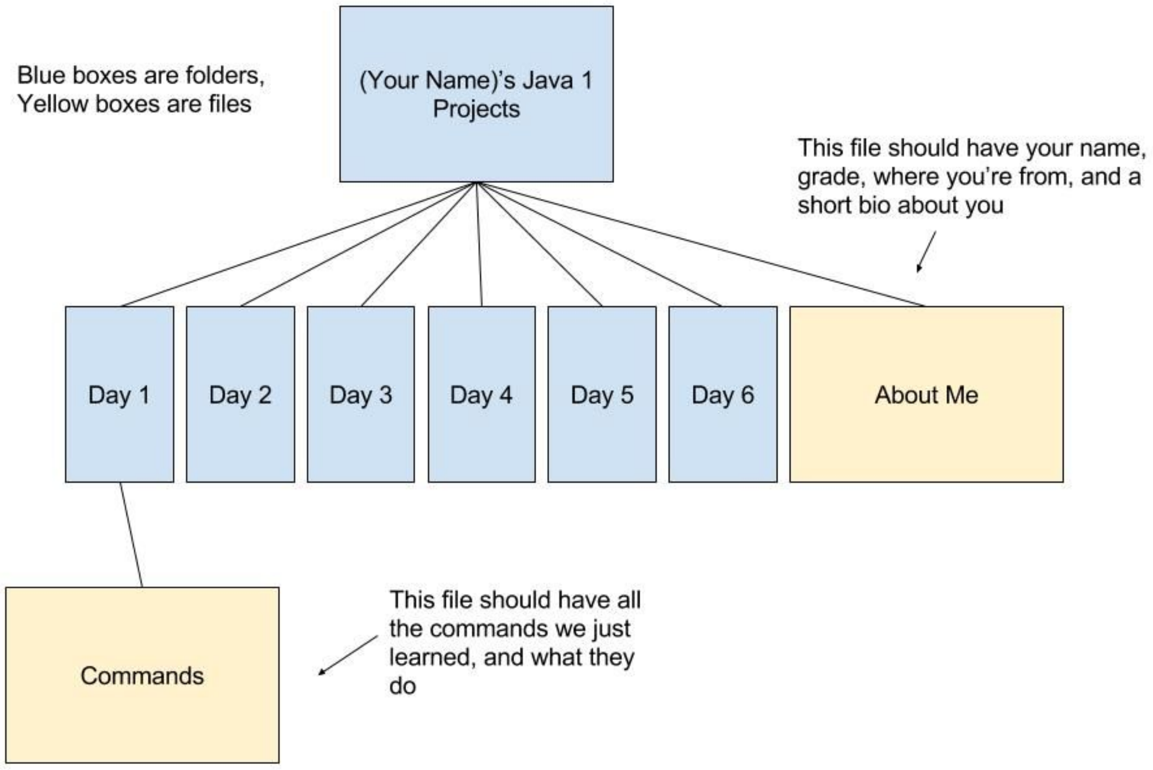
Blue boxes are folders,
Yellow boxes are files
(Your Name)'s Java 1 Projects
This file should have your name, grade, where you're from, and a short bio about you
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
About Me
Commands
This file should have all the commands we just learned, and what they do

Assignment 1: Printing

1. You are given the AboutMe.java class
2. Edit the project so that when it is run, it introduces you by printing your name, age, hometown, hobbies and passions, and favorite color.
3. Now, print those things on at least 3 different lines! You will need more than one "System.out.println" statements for this.
4. Save the project to Git

Assignment 2: Variables

1. You are given the Person.java class
2. Create the following variables:
    a. An int called age which has the value of your age.
    b. A double called height which has the value of your height.
    c. A String called firstName which has the value of your first name.
    d. A String called lastName which has the value of your last name.
    e. A boolean called isAStudent which is true if you are a student and false if you are not a student.
3. Make the program print all of this information using the variables. For example, when you want to print your name, do not do:
    a. System.out.println("My name is Alix Feinsod");

    Instead, do:

    b. System.out.println("My name is" + firstName + " " + lastName);
4. Add 2 more variables of your choice with 2 different types that represent something about you. Then print those variables as well.
5. Run the program and make sure the output works as expected.
6. Save the program in Git.

