

Mathematical-Calculation-Tools

项目更新优化于github，同学下载源码请前往：<https://github.com/KuroChan1998/Mathematical-Caculation-Tools>

- Mathematical-Calculation-Tools 是一个数值计算工具，功能包括整数域的运算(e.g 贝祖等式求解、勒让得符号、原根、素性检验....)；多项式的运算（e.g 贝祖等式求解、不可约多项式、本原多项式判断....）；加密算法（e.g. RSA）；椭圆曲线上的计算。
- 面向人群主要是上海交通大学信息安全专业修读《信息安全数学基础》课程的学生；其他网安专业学习数学理论基础的学生；抽象代数、应用数学领域学习的学生。
- 含图形界面
- 提供jar包，可作为api引用或在装有jre环境的机器上直接运行
- 这里酷乐酱用原生java实现，没有使用任何第三方api，算法原理全部参考陈恭亮老师编著的《信息安全数学基础》教材以及wiki，开源以供大家学习。

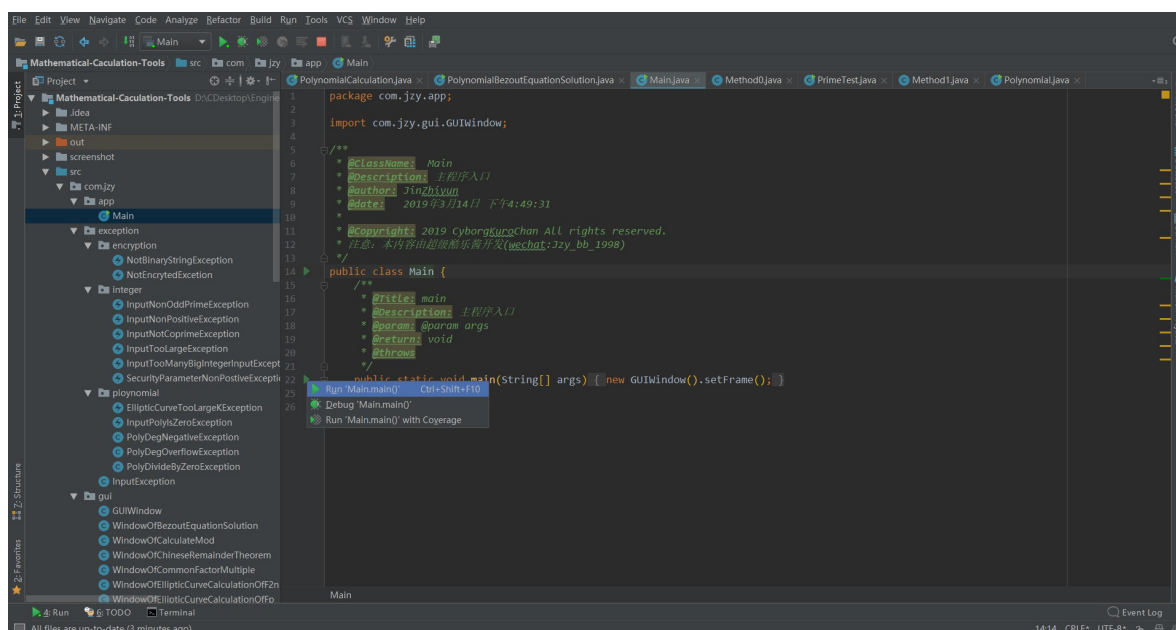
快速上手

使用jar包

在装有jre (>1.5) 的机器上直接双击 Mathematical-Caculation-Tools-1.0-SNAPSHOT-jar-with-dependencies.jar 可以直接得到图形界面

使用开发工具建立项目并运行

如果您装有jdk，以及idea、eclipse等开发环境和开发工具，直接导入该maven项目，找到com.jzy.app.Main.java文件直接运行。



My environment

- java
java version "1.8.0_211" Java(TM) SE Runtime Environment (build 1.8.0_211-b12) Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)

- IDE

IntelliJ IDEA 2018.1.7 (Ultimate Edition) Build #IU-181.5540.23, built on November 12, 2018
JRE: 1.8.0_152-release-1136-b43 amd64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
Windows 10 10.0

项目结构

```
└─java
  └─com
    └─jzy
      └─app      //主函数入口
      └─demo      //使用示例
      └─exception  //异常处理类根目录
        │ └─encryption  //加密类的异常
        │ └─integer    //整数域计算方法的异常
        │ └─polynomial  //多项式域计算方法的异常
      └─gui      //所有图形界面代码的根目录
      └─util      //工具包
      └─xxaqsxjc  //所有关键方法根目录
        └─encryption  //加密类根目录
          │ └─algorithm  //加密类的实现，这里实现了RSA、GoldwasserMicali、
Paillier
          │ └─executor  //加密类接口
          │ └─factory   //加密类工厂实现
          └─method0     //《信息安全数学基础》（1）（大二下）中涵盖的大部分运算的代
码实现
          └─method1     //《信息安全数学基础》（2）（大三上）中涵盖的大部分运算的代
码实现
```

