

Lab #7—Prime Numbers

Objectives:

- Introduce prime numbers.
- Use arrays and methods.
- Introduce algorithms for solving various problems with primes.

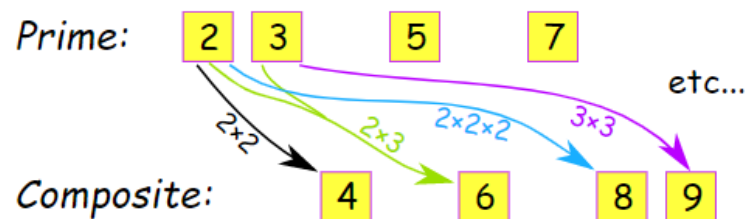
A **Prime Number** is:

a whole number that **cannot** be made by multiplying other whole numbers

(if we **can** make it by multiplying other whole numbers it is a **Composite Number**)

And 1 is not prime and also not composite.

Here we see it in action:



2 is Prime, 3 is Prime, 4 is Composite ($=2 \times 2$), 5 is Prime, and so on...

Here is a list of all the prime numbers up to 1,000:

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73
79	83	89	97	101	103	107	109	113	127	131	137	139	149	151	157	163	167	173	179	181
191	193	197	199	211	223	227	229	233	239	241	251	257	263	269	271	277	281	283	293	307
311	313	317	331	337	347	349	353	359	367	373	379	383	389	397	401	409	419	421	431	433
439	443	449	457	461	463	467	479	487	491	499	503	509	521	523	541	547	557	563	569	571
577	587	593	599	601	607	613	617	619	631	641	643	647	653	659	661	673	677	683	691	701
709	719	727	733	739	743	751	757	761	769	773	787	797	809	811	821	823	827	829	839	853
857	859	863	877	881	883	887	907	911	919	929	937	941	947	953	967	971	977	983	991	997

Problem Statement I: Write a Java program that calculates and displays user specified number of prime numbers. The prime numbers are whole numbers greater than one divisible only by itself and one, e.g.,
 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97... .

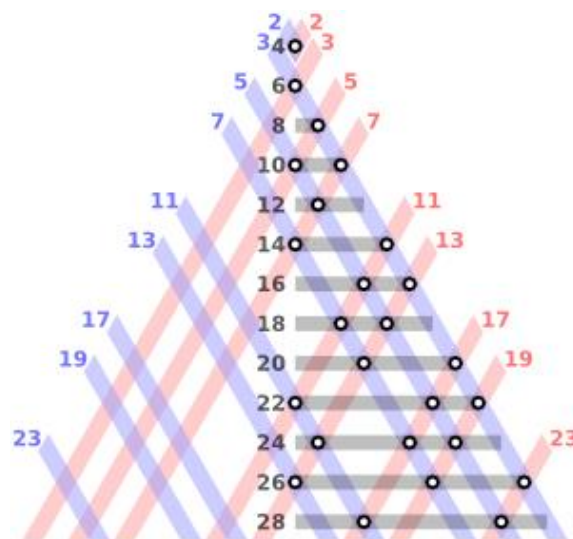
Problem Statement II: Add to the program above so that the user inputs a positive integer and the program check to see if it is a prime number and reports so. If it is a composite (not a prime), the program displays the prime factorization of that number. For example, if the user inputs 277, the program responds with 277 is a prime. If the user inputs 95, the program responds with $95 = 5 * 19$ or $100 = 2*2*5*5$.

Problem Statement III: Add to the program the find prime factorization of any composite (non-prime) number provided by the user. In number theory, the fundamental theorem of arithmetic, also called the unique factorization theorem or the unique-prime-factorization theorem, states that every integer greater than 1 either is a prime number itself or can be represented as the product of prime numbers and that, moreover, this representation is unique, up to (except for) the order of the factors,

Problem Statement IV: Add to the program above so that the user enters two integers and the program displays all primes in that range. For example, if the user inputs 100 and 120, the program should output 101, 103, 107, 109, and 113.

Problem Statement V: Add to the program above the ability to solve Golbach's conjecture. Goldbach's conjecture is one of the oldest and best-known unsolved problems in number theory and all of mathematics. It states: Every even integer greater than 2 can be expressed as the sum of two primes.

Problem Statement VI: Add to the program above to display palindromic primes. A palindromic prime is a prime number and also palindromic (the number remains the same when its digits are reversed). For example, 131 is a prime and also a palindromic prime, as are 313 and 757.



Problem Statement VII: Add to the program above to display emirps. An emirp (prime spelled backward) is a nonpalindromic prime number whose reversal is also a prime. For example, 17 is a prime and 71 is a prime, so 17 and 71 are emirps. Neither are palindromes. The first several emirps are: 13, 17, 31, 37, 71, 73, 79, 97, 107, 113, 149, 157, 167, 179, 199,....