Assignment 3: Math Operations

1. You are given the MathOperations.java class
2. First, read through and run the program. Is the output what you expected?
3. Change the variable ‘together’ so that it says “peanut butter and jelly” (there are several ways to do this).
4. Make at least 5 new variables using all of the operations (addition, subtraction, multiplication, division) on the existing variables at least twice each. Print those new variables as the others are printed in the program.
5. Modify at least 2 variables using the shortcuts: +=, -=, ++, --. Print the new values.
6. Run the program and make sure the output works as expected.
7. Save the program in Git.

Assignment 4: Booleans

1. You are given the Booleans.java class
2. Read through the current code and run it. Make sure the output makes sense to you.
3. Copy and paste the existing code from line 10 to line 23. Change the values of x and y, and predict what the new outcomes will be.
4. Add comments showing the six operators and meanings. Example: // < means less than.
5. Create three new String variables, named s1 with value “Good morning.”, s2 with value “Hi there!” and s3 with value “Howdy, partner!”. Then, write 3 new print statements comparing the Strings. Use the .equals method once and the .compareTo method 2 times.
6. Create two new boolean variables, one named b1 which is true, and one named b2 which is false. Write 5 new print statements using those variables and a combination of and (&&), or (&&) and not (!). Example: System.out.println(“True and not true: ” + (b1 && !b1));
7. Run the program and make sure the output works as expected.
8. Save the program in Git.

Assignment 5: Scanner

1. You are given the UserData.java class
2. Read through the current code and run it. Make sure the output makes sense to you.
3. Store the name in a String variable called ‘name’.
4. Next, use the name to make a personalized prompt for the age, which you should store in an int variable. For example, if the name was Alix, the next line should print something like “Hi, Alix! How old are you?”.
5. Continue prompting and storing the following information from the user. Always use Strings and the nextLine input, but convert to the following types:
    a. Birthday (day as an int, month as a String, year as an int)
    b. Favorite Book (as a String)
    c. Hometown (as a String)
    d. Lucky number (as a double)
    e. Shoe size (as a double)
    f. On a scale of 1 to 5, how excited they are to learn Java (as a double)
    g. One cool fact about them (as a String)
6. Run the program and make sure the dialogue works as expected.
7. Next, have the program print out all the information in the following format:

    Alix Feinsod, age 22, was born on November 22, 1994. From Berkeley, Alix has lucky number 7 and a shoe size of 7.5. On a scale of 1 to 5, Alix is this excited to learn about Java: 10! A fun fact about Alix is: I love cats!
8. Run the program and make sure the output works as expected.
9. Now, we will do some math. Create new variables, using the existing ones, to represent the following information:
    a. An int that is their age in 5 years from now
    b. A boolean that is true if their age is equal to their birthday (day of the month they were born)
    c. A boolean that is true if their birth year divided by 2000 is greater than 1, which will mean they were born in or after the year 2000
    d. A double that is their shoe size mutliplied by their excitement level
10. Have all of the new information printed as well, with explanations of each one.
11. Run the program and make sure the output works as expected.
12. Save the program in Git.

Assignment 6: Conditionals

1. You are given the Quiz.java class
2. Read through the current code and run it. Make sure the code and the quiz make sense to you.
3. Add 5 new questions to the quiz. They can be multiple choice or true/false, with at least one of each. One question should be about math, and one should be about science, and one should be about Java programming. The rest can be whatever you want! Make sure you keep track of the number of right answers and print them at the end.
4. Run the program and make sure the quiz works as expected.
5. Save the program in Git.
6. Trade with a partner and take each others quizzes!

Assignment 7: While Loops

1. You are given the Repeater.java class
2. Read through the current code and run it. Make sure the code and the while loop make sense to you.
3. Make the program print the message back 10 times instead of 3.
4. Put numbers in front of the message each time it prints. For example, if the message is 'Hi!', it should look like this:

   ```
   1. Hi!
   2. Hi!
   3. Hi!
   … etc
   ```

   This is tricky because the numbers start at 0, not 1. You can either change the starting number and the ending number by 1 each, or compute the number to print based on the count value.
5. Change the program so that it asks the user how many times they want the message repeated, and then repeats the message that number of times.
6. Run the program and make sure the output works as expected.
7. Save the program in Git.

Assignment 8: Doubles

1. You are not given a starter class for this project. You can use an old project to get started! Name the class for this project Doubles.java.
2. Create a program that rolls 2 dice (using random numbers between 1 and 6) and prints the results.
3. Use a while loop to keep going until it rolls doubles. If they roll doubles, print a message of congratulations.
4. This all happens so fast! Change the program so that each time it asks the user if they want to roll the dice and waits for user input to go on. If the user does not want to roll the dice, you can break out of the loop.
5. Run the program and make sure it works as expected.
6. Save the program in Git.

