

CST8116 Assignment 01 (24W)

Software Development Process, design and create an Object-Oriented Java Program

Instructions

Any assignment submission which does not include the required .java source code files will receive a maximum grade of 3/20

Any assignment submission which is not object oriented (all input/output/computations are in main and there is no object class for the bacteria calculation) will receive a maximum grade of 6/20

You must use the Java API Math class in your program to compute a value using an exponent. If the Math class is not used: -4 deduction.

You must use printf to format the final number of bacteria to 2 decimal places. If printf is not used, -1 deduction.

- The Software Development Process by Cay Horstmann [1] will be used as the basis for this lab assignment.
 - 1) Understand the problem
 - 2) Develop and Describe an Algorithm
 - 3) Test Algorithm with Simple Inputs
 - 4) Translate the Algorithm into Java
 - 5) Compile and Test Your Program

Learning Resources

Week 2 Lecture regarding classes Math, Scanner

Week 3 Lecture materials regarding creating a class, with variables, constructors, get & set methods, worker method(s) Hybrid 3 for more on UML Class diagrams, and using the UMLet software.

Hybrid 4 for information on using printf to format numeric output.

See [1] for more help with Java, and [2] for more help with pseudocode and UML Class Diagrams

What is the problem to solve?

<https://www.algebra.com/algebra/homework/logarithm/Bacteria-growth-problems.lesson>

You have a petri dish of bacteria with 5 bacteria in it. The bacteria doubles every 15 minutes. Given a particular time, you must estimate how many bacteria there are.

- Create an Object-Oriented Java Program that calculates the bacteria in the dish after the entered time in minutes and seconds
- The program should output the number of bacteria formatted to 2 decimal places.
- Minimally, you will create two classes, one to represent or model a petri dish, and another with a main method. The main method will get user input, create a petri dish object, and make method calls on that object. The class that models the petri dish must perform the estimate of bacteria calculation and returns its value to the main method, which will output the value to the user.

Hint: you may want to convert all time to seconds before doing the Bacteria computations. For example, 66 minutes and 3 seconds = $66 \times 60 + 3 = 3963$ seconds.

1) Understand the problem

- Locate at least one website that details the exponential formula needed to calculate the growth of bacteria other than the one above. Write the formula into your MS Word document, on one line as plain-text, and cite and reference your source in IEEE reference format.
- Write a formula that will perform the required calculation. (it must use exponent math ie. 2^t)
- Write 3 sample calculations. I.e. How many bacteria are produced after
 - 5 minutes?
 - 20 minutes and 30 seconds?
 - 1 minute?

(Tip: See lecture notes week 1 for a document to help with citations and references).

2) Develop and Describe an Algorithm including Classes (MS-Word Document Submission)

2a) UML Class Diagrams

- Determine what objects, the properties of the objects, and behaviors would be required for an object-oriented program, start with simple UML class diagrams and as you work out more details document the design with a detailed UML class diagram.
- Include your detailed UML Class diagrams in your MS Word document. (See lecture notes week 3).
- One class diagram should document field variables, constructor(s), getter(s) and setter(s) and one worker method.
- One class diagram should document a class that only contains method main.

2b) Pseudocode and Flowchart (MS-Word Document Submission)

- NOTE: You are expected to use at least one method in the Java API Math. ~~class and one field value from the Java API Math class.~~
- Write pseudocode and flowchart for the main method, as well as any worker method(s) that calculate how much bacteria is produced.
- Include your pseudocode and flowchart in your MS Word document.
- Note: You are not required to write pseudocode or flowcharts for getter(s) or setter(s) or constructor(s) for this assignment, the expectation would be the main method and one worker method.

3) Algorithm Test Table (MS-Word Document Submission)

- As per the lecture notes, use a table within your MS Word document to test the algorithm for method main. Consider picking numbers that might be expected as input and work through the algorithm documenting expected outputs, you may use a calculator.
- You must have 5 rows!
- If there is a problem with the algorithm based on this desk-check (using the test plan) then correct the pseudocode and UML class diagram(s) and repeat this step again.

4) Translate the Algorithm into Java

- You are to use the Eclipse IDE to create your Java program, use a project name like Assignment 01.
- Don't forget to comment your code files, with the expected code headers
 - File-level comments, similar to an assignment cover page.
 - Class-header comment, just above the class declaration with brief description of the class
 - Method-header comment(s), just above each method or constructor with brief description
 - Cite and Reference the web source where you located the formula for the volume of a cone.

5)Program Test Table (MS-Word Document Submission)

- Re-create your testing table from step 3 in this section, but document what the program outputs are using valid inputs, do they match expectations?
- Test with some invalid inputs, and document what happens. Note that some input will crash your program, this is okay as you may not know how to fix this at this point in the course. Document what the error messages are in your test plan. Suggested invalid tests: enter a String instead of a number, use a negative number as input, use zero as part of the input. Try some things and document what happens.
- **The test plan for the program must be a separate table in your document, if you provide only one test plan table for both testing the algorithm and testing the program you will not be awarded marks for both. Minimally, this second table must also have each row updated to indicate that it was used to test the program, in other words if you copy and paste an exact copy of your first table unchanged you will not earn full marks.**

(4 & 5) Java Program Screen Shot

- Include a screen shot of your program after it has run, place the screen shot into your MS Word document.
- The screen shot should show your full name, as part of the program output in the Eclipse Console Window.
- The program should accept inputs, provide outputs, and complete successfully.
 - Even if the program does not fully work, part marks are available for a screen shot.

