|  |  |
| --- | --- |
| **30 minutes** | **Task 1: Times Tables** |
| This task will iterate over, and print, the values of a times table. The times table being displayed should come from the user as an input.   1. Ask the user for a value using the Scanner class and sc.nextInt() 2. Given the value the user has inputted, iterate 10 times in a loop and display the times-table for that number up to 12x that number   STRETCH CHALLENGE: Change the program to ask the user for how many iterations they would like to see for the number they are wanting to find out the times table for. For example, the user enters 22 and 50. The program should show the 22 times table for the first 50 values.  STRETCH CHALLENGE: Change the program so that the user enters both values from the previous stretch challenge in one input. You will need to use .nextLine() instead of .nextInt(), a .split() on the inputted values, and you will need to cast the strings into numerical values too. | |

|  |  |
| --- | --- |
| **30 minutes** | **Task 2: Finding a Leap Year** |
| This task will have you working on loops once more. The task will be to identify if a year is a leap year or not   1. Create a for-loop that will use an iterator (i) starting at 1900. 2. Use the escape clause within the loop to be when i reaches 2030 3. Within the loop, identify if the current iterator’s value is a leap year. A leap year is a year that is:    1. Divisible by 4    2. NOT divisible by 100 BUT is divisible by 400 4. Print each year that is a leap year to the console.   STRETCH CHALLENGE: Change your if statements to Ternary Operators | |

|  |  |
| --- | --- |
| **30 minutes** | **Task 3: Fibonacci Numbers** |
| The Fibonacci sequence is found throughout nature and is an important aspect for understanding the nature of things.  The sequence works by adding the previous two numbers in the sequence to get the next number. For example:   * 1 + 1 = 2 * 1 + 2 = 3 * 3 + 2 = 5 * 5 + 3 = 8 * 8 + 5 = 13, etc  1. Create a while-loop that will continue to execute while a variable called ‘fib’ is less than 1 million 2. On each iteration of the loop, provide the next number in the Fibonacci sequence | |

|  |  |
| --- | --- |
| **30 minutes** | **Task 4: Guessing Game** |
| Take a look at the following article for random numbers: <https://www.educative.io/answers/how-to-generate-random-numbers-in-java>   1. Have the program generate a random number using one of the methods above (I suggest method 2) 2. Within a loop, and until the user guesses the right number:    1. Ask the user for a value    2. If the user number is higher, print out a statement saying the number is too higher    3. If the user number is lower, print out a statement saying the number is too low   STRETCH CHALLENGE: Change the method of generating a random number – what are the benefits/drawbacks of using other random generating methods? | |

|  |  |
| --- | --- |
| **2 hours** | **Task 5: Visual Sequences** |
| This task is specifically vague!! There are many ways to solve coding challenges, so be as creative as you need to be to solve this task:  Write a program for each of the six tasks. **Using loops**, print the following:  STRETCH CHALLENGE: Change the program(s) so that it asks the user for a numerical value. Continue the loops for the number of times the user has requested. 3, 4, 5, and 6 will prove to be tricky for this stretch challenge, as formatting is important! | |

## **Marking Criteria Task 1/3/4**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Syntax** | * Attempts to use Java syntax with some success | * Java syntax is largely accurate with some errors | * Java syntax is consistently accurate and appropriate to the task |
| **Presentation** | * Some whitespace used to good effect * Indentation attempted but inaccurate * Correct naming convention | * Whitespace used appropriately, at times * Indentation largely accurate | * Clear commentary provided throughout * Code is explicitly clean and easy to read |

## **Marking criteria Task 1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempts to use a loop | * Asks the user for a value correctly * Successfully uses the correct loop | * Manages to complete both stretch challenges |

## **Marking criteria Task 2**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempts to use a loop | * Successfully uses the correct loop | * Successfully implements ternary operators instead of if-statements |

## **Marking criteria Task3**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempts to use a loop | * Successfully uses the correct loop | * Provides the correct Fibonacci sequence |

## **Marking criteria Task 4**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempts to use a loop | * Successfully implements a random number * Successfully implements the correct loop | * Completes the task as described – a working guessing game |

## **Marking criteria Task 5**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Pass** | **Merit** | **Distinction** |
| **Code** | * Attempts to use a loop for some of the tasks | * Successfully implements some of the task loops | * Successfully implements all loops, displaying properly (3, 4, 5, and 6 formatted correctly) |