Assignment 9: HighOrLow

1. You are not given a starter class for this project. You can use an old project to get started! Name the class for this project HighOrLow.java.
2. Your program will play the High-Low game with the user. It should pick a random number between 1 and 100 and prompt the user to guess. If the guess is lower than the secret number, tell the user they are too low. If the guess if higher than the secret number, tell the user they are too low. If they guess the secret number, print a message of congrats!
3. Run the program and make sure it works as expected.
4. Add a counter to keep track of the number of guesses. At the end, print how many guesses they took to get the number.
5. Run the program and make sure it works as expected.
6. Save the program in Git.
7. Swap with a partner and play their High-Low game.

Assignment 10: Functions

1. You are given the class GraphingFormulas.java as a start.
2. Read through the code and run the program. Why do all the answers come out to zero?
3. Edit the slope and distance formulas so that they compute the slope and distance correctly.
    a. Remember the formula for slope using points (x_1, y_1) and (x_2, y_2) is

    $$m = \frac{y_2 - y_1}{x_2 - x_1}$$

    b. Remember the formula for distance using points (x_1, y_1) and (x_2, y_2) is

    $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

    c. You may need these functions:
        i. Math.pow(n1, n2) raises n1 to the n2 power
        For example, Math.pow(5,2) gives 5 squared which is 25.
        ii. Math.sqrt(n) gives the square root of n
        For example, Math.sqrt(49) gives 7.
4. For a bonus, make a third function that is void but prints out the formula for the line between the two given points in y=mx+b form. You should call your slope function inside of this function. Then, add 2 test statements for this function in the main part of the program.
5. Run the program and make sure it works as expected.
6. Save the program in Git.

Assignment 11: Arrays

1. You are not given a starting class for this project. Create one on your own with a main and name it ArrayFun.java.
2. Create an array of integers with size 10. Fill up the array with odd numbers (1,3,5,...). Put the values in the array indices one at a time and print them on the screen one by one. Don't use a while loop. This part of the code will be pretty long.
3. Create another array of doubles with size 10. Fill up the array with the number 1.25, then print all the numbers in the array. You must use a while loop to fill this array and to print. Use the length of the array for the while condition.
4. Create another array of integers with size 1000. Fill it with random numbers between 1 and 1000. Use a while loop! Use the length of the array for the while condition.
5. Challenge: print all the numbers in the random array out with a space between each one, making them start on a new line every 20 numbers. You can use a double while loop to do that! This is tricky!
6. Run the program and make sure it works as expected.
7. Save the program in Git.

Assignment 12: For Loops

1. You are not given a starting class for this project. Create one on your own with a main and name it ForLoopFun.java.
2. In between each of the following parts, print a blank line using System.out.println();
3. First, use a for loop to print "I love Java" 10 times.
4. Then, prompt the user to enter a number, and then use a for loop to count from 0 to that number. Print each number separated by a space.
5. Next, prompt the user for a starting number, an ending number, and a number to count by. Then, count from the starting number to the ending number by the counting number using a for loop, again with a space between each number. For example, if the starting number is 2 and the ending number is 8 and the counting number is 2, you should print "2 4 6 8".
6. Create an array of ints with values 1-10. You can use a for or while loop to fill it. Then, use a for loop to create a copy of the array, where all the values are changed by 2 (add 2 to each number). Create the second array before you fill it using a for loop. Print out the values of the second array separated by spaces.
7. Bonus: Create an array of ints with values 1-100. Use a for loop to print the numbers line by line, but print an exclamation point after multiples of 5. Use an if statement and the math concept of mod:
    a. x mod y gives the remainder when x is divided by y. For example, 5 mod 2 is 1 because 5/2 is 2 remainder 1. In Java, % is mod. So 12 % 5 is 2. Use this to find multiples of 5.
8. Run the program and make sure it works as expected.
9. Save the program in Git.

