Watch video in youube.com “getting started with mongoDB and java”

-replication = taksir

-entirely = kamelan

-whole = kol

-terribly = vahshatnak

-relevant = marboot

-reasonably = manteghi

-aggregation = tajammo va tarakom

-split up = taghsim kardan

-fetch = raftan va avordan(vakeshi)

-ultimately = dar nahayat

-rearrange = tanzim e mojaddad

-constrain = mahdoodiat

-advantage = maziat – fayede – sood

-------------------------------------------------------------not completed.

I install mongoDB on windows 10 and check successfully install it in cmd with its command.

The name of video in youtube = “how to install mongoDB on windows 10”

--------------------------------------------------------------

Watch video “MongoDB crash course”

-replication = taksir

-cluster = khooshe va daste va gorooh

-freedom = azadi va esteghlal va asani

Videos gol = “how to get a local installation set up”

-interchangeably = ghabel e taviz

-associated = mortabet ast.

-increment = afzayesh

-cluster = khooshe

-------------------------------------------------------------------

Connect server too shell and java program in mongoDB.com and cluster and atlas.

Be ok navicat and add mongoDB libraries in java program.

Add mongoDB atlas to navicat

------------------------------------------------------------------------

Add external libraries to java project in intellij



Or from below address :

<https://www.jetbrains.com/help/idea/library.html#define-library>

we can add external libraries to java project in intellij as : file – project structure – libraries – and click on the + (.jar files better that be in the ‘lib’ folder away src folder

------------------------------------------------------------------

**Start javaFX tutorial**

From

1. tutorialspoint.com/javafx/index.htm

-rely = tekye kardan

-advent = zohoor

-assumed = farz shode

-prior = ghabli – sabeghe ghabli

-overview = barresi ejmali

-feature = khasise va vijegi

-delivered = delivered

-delivered = tahvil dadan

-consistently = hamvare

-opt = entekhab mikonad

-sole = tanha

**Scene Builder** − JavaFX provides an application named Scene Builder. On integrating this application in IDE’s such as Eclipse and NetBeans, the users can access a drag and drop design interface, which is used to develop FXML applications (just like Swing Drag & Drop and DreamWeaver Applications).

**Swing Interoperability** − In a JavaFX application, you can embed Swing content using the **Swing Node** class. Similarly, you can update the existing Swing applications with JavaFX features like embedded web content and rich graphics media.

-styling = tarrahi zaher

-capabilities = ghabeliat ha

-concepts = mafahim

-responsible = masool – ohde dar –

-deliver = erae dadan

**javafx.stage** − This package holds the top level container classes for JavaFX application.

**javafx.scene** − This package provides classes and interfaces to support the scene graph. In addition, it also provides sub-packages such as canvas, chart, control, effect, image, input, layout, media, paint, shape, text, transform, web, etc. There are several components that support this rich API of JavaFX.

