Build steps of Sudoku

Java: 18,Javafx: 22,Maven, jlink, jpackager,wix toolset used  
  
jlink is used to assemble and optimize a set of modules and their dependencies into a custom runtime image. It doesn't directly create installers or executable files; rather, it creates a reduced size runtime environment for your application, which can significantly decrease the size of the application bundle and improve startup time.

The output of jlink is a directory containing the custom runtime image. This image includes a bin directory with executables to run your application (like java.exe on Windows), but it's not a standalone installer.

jpackage

jpackage is a tool that can take the output of jlink (among other inputs) and package it into a native installer, such as an .exe or .msi file for Windows, a .dmg or .pkg for macOS, or a .deb or .rpm for Linux.

For Windows specifically, creating .exe or .msi installers requires the WiX Toolset, as it uses WiX's light.exe and candle.exe tools to build the installer. WiX is a third-party toolset for creating Windows installation packages from XML source code. jpackage does not include these capabilities natively and therefore depends on WiX being available in the environment where it's run.

In summary, while jlink and jpackage are powerful tools within the JDK for preparing and packaging Java applications, creating a native Windows installer with jpackage does indeed require the WiX Toolset. This dependency is specific to packaging for Windows; packaging for other operating systems with jpackage does not require WiX or similar external tools.

Install wix3, and add its bin to path of the system variable.

There a lot of compatibility issues between wix 3 and wix 4.  
Wix3 being the one to work in most scenarios, do read up more and consider incompatibilitites here for troubleshooting.  
  
from intellij idea terminal, the project runs well with command: *mvn clean javafx:run*

Detailed documentation of packaging in java18 is given here: [Packaging Overview (oracle.com)](https://docs.oracle.com/en/java/javase/18/jpackage/packaging-overview.html#GUID-E0966C49-ABBB-46A2-8DF7-1D3F96640F05)

STEPS

1. Build a jar using mvn clean package: this will build a jar in the target folder of the project.  
   mvn clean install creates the jar as well.
   1. Use jlink to create a custom runtime image that includes the JavaFX modules needed and jre.

jlink --module-path "C:/Program Files/Java/jdk-18.0.1.1/jmods;C:/Program Files/Java/openjfx-22\_windows-x64\_bin-sdk/javafx-sdk-22/lib" --add-modules java.base,javafx.controls,javafx.fxml,javafx.web,javafx.media --output C:/Users/itsni/IdeaProjects

jpackage --type exe --app-version "1.0" --input target/ --dest target/dist --name Sudoku --main-jar sudoku-1.0-SNAPSHOT.jar --main-class com.sudoku.Sudoku --runtime-image C:/Users/itsni/IdeaProjects/sudoku-runtime --icon C:/Users/itsni/IdeaProjects/sudoku/src/main/resources/com/sudoku/sudoku\_icon.ico  
  
These two steps were expected to run, but it did not, I kept getting error failure to launch JVM.  
  
not tried the solution listed here, but could be super useful:  
<https://stackoverflow.com/questions/61294243/running-javafx-application-after-jpackage>

* + 1. so instead of these 2 commands the commands I followed were  
       in intellij idea, terminal, ran   
         
       **mvn -e -X clean javafx:jlink**   
         
       internally the command run is

[DEBUG] Executing command line: [C:\Program Files\Java\jdk-18.0.1.1\bin\jlink.exe, --module-path, C:\Users\itsni\IdeaProjects\sudoku\target\classes;C:\Users\itsni\.m2\repository\com\dlsc\formsfx\formsfx-core\11.3.2\formsfx-core-11.3

.2.jar;C:\Users\itsni\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.12.1\jackson-annotations-2.12.1.jar;C:\Users\itsni\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.12.1\jackson-core-2.12.1.jar;C:\Users\

itsni\.m2\repository\com\fasterxml\jackson\core\jackson-databind\2.12.1\jackson-databind-2.12.1.jar;C:\Users\itsni\.m2\repository\com\github\almasb\fxgl\17\fxgl-17.jar;C:\Users\itsni\.m2\repository\com\github\almasb\fxgl-core\17\fxg

l-core-17.jar;C:\Users\itsni\.m2\repository\com\github\almasb\fxgl-entity\17\fxgl-entity-17.jar;C:\Users\itsni\.m2\repository\com\github\almasb\fxgl-gameplay\17\fxgl-gameplay-17.jar;C:\Users\itsni\.m2\repository\com\github\almasb\fx