如何使用代码？

这里省略对于图形界面的使用教程，正常脑壳的人都能上手。

- 代码中对您有用的api大部分都涵盖在了 `com.jzy.xxaqsxjc` 包下，所有代码都有详细的注释

这里列举关键方法概览和部分示例。更多的示例请参见 `com.jzy.demo` 包

com.jzy.xxaqsxjc.method0

此包主要是大二下《信息安全数学基础》课学习的知识点实现，您可以直接调用Method0.java中的静态方法，其涵盖了该包下大部分功能的api。使用示例如下：

```
import com.jzy.xxaqsxjc.method0.Method0;

import java.math.BigInteger;

public class Test {
    public static void main(String[] args) {
        BigInteger x=new BigInteger("100");
        BigInteger y=new BigInteger("120");
        //x, y的最大公因数
        System.out.println(Method0.maxCommonFactorXY(x,y));
    }
}
```

```

//x, y的贝祖等式求解
BigInteger [] r=Method0.bezoutSolveQrSt(x,y);
System.out.println("s="+r[0]+" , t="+r[1]);
//费马素性检验
BigInteger p=new BigInteger("912429886857661");
System.out.println(Method0.fermat(p));
//最小原根
p=new BigInteger("23");
System.out.println(Method0.minPrimitiveRoot(p));
    }
}

```

- CalculateMod.java: 计算大整数模
- CommonFactorMultiple.java: 最大公因数和最小公倍数计算
- BezoutEquationSolution.java: 贝祖等式系数求解
- EulerFuction.java: 欧拉函数值计算
- Legendre.java: 勒让得符号计算
- Jacobi.java: 雅可比符号计算
- PrimeTest.java: 素性检验, 集成了三种素性检验和暴力检验
- PrimitiveRoot.java: 原根计算
- ChineseRemainderTheorem.java: 中国剩余定理求解
- Method0.java: 该包下大部分方法的入口

com.jzy.xxaqsxjc.method1

此包主要是大三上《信息安全数学基础》课学习的知识点实现, 您可以直接调用Method1.java中的静态方法, 其涵盖了该包下大部分功能的api。使用示例如下:

```

import com.jzy.xxaqsxjc.method1.Method1;
import com.jzy.xxaqsxjc.method1.Polynomial;

public class Test {
    public static void main(String[] args) {
        //1+x+x^2+x^4
        int[] a = {1, 1, 1, 0, 1};
        //1+x^2+x^3+x^4+x^8
        int[] b = {1, 0, 1, 1, 1, 0, 0, 0, 1};
        Polynomial pa = new Polynomial(a);
        Polynomial pb = new Polynomial(b);
        System.out.println(pa);
        System.out.println(pb);
        //多项式计算
        System.out.println(pa.add(pb));
        System.out.println(pa.multiply(pb));
        System.out.println(pb.divide(pa));
        System.out.println(Polynomial.pow(pb, 10));
        //多项式最大公因式
        System.out.println(Method1.maxCommonFactor(pa,pb));
    }
}

```

- Polynomial.java: 多项式计算基础类
- PolynomialBezoutEquationSolution.java: 多项式贝祖等式系数求解
- PolynomialCalculation.java: 多项式其他一些相关计算

- EllipticCurveCalculationOfFp.java: Fp上的椭圆曲线点的计算

该方法不能通过Method1的静态方法调用，需要手工创建实例对象，使用实例如下：

```
import com.jzy.xxaqsxjc.method1.EllipticCurveCalculationOfFp;

import java.math.BigInteger;
import java.util.ArrayList;

public class Test {
    public static void main(String[] args) {
        //传入椭圆曲线参数，创建实例
        EllipticCurveCalculationOfFp eccfp = new
        EllipticCurveCalculationOfFp(new BigInteger("100823"), new BigInteger("3"),
        new BigInteger("7"));
        //点P1
        BigInteger[] p1 = {new BigInteger("5"), new BigInteger("101")};
        //计算P1、2P1、3P1、...kP1
        ArrayList<BigInteger[]> rs = eccfp.kPointSet(p1, 27);
        for (int i = 0; i < rs.size(); i++) {
            System.out.println("x" + (i + 1) + "=" + rs.get(i)[0]);
            System.out.println("y" + (i + 1) + "=" + rs.get(i)[1]);
            System.out.println();
        }
        //计算10P1
        System.out.println(eccfp.kPoint(p1, 10)[0]);
        System.out.println(eccfp.kPoint(p1, 10)[1]);
        //计算当前椭圆曲线的阶
        System.out.println(eccfp.ordFp());
    }
}
```

