Overview: This project is composed of two parts. This is phase 1. It is due on Friday, November 15th at 11:59pm. Thereafter, you will get one more phase. Read the comments in the code. These comments will give you hints on how to create the code you need.

Github: Part of this assignment is to create a gitub account (if you don't have one already) and clone this project to your github account. Use your github account to backup your code. When you turn in your project, make sure you have committed the most recent version of your code to your github account. This github account will be used to validate your assignment. Don't forget to TURN IN your project by the due date on eCampus.

Set your github to private. This is at the bottom of the "Options" Set "Glen-TAMU" as a collaborator, so I can verify your work. Githubs NOT set to "private" will lose points.

Intro to Github: https://guides.github.com/activities/hello-world/

Cloning a repo: https://help.github.com/en/articles/cloning-a-repository

Coding: Your job is to extend the functionality of the provided code. Do not delete or alter any of the given code. Your code must have the same return values and function prototypes as what is provided, as well as the same level of information hiding and encapsulation.

The object of your program is to generate three things: A list of primes, a list of Twin Primes from those primes, and a list of "Hexagon Crosses" from those Twin Primes. You will be working with very large numbers and will need arbitrary precision (look up what arbitrary precision is), so you can not use "int" or "double." Instead, you will use a special class from the Java mathematics library called "BigInteger" for all your numbers.

A "twin prime" is a pair of primes that only have one integer between them, such as 5 and 7; 11 and 13; 6269 and 6271; or 12539 and 12541.

A "hexagon cross" is a pair of numbers that are the middle integer between the primes in a twin prime, such that the second number in the pair is twice the first number. For example, 6 and 12 make up a hexagon cross, because 6 is the integer between the twin prime 5 and 7, and 12 is the middle number between the twin prime 11 and 13, and 6 * 2 = 12. (6,270 and 12540 are another hexagon cross.)

You will have to design algorithms to find primes, twin primes, and hexagon crosses.

You will notice that there are a lot of pairs in this project. You will also notice that Java surprisingly doesn't have a "pair" class. (You may not use Javafx.) You will have to implement a templated "Pair" class as a private class that is a member of the Primes class. It should implement all needed pair functionality through member functions.

Refactoring: Refactoring is a very powerful tool in modern IDEs. Right click on the code with want to refactor, select "refactor" from the context menu, and perform the following tasks:

- In Project1.java, rename the variable “p” to “testOne”.
- In Project1.java, rename the variable “testtwo” to “testTwo”.
- Rename the Primes class to “PrimeOperations”

Testing: You will want to test your code to ensure it is both correct and highly functional. You will want to generate sufficient test cases and validate your code to ensure it completes all code paths successfully.

Style: In programming, it is very important to use a specific and consistent programming style. Use the refactor tool to rename class names as per the Google Java Style Guide: https://google.github.io/styleguide/javaguide.html When in doubt, use this style guide.

Notes: Indents in these files are set to 2 spaces. You can change this setting in your Eclipse file by clicking Window Preferences. -> Expand Java Code Style ->Click Formatter -> Click the Edit button -> Click the Indentation tab. Under General Settings, set Tab policy to Spaces only, set spaces to 2.

You are going to need to load files in this app. Eclipse, however, is going to be picky about where the “current directory” is. We could solve this by using absolute paths, but that means you may run into problems when you are trying to find that location on your system, which may be laid out differently than my system. So we will instead set up the Eclipse working directory

Go to the "run configuration", open the "Arguments" (second) tab, and click the "Other" radio button. Then enter an absolute pathname as the working directory for the launched application. Click Apply.

Hints:

1. It is normal to temporarily comment out code so you can test the functionality of your code piecemeal. Just remember to uncomment your code before turning it in.
2. It is normal to add debugging information to your code during development. Remember to remove this debugging code before you turn your project in.
3. One of the key aspects of Java is platform independence. Make sure you never hard-code your program to only work on one operating system or in one specific computer.
4. Always check the online Java documentation. Many things you want to do are already included as classes in java. Import and use them rather than reinventing the wheel.
5. Make sure you start a new project in eclipse for this assignment. A common mistake people make is to import the code into an existing project and then have trouble separating old code from the assignment.

What You Turn In:

1. The completed .java files in a ZIP file on eCampus.
2. Your github repo address. Include this at the top of your text file. Remember to set your repo private and add Glen-TAMU as a collaborator.

Grading: Grading will be based upon completeness and correctness. If your code is fully functional and passes all tests while adhering to the design guidelines, you get full points for this section. If your

code is partially functional, you will get partial credit. If your code does not compile or run, you will get minimal credit. If you turn nothing in, you will get no credit.

Academic Honesty: As with most programming project, student code will be subjected to extensive anti-cheating monitoring. Algorithm design will be checked, as well as code layout.

Future Phases and Late Work: A solution to the first phase will be provided, as well as new start files for part 2, so that a student who does poorly in phase one is not starting at a disadvantage in the following section. Students may continue to use their own code, or they may use the provided start files for each subsequent phase.

Because solutions will be provided, late assignments **will not** be accepted.

Test Output:

```
1
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
Total Primes: 21
3, 5
5, 7
11, 13
17, 19
29, 31
41, 43
59, 61
71, 73
```

101, 103
107, 109
137, 139
149, 151
179, 181
191, 193
197, 199
227, 229
239, 241
269, 271
281, 283
311, 313
347, 349
419, 421
431, 433
461, 463
521, 523
Total Twins: 25
Prime Pairs: 5, 7 and 11, 13 separated by 6, 12
Prime Pairs: 29, 31 and 59, 61 separated by 30, 60
Prime Pairs: 659, 661 and 1319, 1321 separated by 660, 1320
Prime Pairs: 809, 811 and 1619, 1621 separated by 810, 1620
Prime Pairs: 2129, 2131 and 4259, 4261 separated by 2130, 4260
Prime Pairs: 2549, 2551 and 5099, 5101 separated by 2550, 5100
Prime Pairs: 3329, 3331 and 6659, 6661 separated by 3330, 6660
Prime Pairs: 3389, 3391 and 6779, 6781 separated by 3390, 6780
Prime Pairs: 5849, 5851 and 11699, 11701 separated by 5850, 11700
Prime Pairs: 6269, 6271 and 12539, 12541 separated by 6270, 12540
Total Hexes: 10