package TI;  
  
  
import java.math.BigInteger;  
import java.util.Random;  
import java.util.Scanner;  
  
public class RSA {  
  
 private static final int *bitLength* = 1024;  
 private static final Scanner *keyboard* = new Scanner(System.*in*);  
 private final Random random = new Random();  
 private Pair publicKey, privateKey;  
  
 private RSA(int length) {  
 BigInteger p, q, n, euler, e = new BigInteger("65537"); // 10000000000000001  
 do {  
 p = getPrime(length);  
 q = getPrime(length);  
 n = p.multiply(q); //2048  
 euler = eulerFunc(p, q);  
 } while (checkGCD(e,euler));  
 BigInteger d = e.modInverse(euler);  
 publicKey = new Pair(e,n);   
 privateKey = new Pair(d,n); //4096  
 }  
  
 private boolean checkGCD(BigInteger e, BigInteger euler) {  
 return euler.mod(e).equals(new BigInteger("0"));  
 }  
  
 private BigInteger eulerFunc(BigInteger p, BigInteger q) {  
 p = p.subtract(new BigInteger("1"));  
 q = q.subtract(new BigInteger("1"));  
 return p.multiply(q);  
 }  
  
 private BigInteger getPrime(int length) {  
 return BigInteger.*probablePrime*(length,random);  
 }  
  
 private BigInteger encrypt(BigInteger message) {  
 return message.modPow(publicKey.getFirst(), publicKey.getSecond());  
 }  
  
 private BigInteger decrypt(BigInteger message) {  
 return message.modPow(privateKey.getFirst(), privateKey.getSecond());  
 }  
  
 public static void main(String[] args) {  
 RSA key = new RSA(*bitLength*);  
 System.*out*.println("Enter message");  
 String message = *keyboard*.nextLine();  
 BigInteger messageBI = new BigInteger(message.getBytes());  
 BigInteger encrypted = key.encrypt(messageBI);  
 BigInteger decrypted = key.decrypt(encrypted);  
 String decryptedString = new String(decrypted.toByteArray());  
 System.*out*.println("Encrypted message: " + encrypted);  
 System.*out*.println("Decrypted message: " + decryptedString);  
 }  
}

package TI;  
  
import java.math.BigInteger;  
  
public class Pair {  
 private BigInteger first;  
 private BigInteger second;  
  
 public Pair(BigInteger \_first, BigInteger \_second) {  
 first = \_first;  
 second = \_second;  
 }  
  
 public BigInteger getSecond() {  
 return second;  
 }  
  
 public void setSecond(BigInteger second) {  
 this.second = second;  
 }  
  
 public BigInteger getFirst() {  
 return first;  
 }  
  
 public void setFirst(BigInteger first) {  
 this.first = first;  
 }  
}