Final 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

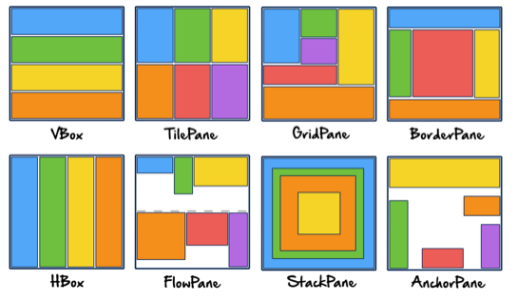

Figure 1-https://dzone.com/refcardz/javafx-8-1?chapter=9

    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

Final 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, SocketException
    c. Unchecked
        i. RuntimeException, NullPointerException, IndexOutOfBoundsException
    d. try catch finally block
    e. try catch resource, used for closable resources like Connection and Socket.

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

Final 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 interface, works with both Thread and ExecutorService
        i. Only has one method called run which takes no arguments and returns void. Cannot throw checked exceptions.
    b. Callable<T> interfaces, only works with ExecutorService
        i. Only has one method called call which takes no arguments and returns generic T. Can throw checked exceptions.
    c. Thread, used to create a new thread instance.
        i. join, a blocking method that will keep the current thread till thread object has been terminated.
        ii. start, start a new thread instance, none blocking method.
        iii. sleep, a static method can put the current thread to sleep for given time.
        iv. getName, get the current name of the thread, default name if none provided.
        v. getState, return the state of this thread as shown below.
        vi. currentThread, a static method that returns the Thread object of current running thread.
        vii. interrupt, a thread can be interrupted using interrupt exception or interrupt method.
        viii. priority, between 1-10 and by default is 5. When started a thread will inherit its parent's priority. Priority can be changed.

d. ExecutorService, an interface used for control of Pool of Threads.
    i. shutdown, must be called at the very end of the pool lifecycle.
    ii. submit, non-blocking method to start a new thread or reuse an inactive thread.
e. Executors, utility class for initializing an ExecutorService
    i. newCachedThreadPool, create a pool with open ended number of threads.
    ii. newFixedThreadPool, create a pool with fix number of threads.
    iii. newSingleThreadExecutor, create a pool with one thread.
f. different states of Threads
    i. New, Runnable (Ready and Running), Timed Waiting, Waiting, Blocked and Terminated
g. 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.
        1. If synchronized is on method definition it means the object itself is the lock.
        2. If synchronized is used as code block a custom lock can be provided.
    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.

15. JDBC, Java Database Connectivity driver provided by database provider
    a. Connection, an Object used for creating PreparedStatement. An instance of this object can be retrieved from DriveManager or javax.sql.DataSource.
    b. PreparedStatement, an Object used for creating ResultSet. This class is used to safely prepare a query to be executed. This means any unsafe character will be skipped.
        i. Set function, set specific values for "?" used in the query. "?" can only be used for input values and the index starts at 1, NOT 0.
        ii. ExecuteUpdate, function used to execute queries with no results.
        iii. ExecuteQuery, function used to execute queries with results.
    c. ResultSet, is a Cursor on the database. Used for retrieving result of the executed query.
        i. Next, function used to get the next row of the result. Next must be called the first time ResultSet is accessed.
        ii. Get functions, are used to get the specific type of data from ResultSet.

16. Design Patterns, Part 2
    a. Delegate, like inheritance done manually through object composition.
    b. Builder, Separate the Construction of a complex object from its representation so that the same construction process can create different representations.
    c. Abstract Factory, Provide and interface for creating families of related or dependent objects without specifying their concrete classes.
    d. DAO, Abstract and encapsulate all access to the data source. The DAO manages the connection with the data source to obtain and store data
        i. Logic, set of classes that use Data Access to CRUD (create, read, update, delete)
            1. Create, INSERT to database if using HTTP do POST
            2. Read, SELECT from database if using HTTP do GET
            3. Update, UPDATE the database if using HTTP do PUT
            4. Delete, DELETE from database if using HTTP do DELETE
        ii. Data Access, set of classes to manipulate the data base
        iii. Transfer Object, set of classes where each represent one row of a table.

Final Topics

17. Know your SQL type to Java type conversion. Posted on Lecture Material - 10.

18. Tomcat, know what was done in assignment 3
    a. Servlet, process HTTP request and response by protocols like post, get, update and delete. We only covered post and get.
        i. HttpServletRequest, holds all the data received from the browser. The data can be extracted using getParameterMap which returns a map of all data in string format.
        ii. HttpServletResponse, holds all the data to be sent to the browser.
        iii. doPost, doGet, doPut and doDelete are equivalent methods for Http methods.
    b. Contect.xml, holds info such as primary address of your webpage and resources like JDBC driver
    c. Web.xml, holds connecting info such as URL patterns and associated servlets and Link to resources defined in contect.xml. there can also be other info like welcome pages which was not covered.
    d. JDNI, Java Naming and Directory Interface which is used to look up resources. Used in DataSource.java
    e. There won't be any html questions.

19. N-Layered System, separation of system in to multiple layers/tires such as database and logic .