کارگاه برنامه نویسی پیشرفته

دستور کار شماره یازده

اهداف

آشنایی با JavaFx

آشنایی با Scene Builder

آشنایی با رویدادها در JavaFx

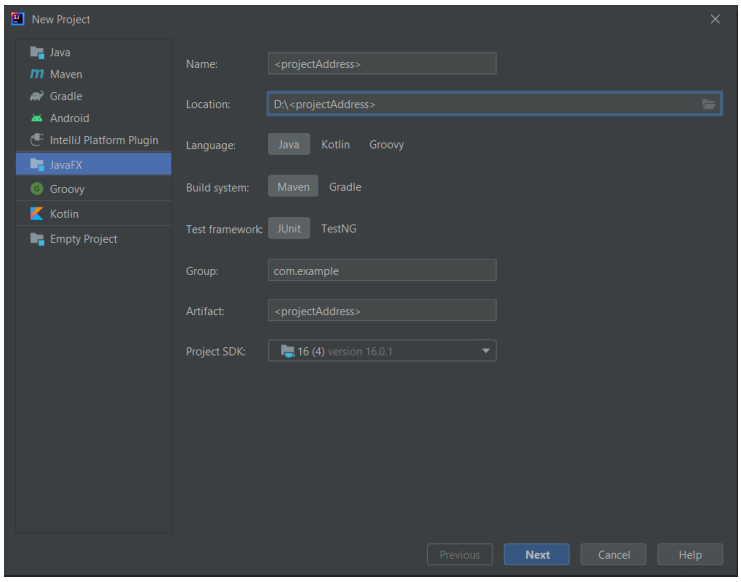
آشنایی با معماری MVC

فهرست مطالب

ساخت پروژه‌ی JavaFx در Intellij

خوشبختانه Intellij از نسخه‌ی 2021.2.1 خود، تنظیمات مورد نیاز برای ساخت پروژه‌ی JavaFx را مدیریت می‌کند و دیگر نیازی به دستی اضافه کردن تنظیمات نیست.

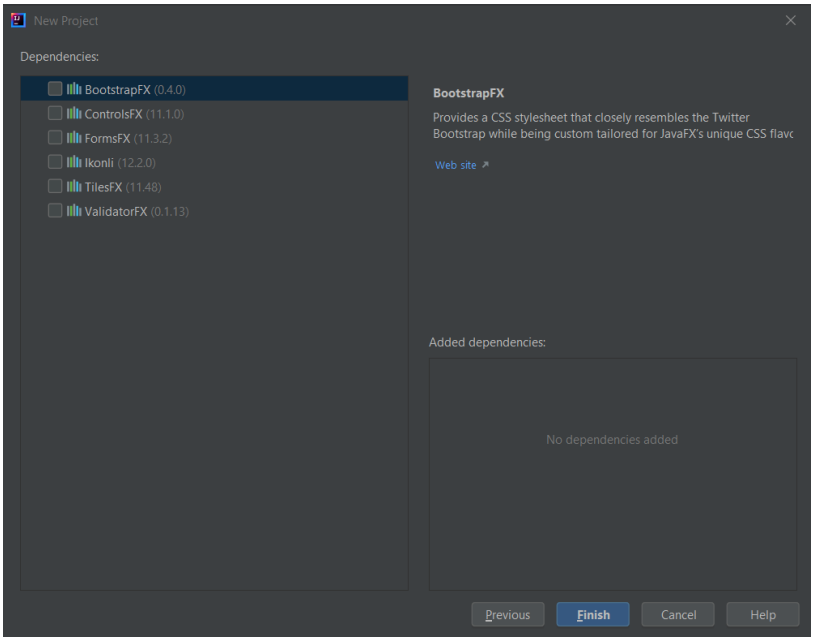
ابتدا قسمتProject -> New -> File را انتخاب کرده و با صفحه‌ی زیر مواجه می‌شویم:



(پنجره‌ی اول New Project)

سپس باید از ستون سمت چپ، گزینه‌ی JavaFX را انتخاب کرده و در قسمت سمت راست ابتدا نام و آدرس پروژه را وارد کرده و از زبان‌ها، جاوا و از سیستم‌های ساخت، Maven و از فریمورک‌های تست Junit را انتخاب کرد. همچنین می‌توان نام پکیجی که همراه با پروژه ساخته می‌شود را در قسمت Group تغییر داد و از قسمت Project SDK نیز می‌توان JDK مورد نیاز را دانلود یا اگر در کامپیوتر نصب شده باشد، آن را اضافه کرد. پس از انجام کارهای بالا، دکمه‌ی Next را می‌زنیم.

