Lab 2 - Drawing, Methods, and Iterative Enhancement
(Train)
CS 161 - Fall 2016

Due date: 10pm, Tuesday, September 13th

Goals

- to use iterative enhancement (start small and grow the program)
- to build methods both without and with parameters
- to consider the sequence of actions needed to solve a problem

Overview

Your Java program created for this assignment will draw a sideways ASCII art train similar to the one shown at right. The train has five cars total: an engine, a caboose, and three cars in the middle. The engine and caboose are just letters and wheels. For the three middle cars, each car has the same shape but is drawn with characters specified by the user. The train will look like the picture and be vertical on the screen.

You will start with a small simple program and enhance it until it has all the features you need. Use the Quilt.java program as an example. Create a single car method with two parameters to draw one car and call it three times with different parameters.

Input Specification: What Input Will Be

Each of the middle cars has a pair of input characters: an edge and a stripe. The edge character is the car's border. The stripe character is decoration. The other features don't change from car to car.

Use six separate type String variables to hold the six input values. Each String will hold just one character for this program (don't check that). Prompt for and read the characters before starting to print your train. You can use input.next( ) to read a String, where input is a Scanner. Order of input is important. The train picture shown results from this prompt and input (user types E s # $ = %)

Please enter six characters for drawing train (separate them with spaces)
Order is edge stripe edge stripe edge stripe
E s # $ %

Output Specification: What Output Should Be

The output will look like the example picture. The edges of each car should be drawn with the user-specified edge character with the edge being the predominant character. The stripe should be drawn with the stripe character. It should be an accent or decoration. You are free to be creative with the engine and caboose. The engine and caboose must exist and must be drawn with methods, but you will not lose points for having just boxes with wheels for the engine and caboose. Drawing the cars in the shape shown is acceptable, but you may be more elaborate. There should be connectors between the cars.

Note that this is console input and output with no display panels.

Getting Started

Think about what you know how to do (create a new Java project, copy and paste, print a prompt, declare string variables, get input, print variables). Think about what parts need to be in your train (engine, caboose, three cars, connectors).

Think about what small part you know how to do. Start with that, get it to work, and then enhance it.

(Additional guidance below.)
Create a New Project in Eclipse by selecting File > New > Java Project. Type Lab2 as the Project Name. Make sure that Lab2 is selected under the Package Explorer pane on the left side of the window.

Create a new Java class file by clicking on the New Java Class button. Make sure that Lab2 is the source folder name (make it so, if it isn’t). Type Train as the Name. Under "Which method stubs would you like to create?”, choose to create "public static void main(String[] args)”. Click the Finish button.

Edit Train.java to be a very simple program solving a small part of the problem. What things do you know to add? Your name, program description, date? Can you ask the user for two Strings for edge1 and stripe1? Can you store those values in variables and print them? You are welcome to copy/paste parts from the Quilt.java example. Or you can type parts from there. Don’t do too much at once. Get something very simple written.

Compile and execute your program. Does it do what you expect it to do? Why or why not?

Approaching the Rest of the Problem

Now that you have something small working, think about the overall problem. The program reads input and prints the train. Printing the train is a sequence of printing each car top to bottom.

Sketching on paper might help you plan your solution. Always have paper beside you.

Develop your solution incrementally. We suggest the following iterations although there are many ways to approach this problem. For each iteration, modify your Train.java source code, save, compile, and execute. Don’t do too much in each iteration.

- Create methods to print the engine, caboose, and one car. They might just print the method name. Would a connector method be handy? Include a comment on each method saying what it does.
- Consider how this problem is like the quilt program. Are the middle cars like the quilt? The car method takes in two Strings as formal parameters (e and s for edge and stripe?) and uses them to print a simple version of a car.
- Define four more input variables, get those values from user, and use those in other calls to the car method
- Enhance car method to match picture
- Enhance engine and caboose methods. They may take parameters if you wish.

You absolutely must use methods for this assignment. There should be only one car method, and it must take two String parameters specified by the user. Call the car method three times with the appropriate different variables as actual parameters. You must have a method to draw the engine and the caboose, but they may print boxes with wheels.

Getting Finished

Test your program with different input. Be sure that you are reading the input in the correct order (edge stripe edge stripe edge stripe). Be sure that your cars are in the correct order with connectors between them.

Did you include a comment at the beginning of each method? Is your name and a description of the program at the top? Did you use good variable names?

Does your indenting match the program logic? You can press Control+Shift+F in Eclipse to format your program. Eclipse may be confused by some program errors such as missing or mismatched parentheses, brackets, or quote marks.

In the terminal, move to your 161 > Lab2 directory so that you can see your Train.java program. Use ls -l to see that it is the current version.

Check that it is what you want by compiling it again with javac Train.java and checking that it runs with java Train

Submit it to my katz161 account as the Train lab. Note that the name to use for submitting is usually at the top of the lab handout. It is also the Java class name. When you submit, always be in the directory where you compile and execute your program.