**CSE 174 – Fall 2016**

**PROGRAM #10: 40 points – Due Sunday, November 13, by 11:59 p.m.**

**Outcomes:**

* Write programs that obtain user input
* Write programs that use loops to perform calculations and validate input
* Write programs that use if and else statements
* Write programs that display numbers formatted according to a given specification
* Format and comment source code that adheres to a given set of formatting guidelines

**Scoring:**

At a bare minimum, the program you submit must have the assigned source code, and your source code must compile and run without crashing.

* If you do not submit a zip file containing your source code (.java file), your score will be zero.
* If you submit source code that does not compile, your score will be zero.
* If you submit source code that roughly resembles the requirements and it compiles, but it crashes under normal operating conditions (nice input from the user), your score will be reduced by 75%.
* Deductions will be made for not meeting the usual requirements:
* Source code that is not formatted according to guidelines
* File and class names that do not meet specifications

|  |  |  |
| --- | --- | --- |
|  | **Full credit** | **No credit or Partial credit** |
| **Use a menu to obtain user input (7 points)** | Your menu allows the user to choose any valid option, but does not allow any invalid integer values. | There are errors in the way the menu handles valid and/or invalid input. |
| **Check if a number is prime (5 points)** | Your program correctly indicates whether a valid integer is prime. It only allows valid integer values from the user. | There are errors in the way the menu option handles valid and/or invalid input. |
| **Factor a number**  **(8 points)** | Your program correctly factors valid integers. It only allows valid integer values from the user. | There are errors in the way the menu option handles valid and/or invalid input. |
| **List prime numbers**  **(12 points)** | Your program correctly lists prime numbers when the user enters valid input. | There are errors in the way the menu option handles valid and/or invalid input. |
| **Decompose a problem into methods (4 points)** | Your program uses short methods to break up a larger programming problem. The main method is fewer than 30 lines. | You have methods that should have been broken into smaller methods. |
| **Format console output**  **(4 points)** | You formatted output as specified, including aligning prime numbers. | Screen output does not match specifications. |

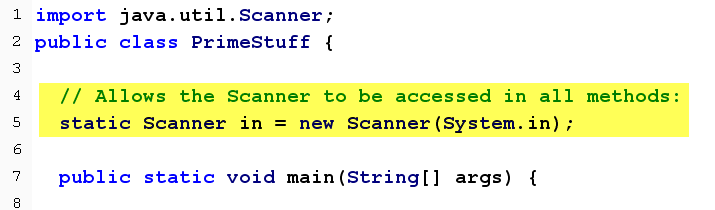
**Preliminaries:**

* Review the definition of a prime number (any positive integer with exactly two factors). The first few prime numbers are 2, 3, 5, and 7. Note that 1 is not a prime number by this definition.
* Review how to format numbers to fit a particular width using printf().

**Assignment requirements:**

Write a program named PrimeStuff that presents the user with a menu of options related to prime numbers. The user should be able to work with prime numbers up to and including the maximum value for Java's **long** data type. Whenever the user enters input, make sure that their numbers are valid, using a loop to prevent them from entering invalid numbers. Match the formatting as close as possible to what is shown in the sample run. Some special notes:

* **Using a Scanner that works in multiple methods:** Do not create a separate Scanner object in several methods. Instead, your program should have one Scanner object that is shared by all the methods that need it. In order to do this, you will create a **global** Scanner object as shown. This creates it inside the class but outside of the methods:



* **Factoring numbers:** Factor the number into prime numbers, from smallest to largest. For example, 36 = 2 \* 2 \* 3 \* 3. Note that if a number is prime, then it's prime factorization is itself. So, if the user wants to factor the number 83 which is prime, it would look like this: 83 = 83. Note that 1 is not prime. If the user wants to factor 1, just display 1 = 1.
* **Listing prime numbers:**  This should let the user list as many prime numbers as she wants (up to 1000 numbers), starting at whichever positive integer she wants, with any number of numbers per row (up to 20 numbers per row). Note that if the user's starting number is a prime number, then the list should begin with that prime number. If the user's starting number is not prime, then the list should begin with the first prime after that number. When listing numbers in multiple rows, the numbers should be right-aligned over one another. See the sample run for examples.
* **Validating input:** You should never allow the user to try to use an illegal value. You may assume that the user will always enter an integer, but you should use a loop to make sure that the integer is always within the appropriate range. You may assume that if the user wants to list primes, she will always use a starting value that will not cause the list to exceed the maximum long value. See sample run for other limits.
* **Use short methods:** No method should contain more than 30 lines of code. Most methods can be written in 15 or fewer lines of code.
* **Avoid code duplication:** As much as possible, do not perform a calculation in more than one place. For example, there should only be one place in your code where you check to see if a number is prime.
* **Submitting:** Whenever you make progress (reach a new milestone), submit a draft. The class name should be PrimeStuff. It should be the only file in a folder named program10. Zip the program10 file and submit the resulting .zip file.

**Sample run:**

**Your program should match this format as closely as possible. Note that text shown in red is there because the user typed it. You are not supposed to print those.**

|  |
| --- |
| What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **0**  \*\*\*\* INVALID OPTION \*\*\*\*  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **8**  \*\*\*\* INVALID OPTION \*\*\*\*  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **1**  Enter a number between 1 and 9223372036854775807: **-6**  Enter a number between 1 and 9223372036854775807: **1001**  ---> 1001 is not prime.  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **2**  Enter a number between 1 and 9223372036854775807: **1001**  ---> 1001 = 7 \* 11 \* 13  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **1**  Enter a number between 1 and 9223372036854775807: **89**  ---> 89 is prime.  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **2**  Enter a number between 1 and 9223372036854775807: **89**  ---> 89 = 89  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **2**  Enter a number between 1 and 9223372036854775807: **256**  ---> 256 = 2 \* 2 \* 2 \* 2 \* 2 \* 2 \* 2 \* 2  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **2**  Enter a number between 1 and 9223372036854775807: **1**  ---> 1 = 1  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **3**  Enter starting number between 1 and 9223372036854775807: **755**  How many primes (1-1000): **0**  How many primes (1-1000): **1001**  How many primes (1-1000): **25**  How many primes per row (1-20): **0**  How many primes per row (1-20): **21**  How many primes per row (1-20): **8**  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  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **3**  Enter starting number between 1 and 9223372036854775807: **15677**  How many primes (1-1000): **100**  How many primes per row (1-20): **10**  15679 15683 15727 15731 15733 15737 15739 15749 15761 15767  15773 15787 15791 15797 15803 15809 15817 15823 15859 15877  15881 15887 15889 15901 15907 15913 15919 15923 15937 15959  15971 15973 15991 16001 16007 16033 16057 16061 16063 16067  16069 16073 16087 16091 16097 16103 16111 16127 16139 16141  16183 16187 16189 16193 16217 16223 16229 16231 16249 16253  16267 16273 16301 16319 16333 16339 16349 16361 16363 16369  16381 16411 16417 16421 16427 16433 16447 16451 16453 16477  16481 16487 16493 16519 16529 16547 16553 16561 16567 16573  16603 16607 16619 16631 16633 16649 16651 16657 16661 16673  What would you like to do?  1) Check if a number is prime  2) Factor a number  3) List prime numbers  4) Quit  Choice: **4** |