در مرحله بعد با صفحه‌ی Dependencies مواجه می‌شویم:



(پنجره‌ی Dependencies)

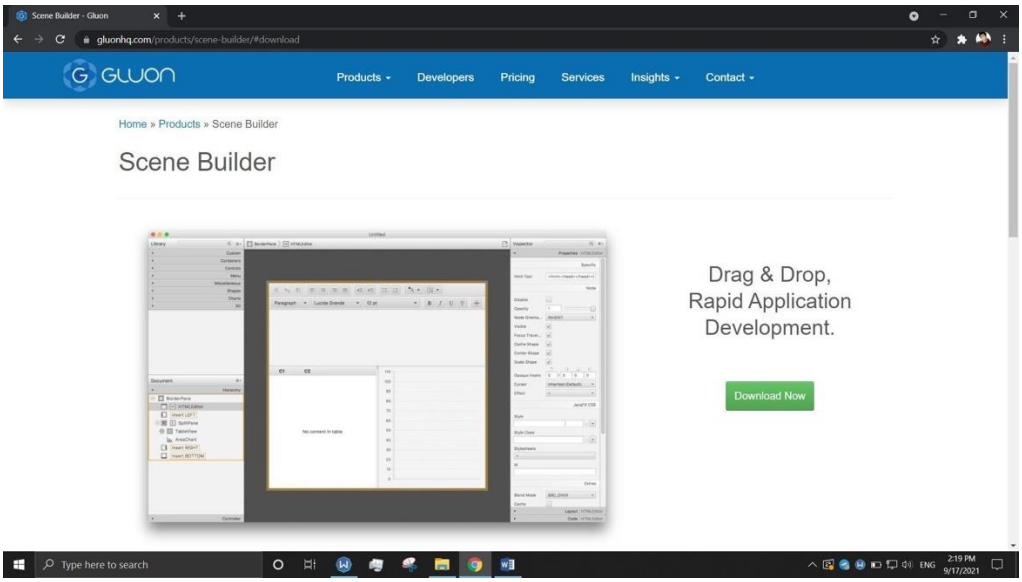
در این صفحه، می‌توان کتابخانه‌های اضافی را به پروژه اضافه کرد. در نهایت، گزینه‌ی Finish را انتخاب می‌کنیم و پس از آن، یک پروژه‌ی JavaFX آماده، در محل مشخص شده، ایجاد می‌شود.

همچنین برای مطالعه‌ی بیشتر می‌توانید به لینک Create a new JavaFX project مراجعه کنید. برای درک تفاوت بین Maven و Gradle می‌توانید به لینک Gradle vs Maven مراجعه کنید.

آشنایی با سین‌بیلدر[1]