Assignment 13: Car

Part 1

1. You are given the Car and CarExample examples. Run them and make sure you understand what the following words mean: variable, method, input variable, return value.
2. Add the following variables to the Car class:
    a. A String called nickName, to represent the nick name for the car
    b. A String called color, to represent the color of the car
    c. An int called yearsOwned, to represent the number of years the current owner has owned the car
    d. A double called miles, to represent the number of miles the car has driven
3. Change the constructor so that it includes input variables for all the new ones above.
4. Add getters and setters for each of the new variables. There is a shortcut for this: highlight the variables you want to make getters and setters for. Right click on them. Click 'Insert Code,' then 'Getter and Setter'. Check the ones you want, then click generate.
5. Then, add variables and put them in the constructor in the CarExample class so that it doesn't have an error.
6. Add all the new information to the print statement below. You can use new lines.
7. Run the program and make sure it works as expected.
8. Save the program in Git.

Assignment 13: Car

Part 2

9. You are given the CarTest file with unit tests on it.
10. Modify the constructor in the CarTest class so that it works correctly with all your new variables.
11. Add unit tests for all the new methods you added. There should be 14 tests when you are done.
    a. Note that for the milesGetter and milesSetter tests, assertEquals will work a little differently. For doubles, the function takes in the value, expected value, and a double that is the allowed error. It checks if the value is within the error's distance of the expected value. Use the variable e as your error value.
        i. Ex. assertEquals(c.getMiles(), 67152.2, e); will check if c.getMiles is within e, which is a very small number, of 67152.2.
12. Run the program and make sure it works as expected.
13. Save the program in Git.

Assignment 14: Person

1. Open your project from earlier called Person.
2. Remove the 'main' block. Remove all the specific values from the variables age, height, firstName, lastName, and isAStudent. Remove the 2 variables you added and all the print statements.
3. Create a new String variable called phoneNumber.
4. Create a constructor using only firstName, lastName, and phoneNumber. Create getters and setters for each variable.
5. Create a new class called PhoneBook. Make a variable that is an array of Person objects.
   a. Person[] is the way to represent that.
   b. Make another variable that is an int to represent the length of the phone book, ie. how many people are in it.
6. Make a constructor that doesn't take in anything, so that its use will be:
   new PhoneBook();
   Inside, set the length variable to 0, and create the array of people, with length 100.
7. Create a method called addPerson that takes in a person and adds them to the array. Use the length variable to keep track of where you are in the array, and make sure to add one to it (++) each time you add someone.
8. Create a method called printEntries that will print the phonebook out. Only print the number to the length of the phonebook so far (not all 100 entries, because it will probably not be full! Make sure it prints in this format:
   ```
   Alix Feinsod, 5105555555
   Bob Smith, 5108888888
   ```
9. Create a main class called PhoneBookExample. In it, create several people. Create a phone book and enter the people into the phone book. Then, call the printEntries method on that phonebook. Run the main class and make sure it works as expected.
10. Create a Junit test for the PhoneBook. Add tests for all the methods.
11. Run the tests and make sure they all pass.
12. Save the program in Git.

Assignment 15: I/O

1. You are given the project MathIO.java to start. Note that this file has a lot in it to start with, this is very similar to the 2 examples we just looked at the read and write files.
2. Part 1: Modify the program to write 5 separate lines to the file. The lines should each contain one double. Make sure to include \n at the end of each line so that the file knows to go to a new line.
3. Part 2: this part currently reads in the entire file and prints it line by line. Modify it so that it keeps a running sum total of all the numbers read in. You have to convert the Strings to doubles. At the end, print the sum.
4. Save the program in Git.

Assignment 16: Puzzles

1. You are given the project Puzzles.java to start. It currently reads in the file “puzzle.txt” and prints it line by line.
2. Modify the code so that it only prints every third line.
3. Create a file “puzzle.txt” that contains a secret message in it on every third line.
4. Run the project to see if the secret message comes out.
5. Share your file with a friend and see if they can find the secret message.
6. Save the program in Git.

Assignment 17: ArrayLists

1. You are not given a starting project. Create one of your own with a main class called SearchList.java.
2. Create an ArrayList and fill it with 10 random integers between 0 and 50.
3. Prompt the user for a number to search.
4. Go through the ArrayList and see if that number is in it. The best way to do with will be to create a boolean called found, that begins as false. Then, loop through the ArrayList with a while or for loop and change found to true if any value in it is equal to the user's number.
5. Print that you found the number or that you didn't find it, depending on the boolean.
6. Next, find the largest number in the ArrayList. You can use a similar loop as before, except instead of a boolean you should create an integer that holds the largest value so far as you go through the ArrayList. Print that value at the end.
7. Create another ArrayList and add all the words in this sentance to it as Strings: A bunch of revolutionary manumission abolitionists, give me your position show me where the ammunition is!
8. Use String comparison (the .compareTo) method to sort this list into alphabetical order. This is tricky! Scan through the list to find the index of the first alphabetic word. Copy it into a second ArrayList, remove it from the first, and repeat the process until the first ArrayList is empty. Print the sorted list.
9. Run the program and make sure it works as expected.
10. Save the program in Git.

