Name_____ CSC 210 Java II
Due: 23 November 2025 Fall 2025

Lab 09: Streaming Data

Setting Up

1. Download the zip file for lab-09 on Brightspace.

2. Unzip the file to a nice working directory where you usually do your work

3. Enter the unzipped directory. Consider loading it in Visual Studio Code, Eclipse, Intellij, or the terminal

4. Review the `overview-lab-09.pdf` to make sure you understand the concepts of population growth using the logistic map for both discrete and continuous population models as well as for what happens to these models when more than two species are added to the system. You need not be incredibly proficient with differential equations and their calculations. It is entirely normal to build programs for systems with information inside of them that you don't entirely understand, but if you are familiar, feel free to enjoy working through the equations.

Exploring the Code Examples

5. Review the program CircleDisplay.java. Compile and run it. Make sure you understand how the constructor and the paintComponent method are used by the main function to make the program work.
  a. What is CircleDisplay() doing?
  b. When/how often does paintComponent() run?
  c. What is the JPanel class from which CircleDisplay inherits? Look up its official Java documentation online.
  d. Alter the color and size of the circle and compile and run the program.

6. Review the program ContinuousLogisticGrowth. Compile and run it. Adjust the initial values of the formula to learn how this continuous model behaves in contrast to the discrete model. You may use the program InteractiveLogisticGrowth.java to make iterating on changing the initial values faster.
  a. What term makes this function different from the traditional exponential growth function?
  b. When you adjust the initial values, do you find the behavior of this growth model to be unpredictable and chaotic like its discrete cousin?
  c. Does the population always converge to a value?
  d. Does the population ever collapse?
  e. Does the population ever hit a run away exponential growth explosion to infinity?
  f. Compare what you see in this program with what is output by the program RabbitDiscreteSimulationGUI.java

7. Compile and run the SierpinskiTriangle.java program.
  a. How is the program able to draw the same pattern at ever smaller scales over and over again.
  b. What function is responsible for the shape of any particular triangle (not painting)?
  c. What function is responsible for the shape of the entire image?

d. In what way is the recursive nature of drawing this triangle similar to the discrete logistic map function?

8. Looks at the programs RandomHexPrinter.java and RawRandomHexStreamer.java. Compile and run each in order. You may need to press CTRL-C to stop the second program from running.
a. What are these programs doing?
b. How are these program getting their randomness?
c. Is the randomness truly random?
d. Based on what you see here, write a simple program named GetWebPage.java that returns www.java.com

9. Compile and run the program CarrotGenerator.java. Review the code.
a. Are the random numbers generated by this program random or pseudorandom?
b. What do the random values get used as in the Carrot model?
c. How often does the timer execute?
d. What code executes when the timer goes off?
e. What variable keeps track of all of the Carrots?

10. Compile and run the RabbitsMover.java program. Review the code.
a. When a rabbit is created in this program, what determines its starting location?
b. What line of code initializes the rabbit's velocity?
c. How many times per second are the positions of rabbits updated to the screen?
d. What function updates a rabbit's position?
e. How do the rabbits know to bounce off the walls? Temporarily take out the code that does this and compile and run the program to prove that you are right.
f. Adjust the program so that the balls move really fast or really slow. Compile and run it.

11. Compile and run the WolfMover.java program. Review the code.
a. How often does the program update the position of the wolf? What code determines this?
b. In the constructor of WolfMover, what code enables keyboard input to be captured?
c. What method does the timer execute on WolfMover every time the timer goes off?
d. Change the code in the following way:
- make the wolf faster
- make the wolf bigger
- change the color of the wolf
- invert the controls so that pressing the left arrow key goes right and the left arrow key goes left, up goes down, and down goes up.
- recompile and run the program.

12. Run the programs SpriteLoader.java, SpriteLoader2.java, and SpriteAnimationPanel.java. Review the code.
a. Notice how the image is loaded in SpriteLoader.java. What is the type of the variable that holds the image data?
b. Where is the image that is loaded and what is it a picture of?
c. In SpriteLoader2.java notice the code that runs to actually paint the sprite on the screen. What is the type of the variable that is actually loaded onto the window to be painted?

d. Change the name of the file being loaded in the second program to “resources/spriteSheet.png” and recompile and run the program. What do you see?

e. Notice when running the SpriteAnimationPanel.java program that the image is animated. What is different about this program that causes the spritesheet to be animated rather than loaded all at once as it was in part 12d?

13. Now compile and run the MultispeciesContinuousSimulation.java program. Notice that there are now 3 graphs for 3 separate creature populations. Play around with the initial values and notice how the graph changes. You can use the program MultispeciesSimulationUI.java to more easily manipulate and see the affects of the initial values. Notice that each function grows exponentially until it starts hitting a limit. Also notice how they always converge and how adjusting the initial values of one organism sometimes has an unexpected affect on another organism as a result of some multi-step consequence that you can probably reason through intuitively.

a. If this program is graphing continuous equations, why is it not using integration and differentiation to get its values? What is it using instead?

b. Based on your reading of the overview documentation handed out with the lab, what does each initial value represent in the simulation?

c. Why are the curves so smooth?

14. Run the program CarrotRabbitSimulation.java. Watch the graph over several iterations of carrot re-population.

a. In this continuous agentic simulation, what is the main limiting factor on the maximum size of the rabbit population?

b. Does this simulation more closely follow the discrete or continuous logistic map? Why?

c. How often does the carrot population re-populate and what is the maximum number of carrots?

d. Adjust the frequency of the carrot repopulation so that it happens every second or so. Watch the graph again. Does it now more closely match the discrete or continuous logistic map function?

e. Notice that there is an artificial factor named maxPop in the program limiting the rabbit population. Comment out this formula and set the variable to a really high number that the population would never reach given the number of carrots. Run the simulation again with a high number of carrots (200-ish) and a very short carrot repopulation time. Does the simulation cause the rabbit population to converge, change chaotically, tend to infinity, or go extinct? Why?

f. Do the same with the program ArcadeEcosystemSimulation.java and see if you can adjust the parameters similarly to match the continuous counterpart to the logistic map function for the multispecies graphs. See how closely you can get the agentic simulation to match the predictions of the graphs in the program MultispeciesSimulationUI.java. What values minimized the difference between the two simulations?

15. Refactor the code in `ArcadeGame.java` program so that you have a project inside your lab directory in your repo with the following class files:

a. `InputHandler.java`
b. `Wolf.java`
c. `Carrot.java`
d. `Rabbit.java`
e. `Ecosystem.java`
f. Add a fourth creature named Fungus that eats wolves and carrots
g. Give the wolves and rabbits spritesheets that animate