نصب سین‌بیلدر

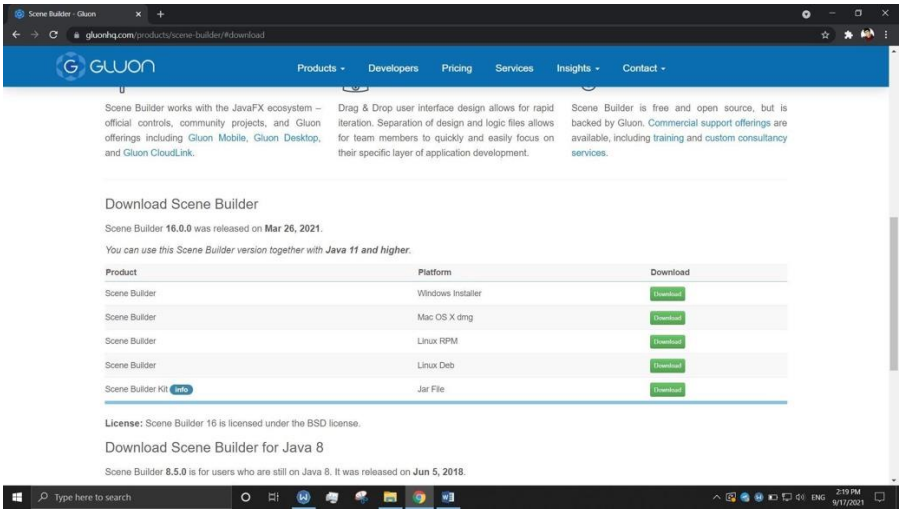
سین‌بیلدر، ابزار گرافیکی مورد تأیید شرکت اوراکل[2] (مالک جاوا) برای طراحی برنامه‌های گرافیکی با فریمورک JavaFX است که فرآیند طراحی فایل‌های fxml مربوط به sceneها را بهتر و راحت‌تر می‌کند. می‌توانید این ابزار را از این آدرس دانلود کنید (برای این کار نیاز به ابزارهای تغییر آی‌پی خواهید داشت):



(صفحه‌ی اصلی وب‌سایت Gluon)

در صفحه‌ی گفته شده بر روی گزینه Download Now کلیک کرده و با توجه به سیستم‌عامل خود، نسخه‌ی مناسب را دانلود کنید:

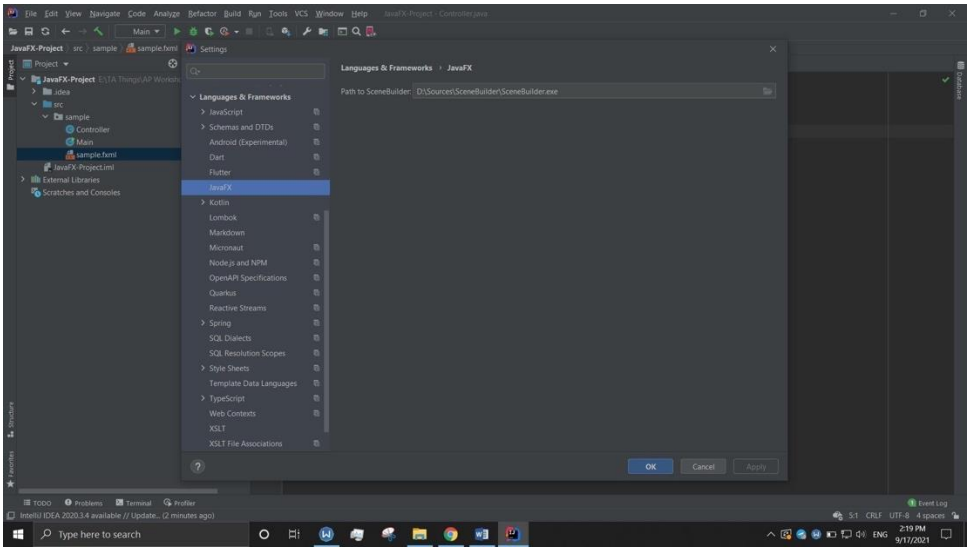
[1] Scene Builder
[2] Oracle



(انتخاب سیستم‌عامل مورد نظر)