gl-io\17\fxgl-io-17.jar;C:\Users\itsni\.m2\repository\com\github\almasb\fxgl-scene\17\fxgl-scene-17.jar;C:\Users\itsni\.m2\repository\com\gluonhq\attach\audio\4.0.9\audio-4.0.9.jar;C:\Users\itsni\.m2\repository\com\gluonhq\attach\li

fecycle\4.0.9\lifecycle-4.0.9.jar;C:\Users\itsni\.m2\repository\com\gluonhq\attach\storage\4.0.9\storage-4.0.9.jar;C:\Users\itsni\.m2\repository\com\gluonhq\attach\util\4.0.9\util-4.0.9.jar;C:\Users\itsni\.m2\repository\eu\hansolo\t

ilesfx\11.48\tilesfx-11.48.jar;C:\Users\itsni\.m2\repository\org\controlsfx\controlsfx\11.1.1\controlsfx-11.1.1.jar;C:\Users\itsni\.m2\repository\org\jetbrains\kotlin\kotlin-stdlib\1.5.32\kotlin-stdlib-1.5.32-modular.jar;C:\Users\it

sni\.m2\repository\org\kordamp\bootstrapfx\bootstrapfx-core\0.4.0\bootstrapfx-core-0.4.0.jar;C:\Users\itsni\.m2\repository\org\kordamp\ikonli\ikonli-core\12.3.0\ikonli-core-12.3.0.jar;C:\Users\itsni\.m2\repository\org\kordamp\ikonli

\ikonli-javafx\12.3.0\ikonli-javafx-12.3.0.jar;C:\Users\itsni\.m2\repository\org\openjfx\javafx-base\22\javafx-base-22-win.jar;C:\Users\itsni\.m2\repository\org\openjfx\javafx-controls\22\javafx-controls-22-win.jar;C:\Users\itsni\.m

2\repository\org\openjfx\javafx-fxml\22\javafx-fxml-22-win.jar;C:\Users\itsni\.m2\repository\org\openjfx\javafx-graphics\22\javafx-graphics-22-win.jar;C:\Users\itsni\.m2\repository\org\openjfx\javafx-media\22\javafx-media-22-win.jar

;C:\Users\itsni\.m2\repository\org\openjfx\javafx-web\22\javafx-web-22-win.jar, --add-modules, com.sudoku.sudoku, --output, C:\Users\itsni\IdeaProjects\sudoku\target\app, --strip-debug, --compress, 0, --no-header-files, --no-man-pages, --launcher, app=com.sudoku.sudoku/com.game.Sudoku]  
  
  
once the jlink command is run to create the run time image and the jar is created, we package it all together using jpackage  
  
jpackage --type exe --app-version "1.0" --dest target/dist --name Sudoku --main-class com.game.Sudoku --main-jar C:/Users/itsni/IdeaProjects/sudoku/target/sudoku-1.0-SNAPSHOT.jar --runtime-image target/app --icon C:/Users/itsni/IdeaProjects/sudoku/src/main/resources/com/game/sudoku\_icon.ico --input target/  
  
jpackage --type exe --app-version "1.0" --dest target/dist --name Sudoku --main-class com.game.Sudoku --main-jar C:/Users/itsni/IdeaProjects/sudoku/target/sudoku-1.0-SNAPSHOT.jar --runtime-image target/app --icon C:/Users/itsni/IdeaProjects/sudoku/src/main/resources/com/game/sudoku\_icon.ico --input target/classes  
  
this one creates a smaller exe 😊

This creates an exe file in the target/dist folder which can now be used to install and run sudoku as a stand alone program on any windows operating system without any pre requisite of having java installed.

some useful links:  
<https://stackoverflow.com/questions/68871952/how-to-use-jpackage-to-make-a-distribution-format-for-javafx-applications>

<https://stackoverflow.com/questions/61294243/running-javafx-application-after-jpackage>

<https://github.com/dlemmermann/JPackageScriptFX>

<https://docs.oracle.com/en/java/javase/15/docs/specs/man/jpackage.html>

<https://openjfx.io/openjfx-docs/#introduction>

<https://stackoverflow.com/questions/58821622/javafx-difference-between-modular-and-non-modular#:~:text=The%20word%20Modular%20defines%20that,is%20a%20modular%20JavaFX%20project>.