Assignment 18: Challenge Projects

Page 1 of 2

This is a list of projects that are increasingly challenging. Complete all that you can within the class time.

Project 1: Choose Your Own Adventure

1. You are not given a starting project. Create one of your own with a main class called ChooseYourOwnAdventure.java.
2. Make a "Choose Your Own Adventure" game. The starting room should give the user two choices. Then the second room they travel to should give them two more choices. Finally the third room should give them two choices. Each ending room should have a different message. Your game will have a total of fifteen rooms: 1 starting room, 2 possibilities from there, 4 from those, 8 final rooms from those.
3. Run the program and make sure it works as expected.
4. Save the program in Git.

Project 2: Hat Shop

5. You are not given a starting project. Create one of your own with a main class called HatShop.java.
6. This program should pull up a menu for a Hat Shop with 4 options: Add a hat to cart, see your cart, remove hats from the order, and check out (which should end the program). These 4 options should call 4 different functions. The program should keep track of how many hats are in the order at any time. For an extra challenge, allow hats to have different colors and allow for adding and removing different colors, and print the colors when you see the cart.
7. Run the program and make sure it works as expected.
8. Save the program in Git.

Project 3: Even Fibonacci Numbers

9. You are not given a starting project. Create one of your own with a main class called Fibonacci.java.
10. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:
11. 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

    By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms. Print the sum out.
12. Run the program and make sure it works as expected.
13. Save the program in Git.

Assignment 18: Challenge Projects

Page 2 of 2

Project 4: Hangman

14. You are not given a starting project. Create one of your own with a main class called Hangman.java.
15. It must randomly choose a word from a list of words, stop when all the letters in the word are guessed. It must give them limited tries and stop after they run out. It must display the letters they have already guessed. For a bonus, take in a file with the words to choose from. Suggestion: make a list of the letters in the word.
16. Run the program and make sure it works as expected.
17. Save the program in Git.

Project 5: Tic Tac Toe

18. You are given starter code in a file called TicTacToe.java.
19. Complete the project so that it makes a 2-player game of Tic Tac Toe, alternatively prompting X and O to choose their location, which is one int representing the row on the board and one int representing the column. The game should end if someone wins (gets 3 in a row) or if the board is full and it is a tie.
20. Run the program and make sure it works as expected.
21. Save the program in Git.

Final Project Options

Address Book - Write an application that acts like an address book. Each entry in it should contain first and last name, phone number, and email. It should be able to sort by at least one field (like last name) - more would be excellent! Next, make it be able to only print entries that match certain criteria (like certain area codes or first names beginning with A).

Calculator - Write a program that acts as a calculator. It should take integers and decimals, and do addition, subtraction, multiplication, division, exponents, and give error messages when something is typed wrong. An excellent project will also do modulus, factorial, square roots, and if you press a certain button, it displays a manual of how to use it.

Quiz Maker – Make an application which takes various questions form a file, picked randomly, and puts together a quiz for students. Each quiz can be different and then reads a key to grade the quizzes.

Product Inventory Project – Create an application which manages an inventory of products. Create a product class which has a price, id, and quantity on hand. Then create an inventory class which keeps track of various products and can sum up the inventory value.

Movie Store – Manage video rentals and controls when videos are checked out, due to return, overdue fees and for added complexity create a summary of those accounts which are overdue for contact.

Student Grade Book Application – Keep track of students (with a student class that has their name, average, and scores) in a class and their grades. Assign their scores on tests and assignments to the students and figure out their average and grade for the class. For added complexity put the students on a bell curve.

Bank Account Manager - Create a class called “Account” which will be an abstract class for three other classes called “CheckingAccount”, “SavingsAccount” and “BusinessAccount”. Manage credits and debits from these accounts through an ATM style program.

Library Catalog – Create a book class with a title, page count, ISBN and whether or not it is checked out or not. Manage a collection of various books and allow the user to check out books or return books. For added complexity generate a report of those books overdue and any fees. Also allow users to put books on reserve.