Microsoft Word Document: Suggested Headings

Understand the problem

UML Class Diagram(s)

Pseudocode

Before Test Table

After Test Table

Screen Shot of Program Execution

References

Submission Requirements

- You will need to submit your MS Word document and your Java source code file(s)
- Follow your lab professor's submission guidelines.
- You are not required to copy and paste Java code into the MS Word document, however your .java file(s) must be submitted as Java source code files along side your MS Word document.

Grading (22 points)

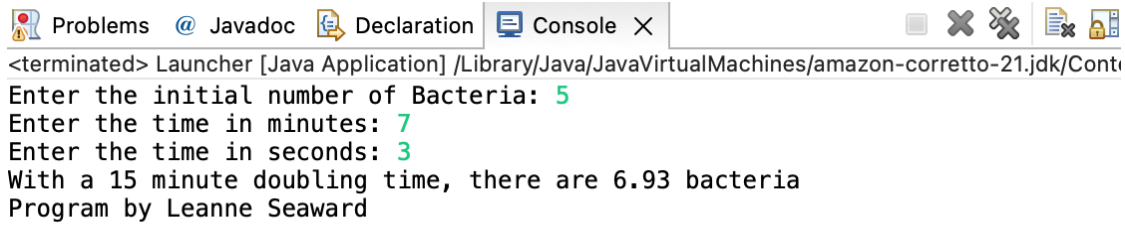
Criteria	Missing / Poor (0)	Below Expectations (1)	Meets Expectations (2)
Understand the problem	Missing or poorly done.	Problem solution statement is partly correct, may not have examples of the calculations required to solve the problem.	Problem solution statement is correct. Sample calculations needed to solve the problem are provided. Includes cited and referenced source.
Algorithm: UML class diagram	Missing or poorly done.	Class diagram(s) are not in correct format, properties and methods may not be assigned correctly to the classes and /	Class diagram(s) are correct format, properties and methods are assigned to appropriate classes, based on the word problem.

		or there is only one class that contains all of the program functionality.	
Algorithm: pseudocode	Missing or poorly done.	Not in correct format and / or steps are not in an order that produces correct results. Not all worker method(s) and / or method main are documented.	Correct format, steps are in order that produces correct results. Worker method(s) and method main are documented.
Algorithm: flowcharts	Missing or poorly done.	Not in correct format and / or steps are not in an order that produces correct results. Not all worker method(s) and / or method main are documented.	Correct format, steps are in order that produces correct results. Worker method(s) and method main are documented.
Test Plan: Algorithm	Missing or poorly done.	Does not have correct table format as seen in lecture notes and lab exercises, and / or does not test program using samples of acceptable inputs. Less than 5 rows.	Has correct table format as seen in lecture and lab exercises, tests program using samples of acceptable inputs.
Test Plan: Program	Missing or poorly done or unchanged copy of algorithm test table.	Does not have correct table format as seen in lecture notes and lab exercises, and / or does not test program using samples of acceptable inputs and / or does not document program errors resulting from invalid inputs. Less than 5 rows	Has correct table format as seen in lecture and lab exercises, tests program using samples of acceptable inputs, and documents program errors resulting from invalid inputs.
Source Code: *.java file(s) Comments and Conventions	Missing or poorly done.	File comment header with student name is present. Class and / or class-member (constructors, methods) are missing comment headers. Loosely follows Java coding conventions for identifiers, indentation.	File comment header with student full name is present. Class and / or class-member (constructors, methods) have comment headers. Closely follows Java coding conventions for identifiers, indentation.
Source Code: *.java file(s) program structure and logic.	Missing or poorly done.	Program may have small syntax mistakes or produces incorrect output. Program may consist entirely within method main (is not object-oriented).	Program has correct syntax and program logic that produces correct output. Program is object-oriented with classes, fields, get / set methods, worker methods.
Running Program *.java file(s) Program Structure and Logic	Student files will not compile and run on professor's computer due to syntax mistakes in the student's source code.	Program compiles and runs; however, program does not work correctly with valid input value(s) as used by lab professor to test program.	Program compiles and runs and program does work correctly with valid input value(s) as used by lab professor to test program.
Screen Shot of Program Execution in Eclipse Console Window	Missing or poorly done (missing student name in screen shot)	Program runs however may not be shown to work correctly. May only have part of the student's name in the screen shot.	Shows the program running successfully with correct outputs, based on inputs. Student's full name is visible in the screen shot.
Submission	Missing	Student does not provide both MS Word and .java file(s) with their submission, and/or does not follow lab professor's submission requirements.	Student does provide both the MS Word document and .java file(s) with their submission, and does follow lab professor's submission requirements.

References

- [1] Cay Horstmann. (2019). Big Java Early Objects. 7th Ed. Wiley.
- [2] Joyce Farrell. (2018). Programming Logic & Design Comprehensive. 9th Ed. Cengage Learning.

Appendix: Sample of Program Running



The screenshot shows an IDE interface with a 'Console' tab selected. The console output displays the program's execution flow, including user input for initial bacteria count, time in minutes, and time in seconds, followed by a calculated result and the program author's name.

```
<terminated> Launcher [Java Application] /Library/Java/JavaVirtualMachines/amazon-corretto-21.jdk/Cont  
Enter the initial number of Bacteria: 5  
Enter the time in minutes: 7  
Enter the time in seconds: 3  
With a 15 minute doubling time, there are 6.93 bacteria  
Program by Leanne Seaward
```