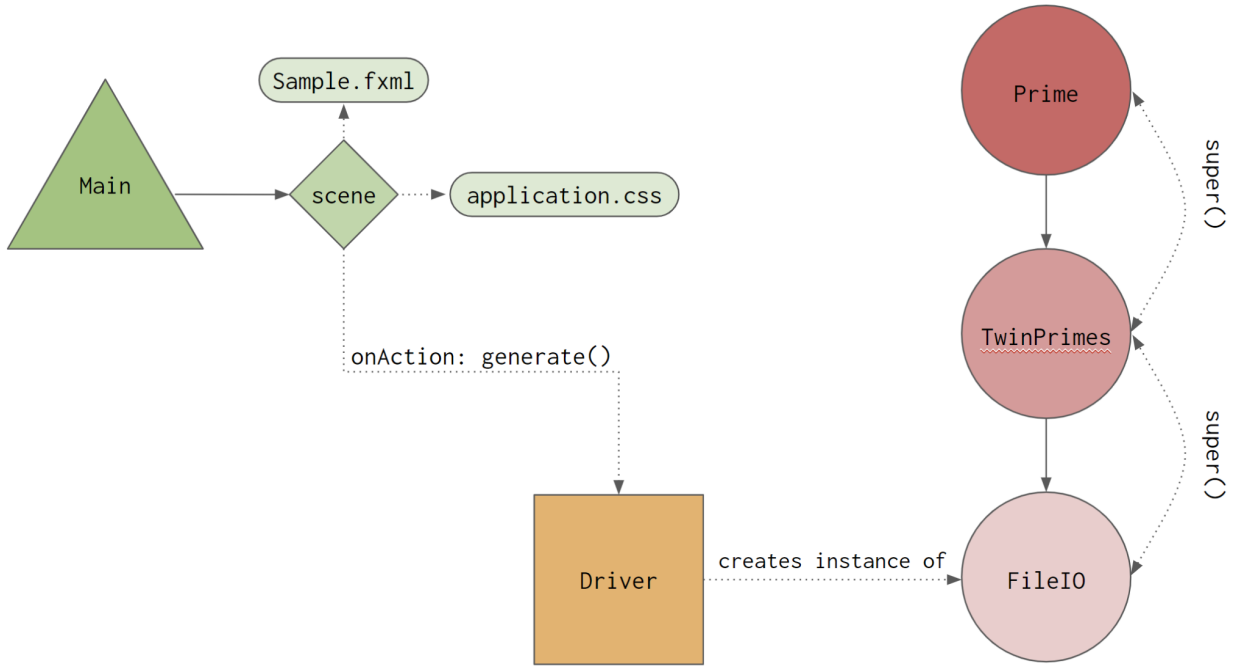
Quick Guide to Twin Primes (1.0.0)

Developer: Esha Adhawade

Files: Driver.java, Prime.java, TwinPrimes.java, FileIO.java, Main.java

Class Schematic Diagram:



Description of Each File:

1. **Driver.java**
   - The purpose of the Driver class is to gather user input from the GUI and display the results using an instance of FileIO once the "generate" button is clicked.

   a. *Attributes:*
      - public int n_pairs;
      - public String file_location;
      - private static final String DEFAULT_LOC = “C:\\Results”.
      - private static final int DEFAULT_PAIRS = 10;

   b. *public void generate(ActionEvent e)*
      - This function collects the n_pairs and file_location from the text fields. It then creates a new instance of FileIO using the n_pairs and file_location (either default or user specified) to gatherHexagonalCrosses() and get the toString() to write to the display area to display the results.

2. **Prime.java**
    - The purpose of the Driver class is to gather user input from the GUI and display the results using an instance of FileIO once the "generate" button is clicked.

    a. *Attributes:*
        - public int[] next_prime_pair;

    b. *public void findNextPrimePair(int prev_prime)*
        - Consider the previous prime pair was {3, 5}. The function findNextPrimePair() uses the larger of the previous pair as the previous prime number.It then uses this number to find the next prime number (i.e. 7). It then continues to find the prime number after that (i.e. 11). The array next_prime_pair is populated with these two numbers (i.e. {7, 11, _}). The third spot in next_prime_pair is to indicate whether or not it is a twin prime pair. If the difference between the pair is 2, it then puts the first prime number plus 1 as the number between the twin prime. Otherwise, it puts -1, indicating it is not a twin prime.

    c. *private static boolean isPrime(int num)*
        - This is a helper function for findNextPrimePair(). Given a number this returns whether or not the number is prime using a simple algorithm. If, for any number i in [2, num-1], the num is divisible by i, it is prime since it is not only divisible by 1 and the num itself.

    d. *public int[] getNextPrimePair()*
        - This function is the getter and returns the current instance of the attribute next_prime_pair;

3. **TwinPrimes.java**
    - This file is TwinPrimes and is the child class of Prime and the parent class of FileIO. The purpose of this file is to find all the twin prime pairs and the hexagonal_crosses for the requested number of pairs.

    a. *Attributes:*
        - public int N; (the number of hexagonal crosses requested)
        - public List<int []> hexagonal_crosses (list of hexagonal crosses)

    b. *public void findHexagonalCrosses()*
        - The purpose of findHexagonalCrosses() is to populate the hexagonal_crosses list with valid hexagonal crosses. First, we use a hash map called twin_pairs to store the value in between the twin pairs as the key and the actual pair of twin primes as the value (i.e. [4 : {3,5}, ...]). Although we don't use the twin pair primes, it is convenient to have a key-value pair in the

case we would want the actual twin prime pairs. Next, we use a while loop to go through all N pairs the user requested. First, we findNextPrimePair from the Prime class. We then getNextPrimePair() and extract the key. If the key is not equal to -1, we insert it into the dictionary. We then check if the keySet contains half of the current value of the key. If so, we add it to hexagonal_crosses and then decrement N. We set prev_prime and continue until N is equal to 0.

c. *public List<int []> getHexagonalCrosses()*
   - This is a getter function that returns the current instance of the list of hexagonal_crosses.

4. **FileIO.java**
   - The file is the child class of TwinPrimes and grandchild class of Prime. The purpose of this file is to collect all the hexagonal crosses from TwinPrimes and write it to a file called "results.txt". The user can view the hexagonal crosses in “results.txt” wherever they chose to store it (or, for the default location: C:\Results). They can also receive the string version of the result through the toString() function.

   a. *Attributes:*
      - public String file_location
      - private List<int []> hexagonal_crosses
      - private static final String FILENAME = “results.txt”

   b. *public void gatherHexagonalCrosses()*
      - The purpose of gatherHexagonalCrosses() is to find the hexagonal crosses from TwinPrimes and write it into a file. We write the hexagonal crosses to a file called "results.txt". The user can also give a file_location in which the function, gatherHexagonalCrosses(), will create the file if it doesn't already exist. If the user doesn't have a preferred location, the default location is in "C:\Results" (defined in Driver class). In any case, the function can create a new directory if it doesn't exist and the new "results.txt" file to write the hexagonal crosses to.

   c. *private static String getPath(String file_loc)*
      - This function adds a backslash to the end of the file_location if it does not already exist. This ensures that we can properly create the path to write to the "results.txt" file.

   d. *public String toString()*
      - This is a toString() function that returns a string of the hexagonal crosses. The “@Override” above it is because the toString() signature can be used in several classes, so it helps to extract a warning in the compiler.

5. **Main.java**
    - In this file, a root is created to read from the Sample.fxml file that contains the GUI built on SceneBuilder. Then a scene is created by using application.css as a resource to build and display the GUI. In essence, the Main.java file creates and launches the GUI so that the user can interact with the program as intended.