Problem Statement VIII: Add to the program above to display user specified number of Sophie Germain primes. In number theory, a prime number p is a Sophie Germain

prime if $2p + 1$ is also prime. Sophie Germain primes are named after French mathematician Sophie Germain, who used them in her investigations of Fermat's Last Theorem. Sophie Germain primes and safe primes have applications in public key cryptography and primality testing. It has been conjectured that there are infinitely many Sophie Germain primes, but this remains unproven. The first several Sophie Germain primes are:
2, 3, 5, 11, 23, 29, 41, 53, 83, 89, 113, 131, 173, 179, 191, 233, 239, 251, 281, 293, 359,...

Problem Statement IX: Add to the program above to display user specified number of safe primes. In number theory, a prime number p is a safe prime if $2p + 1$ is also prime. The first several safe primes are: 5, 7, 11, 23, 47, 59, 83, 107, 167, 179, 227, 263, 347, 359, 383, 467,

Problem Statement X: Add to the program above to display user specified number of Fermat numbers (but no more than 6?). In mathematics a Fermat number, named after Pierre de Fermat who first studied them, is a positive integer of the form

$$F_n = 2^{2^n} + 1$$

where n is a nonnegative integer. The first few Fermat numbers are:

$$3, 5, 17, 257, 65537, 4294967297, 18446744073709551617, \dots$$

Fermat numbers and Fermat primes were first studied by Pierre de Fermat, who conjectured (but admitted he could not prove) that all Fermat numbers are prime. Indeed, the first five Fermat numbers F_0, \dots, F_4 are easily shown to be prime. However, the conjecture was refuted by Leonhard Euler in 1732 when he showed that

$$F_5 = 2^{2^5} + 1 = 2^{32} + 1 = 4294967297 = 641 \times 6700417$$

There are no other known Fermat primes F_n with $n > 4$. However, little is known about Fermat numbers with large n . In fact, each of the following is an open problem:

- Is F_n composite for all $n > 4$?
- Are there infinitely many Fermat primes?
- Are there infinitely many composite Fermat numbers?
- Does a Fermat number exist that is not square-free (an integer which is divisible by no perfect square other than 1. For example, $10 = 2 \cdot 5$ is square-free, but $18 = 2 \cdot 3 \cdot 3$ is not, because 18 is divisible by $9 = 3^2$). The smallest positive square-free numbers are

$$1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 33, 34, 35, 37, 38, 39)?$$

Prime Numbers

File Action Help

Prime Number Action

☒ How many prime numbers?

☐ Is this a prime number?

☐ Find prime factorization of?

☐ Range of prime numbers
 From To

☐ Goldbach's conjecture

☐ How many palindromic primes?

☐ How many emirp primes?

☐ How many S. Germain primes?

☐ How many safe primes?

The first 20 prime numbers are:

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71

Calculate Clear Print Quit

Prime Numbers

File Action Help

Prime Number Action

☐ How many prime numbers?

☐ Is this a prime number?

☒ Find prime factorization of?

☐ Range of prime numbers
 From To

☐ Goldbach's conjecture

☐ How many palindromic primes?


☐ How many emirp primes?

☐ How many S. Germain primes?

☐ How many safe primes?

3857 = 7*19*29

Calculate Clear Print Quit

 Prime Numbers

File Action Help


Prime Number Action

☐ How many prime numbers?
☐ Is this a prime number?
☐ Find prime factorization of?
☐ Range of prime numbers
 From To
☒ Goldbach's conjecture
☐ How many palindromic primes?
☐ How many emirp primes?
☐ How many S. Germain primes?
☐ How many safe primes?

The combinations of prime numbers whose sum is 48 are:

$48 = 5 + 43$
 $48 = 7 + 41$
 $48 = 11 + 37$
 $48 = 17 + 31$
 $48 = 19 + 29$

Calculate Clear Print Quit

 Prime Numbers

File Action Help


Prime Number Action

☐ How many prime numbers?
☐ Is this a prime number?
☐ Find prime factorization of?
☐ Range of prime numbers
 From To
☐ Goldbach's conjecture
☒ How many palindromic primes?
☐ How many emirp primes?
☐ How many S. Germain primes?
☐ How many safe primes?

The first 20 number of palindromic prime numbers whose reverses are primes are:

2	3	5	7	11
101	131	151	181	191
313	353	373	383	727
757	787	797	919	929

Calculate Clear Print Quit

 Prime Numbers

File Action Help


Prime Number Action

☐ How many prime numbers?
☐ Is this a prime number?
☐ Find prime factorization of?
☐ Range of prime numbers
 From To
☐ Goldbach's conjecture
☐ How many palindromic primes?
☒ How many emirp primes?
☐ How many Germain primes?
☐ How many safe primes?

The first 30 emirps
= a nonpalindromic prime numbers whose
reverses are primes are:

13	17	31	37	71
73	79	97	107	113
149	157	167	179	199
311	337	347	359	389
701	709	733	739	743
751	761	769	907	937

Calculate Clear Print Quit

 Prime Numbers

File Action Help

Prime Number Action

☐ How many prime numbers?
☐ Is this a prime number?
☐ Find prime factorization of?
☐ Range of prime numbers
 From To
☐ Goldbach's conjecture
☐ How many palindromic primes?
☐ How many emirp primes?
☒ How many Germain primes?
☐ How many safe primes?