- EllipticCurveCalculationOfF2n.java: F2n上的椭圆曲线点的计算
使用类比EllipticCurveCalculationOfFp.java
- Method1.java: 该包下除椭圆曲线计算所有方法的入口

com.jzy.xxaqsxjc.encyption.algorithm

该包下提供三种加密算法，这里处于代码实现的简便，直接对明文逐个字符进行加密，使用实例如下：

- RSAEncryption.java: RSA加密算法
see more about RSA: [https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem)).
 - 方法一：通过有参构造器传入明文

```
import com.jzy.xxaqsxjc.encyption.algorithm.RSAEncryption;

public class Test {
    public static void main(String[] args) {
        RSAEncryption rsa1 = new RSAEncryption("我是明文");
        //      System.out.println(rsa1.encrypt()); //加密得到密文
        //      System.out.println(rsa1.decrypt()); //解密得到明文
        //这里通过show()方法直观展示加密情况，也可以使用encrypt()单独输出密文等等
        rsa1.show();
    }
}
```

```
随机生成1024位大素数p=1762550049275813061830880206094716628553378183078296078719418440500155005471188109165892
随机生成1024位大素数q=1524657411969234373826901292112828122518381404166789173254119325193790527444007256558056
2048位公钥n=p*q=268728499659510766111946942508407918234406705116888128353603519697731364876983653349803552962
随机生成公钥e=1946235712721224933478595098157403251530390953991803472962338522235051296438196037226162482650
进而生成私钥d=19861689311841265878905415385145714474522113591518360739811226164908880792900949051562539960875
加密得到密文: G,v
解密密文得到的明文: 我是明文
```

- 方法二：通过无参构造器创建对象，通过setPlainText方法传入明文

```
import com.jzy.xxaqsxjc.encyption.algorithm.RSAEncryption;

public class Test {
    public static void main(String[] args) {
        RSAEncryption rsa1 = new RSAEncryption();
        rsa1.setPlainText("我是明文");
        rsa1.show();
    }
}
```

- 方法三：通过工厂获得加密类实例（单例），再通过setPlainText方法传入明文

```
import com.jzy.xxaqsxjc.encyption.EncryptionAlgorithm;
import com.jzy.xxaqsxjc.encyption.algorithm.RSAEncryption;
import com.jzy.xxaqsxjc.encyption.factory.EncryptionFactory;

public class Test {
    public static void main(String[] args) {
        //传入枚举参数RSA，从工厂获得实例
        RSAEncryption rsa1 = (RSAEncryption)
        EncryptionFactory.getEncryption(EncryptionAlgorithm.RSA);
        rsa1.setPlainText("我是明文");
        rsa1.show();
    }
}
```

- 其他：重置密钥

默认密钥在编译代码时确定，默认位宽1024bit。可以通过resetKeys()静态方法重置（指定位宽）密钥

```
import com.jzy.xxaqsxjc.encyption.EncryptionAlgorithm;
import com.jzy.xxaqsxjc.encyption.algorithm.RSAEncryption;
import com.jzy.xxaqsxjc.encyption.factory.EncryptionFactory;

public class Test {
    public static void main(String[] args) {
```

```

        //传入枚举参数RSA，从工厂获得实例
        RSAEncryption rsa1 = (RSAEncryption)
EncryptionFactory.getEncryption(EncryptionAlgorithm.RSA);
        rsa1.setPlainText("我是明文");
        rsa1.show();
        //重置密钥位宽512bit
        RSAEncryption.resetKeys(512);
        rsa1.setPlainText("使用512bit密钥加密，我是明文");
        rsa1.show();
    }
}

```

```

随机生成1024位大素数p=13537171519182940149040475644409222749745575338544928664623093
随机生成1024位大素数q=10010242189445639153181354926493471070842137815820153175907491
2048位公钥n=p*q=1355103654670869839412663586101511870960299619697024578616766087727
随机生成公钥e=1310895768176814300084956385120836713884167020059602257343990914563286
进而生成私钥d=1327019676714930587357207475728921097421406080774206158690930129692112
加密得到密文: `47U
解密密文得到的明文: 我是明文
随机生成512位大素数p=123617486806143465904763139247636813340512036531765403503926976
随机生成512位大素数q=124429986663865901371607298042886630839244618172478744192876975
1024位公钥n=p*q=1538172223470905048037435934091102153241982371659746370917537827620
随机生成公钥e=7962891840414472763878796601498415233342329333650419973174828514266191
进而生成私钥d=1096131561620637123585663316083028524887431627196873168048667817445060
加密得到密文: txC-X)V'TzVTOr<I|
解密密文得到的明文: 使用512bit密钥加密，我是明文

```

- GoldwasserMicaliBinaryEncryption.java：GoldwasserMicali二进制串加密算法

see more about GoldwasserMicali :https://en.wikipedia.org/wiki/Goldwasser-Micali_cryptosystem

使用类比RSA加密

- PaillierEncryption.java：Paillier加密算法

see more about Paillier : https://en.wikipedia.org/wiki/Paillier_cryptosystem

使用类比RSA加密

联系方式

- qq: 929703621
- wechat: Jzy_bb_1998
- e-mail: 929703621@qq.com
- github: <https://github.com/jinzhiyun1998/Mathematical-Caculation-Tools>

欢迎提出意见与建议~