(phi);
```

```
            if (!e.equals(d)) {
                break; // valid e found
            }
        }
    }

    System.out.println("\nKeys Generated Successfully!");
    System.out.println("Public Key (e, n): (" + e + ", " + n + ")");
    System.out.println("Private Key (d, n): (" + d + ", " + n + ")");

    System.out.print("\nEnter original message (number): ");
    BigInteger message = sc.nextBigInteger();

    BigInteger signature = modExp(message, d, n);
    System.out.println("Generated Digital Signature: " + signature);

    System.out.print("\nEnter received message (number): ");
    BigInteger receivedMessage = sc.nextBigInteger();

    System.out.print("Enter received signature: ");
    BigInteger receivedSignature = sc.nextBigInteger();

    BigInteger verified = modExp(receivedSignature, e, n);
    System.out.println("Decrypted Signature Value: " + verified);

    if (verified.equals(receivedMessage)) {
        System.out.println("Signature Verified Successfully!");
    } else {
        System.out.println("Signature Verification Failed!");
    }

    sc.close();
  }
}
```