The first 30 Sophie Germain primes
= a prime $2p + 1$ is also prime are:

2	3	5	11	23
29	41	53	83	89
113	131	173	179	191
233	239	251	281	293
359	419	431	443	491
509	593	641	653	659

Calculate Clear Print Quit

Guidelines and hints:

- Construct the GUI class PrimeNumbers in NetBeans as in Lab 6—it's a JFrame with BorderLayout. It contains 3 JPanels and a JMenuBar:
 - choiceJPanel in West with a Free Design Layout that contain all the radio buttons and JTextFields. Note that initially all JTextFields are disabled except for the first one that corresponds to the initially selected radio button How many prime numbers. As the user selects another radio button, its corresponding JTextField becomes enabled and the focus is set to it while the other JTextField is cleared and becomes disabled.
 - controlJPanel in South with a GridLayout (4 columns and 1 row) that contains the four JButtons.
 - displayJPanel in Center with a Free Design Layout that contain displayJScrollPane (to provide vertical and horizontal scroll bars) and a displayTextArea inside of it for the display of results.
- The project contains a choicebuttonGroup that goes under Other Components because it is not a visible component. It's needed in order to group all radio buttons (set each one's ButtonGroup property to the choicebuttonGroup). Without this step, the radio button are not mutually exclusive and behave like check boxes in that you can select multiple of them. Note also that the first radio button is selected at design time and its corresponding JTextField is enabled.
- Each of the first four radio buttons have an event handler that enables the corresponding JTextField and sets the focus to it if the radio button is checked. If the radio button is unchecked (because another is checked), the corresponding JTextField is cleared and disabled. Here is the code for one of them:

```
private void listAllJRadioButtonItemStateChanged(java.awt.event.ItemEvent evt)
{
    // Enable corresponding text field
    if (listAllJRadioButton.isSelected())
    {
        whichRadio = 1;
        listAllJTextField.setEditable(true);
        listAllJTextField.requestFocus();
    }
    else
    {
        listAllJTextField.setText("");
        listAllJTextField.setEditable(false);
        displayTextArea.setText("");
    }
}
```

- Selecting the otherJRadioButton should enable the otherJTextField, clear it, and requestFocus to place the cursor inside of it.

5. All of the action occurs in the `calculateJButtonActionPerformed` event handler. In it call a method `getButton()` which determines and returns which radio button was selected and then perform one of four operations in a switch statement:

```
int whichButton = getButton(); // decide which radio button is selected
switch(whichButton)
{
    case 1:          // the first n prime numbers
        break;
    case 2:          // yes or no on prime for a given number
        break;
    case 3:          // prime factorization
        break;
    case 4:          // range of prime numbers
        break;
    case 5:          // Goldbach conjecture
        break;
    case 6:          // palindromic primes
        break;
    case 7:          // emirps
        break;
    case 8:          // Sophie Germain primes
        break;
    case 9:          // safe primes
        break;
}
```

6. Here is the code for the `getButton` method:

```
// method to return integer for which radio button is selected
private int getButton()
{
    if (listAllJRadioButton.isSelected())
        return 1;          // first radio button selected
    else if (isPrimeJRadioButton.isSelected())
        return 2;          // second radio button selected
    else if (factorizationJRadioButton.isSelected())
        return 3;          // third radio button selected
    else if (rangeJRadioButton.isSelected())
        return 4;          // fourth radio button selected
    else if (goldbachJRadioButton.isSelected())
        return 5;
    else if (palindromicJRadioButton.isSelected())
        return 6;
    else if (emirpJRadioButton.isSelected())
        return 7;
    else if (sophieGermainJRadioButton.isSelected())
        return 8;
    else
        return 9;
}
```


7. Consider a boolean method `isPrime()` which determines if an integer is a prime number. The number 2 is the only even prime, so you need consider the odd integers 3 and thereafter for “primeness” (a natural number is a prime if it is > 1 and whose only factors are 1 and itself).
8. Add method `isPalindrome()` to check if a number is palindromic (the number in reverse is also a prime).
9. Add method `reverse()` to reverse the digits of a number and return it as integer.
10. Consider saving the prime numbers in an array:

```
int max = Integer.parseInt(listAllJTextField.getText());
if (max < 2 || max > MAX_INPUT)
    throw new NumberFormatException();
// allocate dynamically array for primes
int[] primes = new int[max];
primes[0] = 2;           // 2 is special
int count = 1;          // only 1 prime so far
int trialNumber = 1;    //start with trial number = 1
boolean isPrime = false;
StringBuilder output = new StringBuilder("");           // contains output
```

11. Consider adding a Splash screen (with a progress bar) that starts the program and closes itself after several seconds and an About form that pops up from the Help menu.
12. Add the ability to print the form (use provided `PrintUtilities` class) as well as the output in the `JTextArea`.
13. Add a detailed About form (`JDialog` with type `Utility`).
14. Redesign the lab to include a `Primes` class that contains all 10 algorithms. Each should take argument for how many primes (or min/max for range) and return the formatted `StringBuilder` output.
15. Here is a detailed list with names of all components:

