Midterm Topics

1. Know your topics from previous java programing course
    a. Interface, class
        i. Functional interfaces
        ii. Lambdas, function reference
        iii. Default
        iv. Varargs (…)
    b. Methods, constructors, variables
        i. Chaining constructors and methods
    c. Static, final, accessors
    d. Overload and override
    e. Inheritance and polymorphism

2. JavaFX, Look at JavaFX Node Structure PDF
    a. Layouts which all inherit from Pane

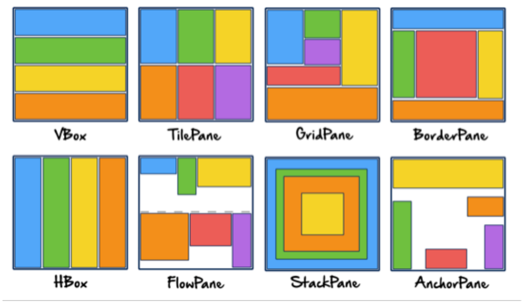

Figure 1-https://dzone.com/refcardz/javafx-8-1?chapter=9

    b. Controls which all extend from Control. Used for interacting with user.
    c. Event handling
        i. MouseEvent
            1. Pressed
            2. Dragged
            3. Released
        ii. KeyEvent
            1. Pressed
            2. Released
        iii. TouchEvent
            1. Pressed
            2. Moved
            3. Released
        iv. ActionEvent

3. Know the code for the assignments and sample codes

Midterm Topics

4. UML, Class Diagram and Sequence Diagram, look at the sample diagrams provided
    a. Know the arrows and symbols
    b. Know how the components look and connect together
    c. Create code from diagram

5. Junit, look at the JUnit slides posted
    a. Know the different asserts and how to use them
        i. assertEqual, compare two variables using equals
            1. use assertEqual with delta for double and float comparison
        ii. assertTrue, except a true variable
        iii. assertFalse, except a false variable
        iv. assertSame, compare two variables using "=="
        v. fail, force fail every time
    b. know the annotations and how to use them
        i. @Before/@BeforeEach, run before each @Test.
            1. Used for initialization of fresh object before each @Test
        ii. @After/@AfterEach, run after each @Test
            1. Used to clear any initialization or close resources after each @Test
        iii. @Test, primary test method
        iv. @BeforeClass/@BeforeAll, run once before all tests in this class
        v. @AfterClass/@AfterAll, run once after all tests in this class
    c. Know how to catch exceptions for testing
        i. assertThrows
        ii. try{ fail} and catch{assertTrue}
        iii. @Test(expected = IndexOutOfBoundsException.class)
    d. Construct test code for given code with exceptions

6. Jar File, can it be executed. What is special about executable jar file, look at slides posted

7. Exceptions
    a. When and how to use them
    b. Checked
        i. Exception, IOException, IntruptedException
    c. Unchecked
        i. RuntimeException, NullPointerException, IndexOutOfBoundsException
    d. try catch finally block
    e. try catch resource

8. Enum, definition and uses, look at sample code posted

Midterm Topics

9. Design Patterns, know what they are, how they are used. Recognize them from UML Diagrams
    a. MVC, an architecture for separating parts of code into model, view and controller.
    b. Observer, a behavioral pattern for notifying many observers of changes in model.
    c. Singleton, a creational pattern which enforces use of an instant through application.
    d. Fluent interface, a creational pattern for creating complex objects using method chain.

10. Data Structures
    a. List, add, get, remove. Interface used by ArrayList, Stack, LinkedList and other
    b. Map, put, get, remove. Interface used by HashMap and other
    c. Stack, LIFO, push, pop, peek.
    d. Queue, FIFO, add, poll, peek. Interface used by LinkedList
    e. ArrayList and Vector. Based on array, Arraylist is not synchronized while vector is.
    f. LinkedList and DoublyLinkedList. Based on connecting Objects (nodes)

11. Big-O Notations. Simply distinguish which is better
    a. $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$

12. Sorting
    a. Bubble, Insertion and Selection
    b. Quick and Merge (divide and conquer)
    c. Detect from code and or pics the type of sort

13. Thread
    a. Runnable and Callable interfaces
    b. Thread
        i. join, start, sleep, getName, getState, interrupt
    c. ExecutorService
        i. shutdown, submit
    d. Executors
        i. newCachedThreadPool, newFixedThreadPool, newSingleThreadExecutor
    e. different states of Threads
        i. New, Runnable (Ready and Running), Timed Waiting, Waiting, Blocked and Terminated
    f. Know the definition
        i. Synchronized, java keyword, allow access to a section of code for one thread only. Other threads will have to wait till working thread leaves block.
        ii. Deadlock, two threads gain sync lock while waiting for other one to release the lock. One way to prevent is from all threads to take their locks in the same order.
        iii. Race condition, multiple threads try and access the same code and create a buggy outcome.
        iv. Starvation, when one or more thread must wait for access because other threads have higher priority.

14. Networking
    a. TCP, guaranties delivery of data
        i. ServerSocket, used on server side to accept connections from clients. Must be closed.
            1. Accept, a blocking method. waits for client to connect and return a Socket for the received connection.
            2. setSoTimeout, defines how long accept should wait.
        ii. Socket, used on client and server to communicate between two systems. Must be closed.
            1. setSoTimeout, defines how long accept should wait.
            2. getOutputStream, get and output stream to send data to client/server. Must be closed.
            3. getInputStream, get input stream that receives data from client/server. Must be closed.
    b. UDP, delivery is not guaranteed. Data is sent in packages and order is not guaranteed.
        i. DatagramSocket, create socket that uses a given port to send or receive. Must be closed.
            1. send, a DatagramPacket to server/client.
            2. receive, a DatagramPacket to server/client.
        ii. DatagramPacket, use an array of byte to store information. DatagramSocket uses this package to send/receive data.
            1. getPot, get source port of the package.
            2. getAddress, get source address of the package.
            3. getLength, get the total length of bytes received.
            4. getData, return an array of bytes with bytes read from client/server.
    c. InetSocketAddress, pass IP address and port number to constructor to create a socket address. Can be used with DatagramPacket to connect to server.