-architecture = memari

-node = eshkal

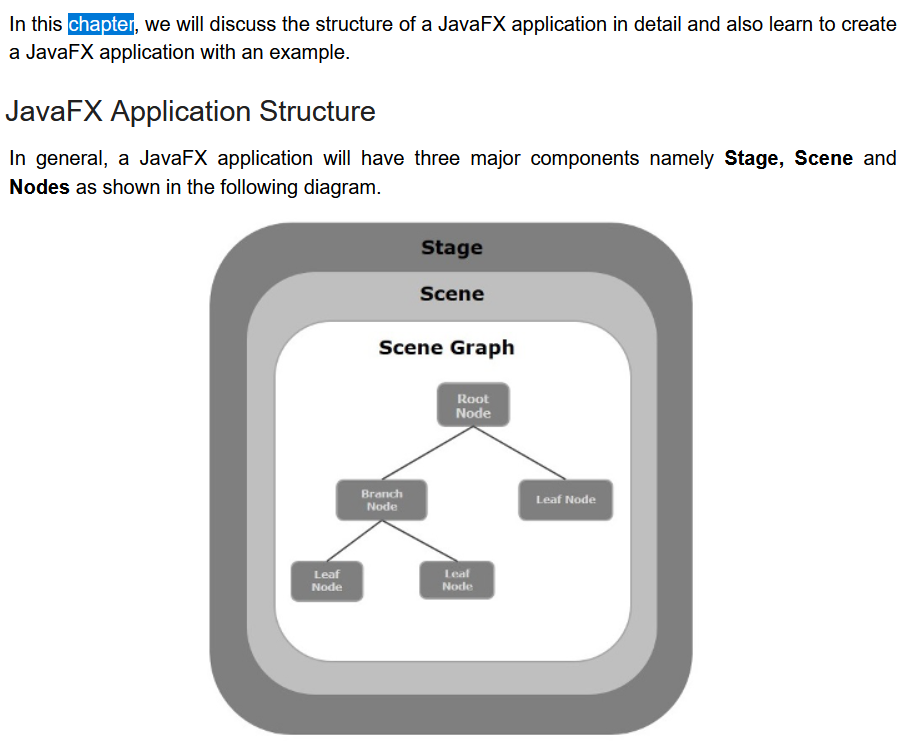
-hierarchical = bar asas e selsele marateb

-chapter = fasle – ghesmat

-transformation = tagheer shekl – degargooni

-mandatory = ejbari

-

= ba

-alteration = tagheer

-retrieve = bazyabi

-attach = chasbandan

-terminating = payan dadan

**Terminating the JavaFX Application**

When the last window of the application is closed, the JavaFX application is terminated implicitly. You can turn this behavior off by passing the Boolean value “False” to the static method **setImplicitExit()** (should be called from a static context).

You can terminate a JavaFX application explicitly using the methods **Platform.exit()** or **System.exit**(int).

-individually = be tor e jodagane

-transitions = enteghal

-Straight = mostaghim

-coordinates = mokhtassat

-respectiv = marboote

//Creating a Group object

Group root = new Group();

//Retrieving the observable list object

ObservableList list = root.getChildren();

//Setting the text object as a node to the group object

list.add(text);

-geometrical = hendesi

-ellipse = beyzi

-polygon = chand zelei

text.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 20));

-scaling up = bala bordan

-shearing = boresh

-translation = harkat e enteghli

-translation = be kar bordan va emal kardan

-uniformly = yeknavakht

-radial = shoaei

In the same way, you can also use the RGB values or HSB standard of coloring or web hash codes of colors as shown below −

//creating color object by passing RGB values

Color c = Color.rgb(0,0,255);

//creating color object by passing HSB values

Color c = Color.hsb(270,1.0,1.0);

//creating color object by passing the hash code for web

Color c = Color.web("0x0000FF",1.0);

Image image = new Image(new FileInputStream("C:\\images\\logo.jpg"));

## Writing Pixels in part of the images of the tutorials is very important

-aspect = janbe va manzar

//Styling nodes

button1.setStyle("-fx-background-color: darkslateblue; -fx-text-fill: white;");

button2.setStyle("-fx-background-color: darkslateblue; -fx-text-fill: white;");

text1.setStyle("-fx-font: normal bold 20px 'serif' ");

text2.setStyle("-fx-font: normal bold 20px 'serif' "); gridPane.setStyle("-fx-background-color: BEIGE;");

//list View for educational qualification

ObservableList<String> names = FXCollections.observableArrayList( "Engineering", "MCA", "MBA", "Graduation", "MTECH", "Mphil", "Phd"); ListView<String> educationListView = new ListView<String>(names);

**CSS(Cascading Style Sheets)**

- facility = emkanat

- enhance = taghviat

- comprise = shamel

- applicable = ghabele ejra

------------------------------------------------

Watch video tutorial javaFX as faradars

-occurrence = voghoo

-target = hadaf

-route construction = sakht e masir

-capturing = gereftan