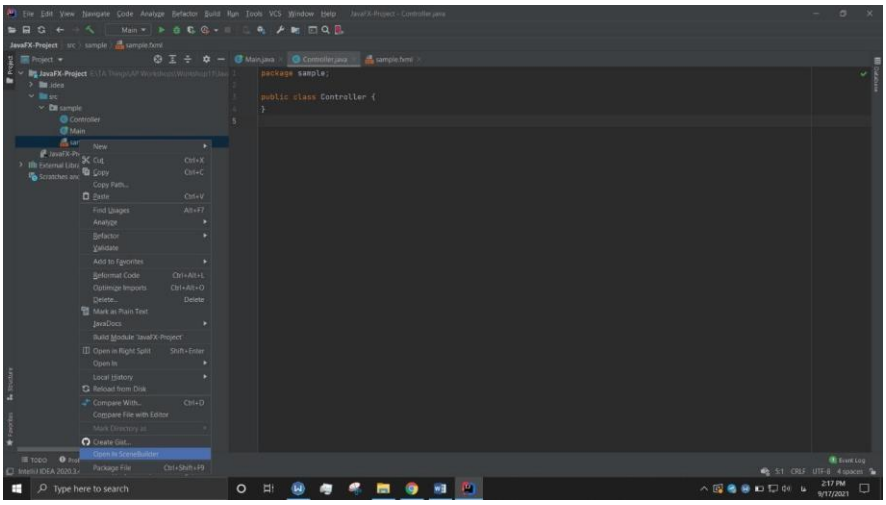
پس از به پایان رسیدن دریافت فایل مورد نیاز، آن را اجرا کرده و پس از انتخاب تیک مربوط به موافقت با لایسنس نرم‌افزار، گزینه‌ی Install را انتخاب کنید. پس از اتمام نصب، بر روی گزینه Finish کلیک کنید.

اکنون برای معرفی سین‌بیلدر به IDE، در محیط برنامه‌ی اینتلیجی، بر روی منوی File کلیک کرده و گزینه‌ی Settings را انتخاب کنید. سپس وارد بخش Languages Frameworks شده و بر روی گزینه‌ی JavaFX کلیک کنید. پس از آن، در صفحه‌ی باز شده در بخش Path to SceneBuilder، آدرس فایل SceneBuilder.exe که در محل نصب SceneBuilder قرار دارد را وارد کنید:



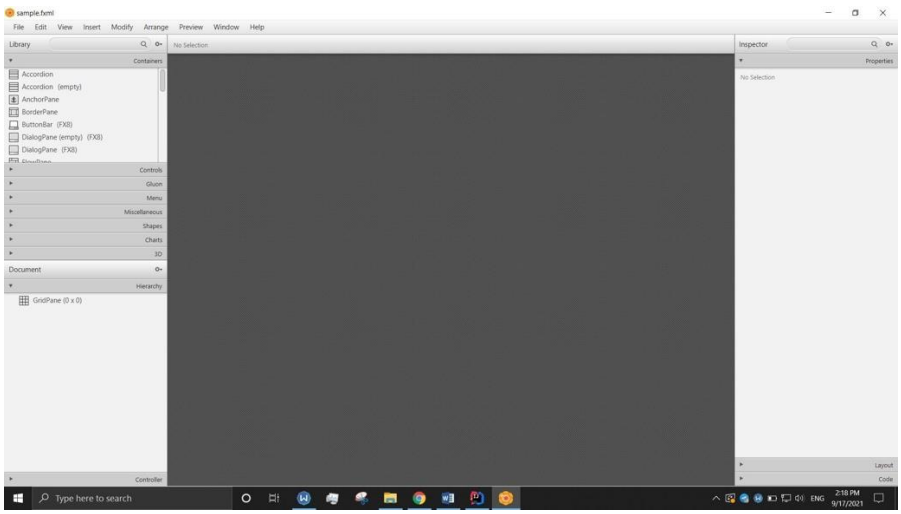
(انتخاب آدرس سین‌بیلدر)

اکنون با کلیک راست بر روی هر فایل fxml، می‌توان با انتخاب گزینه‌ی Open In SceneBuilder، به ویرایش فایل انتخاب شده در محیط گرافیکی برنامه SceneBuilder پرداخت:



(انتخاب گزینه‌ی Open In SceneBuilder)

در محیط برنامه‌ی سین‌بیلدر، می‌توان به عناصر طراحی شده در فریم‌ورک JavaFX دسترسی داشت و از آن‌ها در طراحی scene مورد نیاز، استفاده کرد:



(صفحه‌ی اصلی نرم‌افزار سین‌بیلدر)

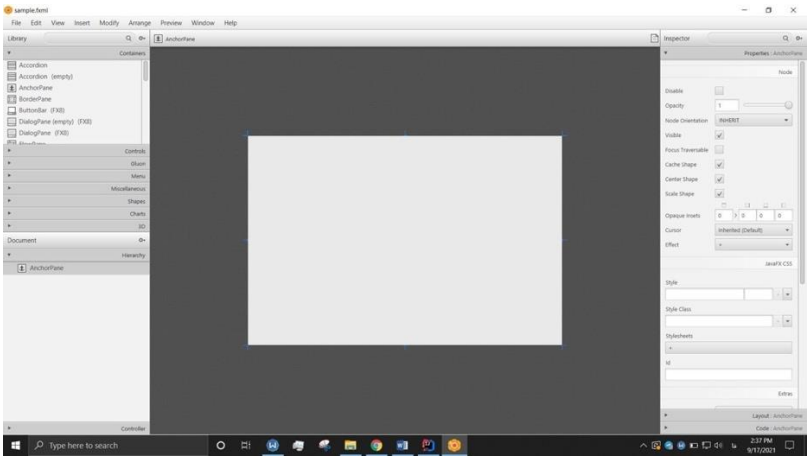
در ادامه، قصد داریم با تعدادی از این عناصر آشنا شویم.

برخی از عناصر سین‌بیلدر

همان‌طور که در منوی سمت چپ سین‌بیلدر مشاهده می‌کنید، این عناصر به چند دسته تقسیم می‌شوند که با کلیک بر روی هر دسته، عناصر مربوط به آن نمایش داده می‌شود.

دسته‌ی اول، کانتینرها[1] هستند؛ عناصری که دیگر اجزا را در بر می‌گیرند و با توجه به نوع کانتینر مورد استفاده، می‌توان چینش‌های گوناگونی را برای اجزای درون آن مشخص کرد.

یکی از عناصر این دسته، AnchorPane است. برای شروع به کار با این کانتینر، ابتدا از منوی Document واقع در پایین سمت چپ صفحه، کانتینری که به طور پیشفرض در scene قرار داده شده را با انتخاب و فشردن Delete پاک کنید و سپس با کشیدن AnchorPane به درون صفحه (Drag and Drop)، آن را به scene خود اضافه کنید:



(اضافه کردن AnchorPane)

همان‌طور که می‌بینید، در منوی سمت راست می‌توان ویژگی‌های عنصر انتخاب شده (در اینجا AnchorPane) را ویرایش کرد.

این منو شامل سه بخش Properties، Layout و Code است که در هر بخش می‌توان به ویژگی‌های مختلف عنصر انتخاب شده دسترسی داشت.

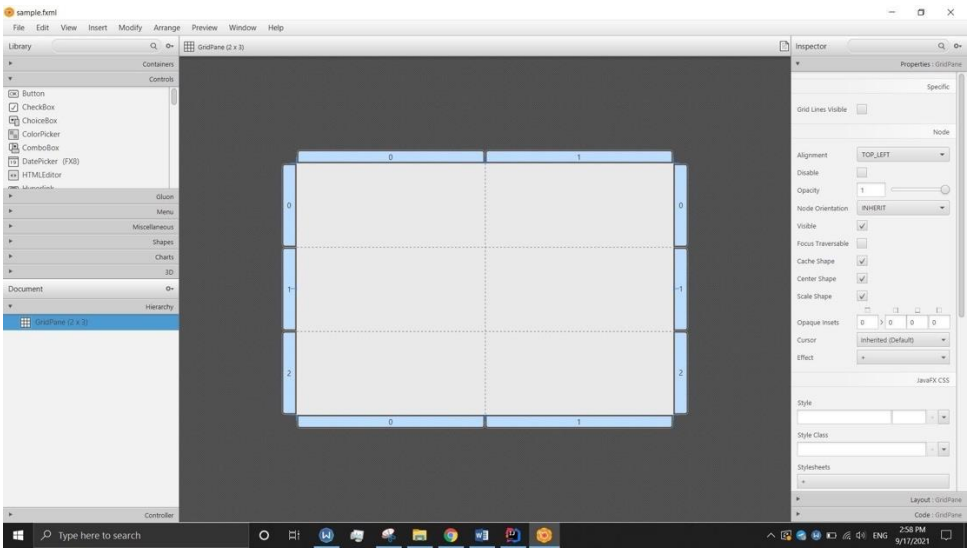
منوی Properties شامل ویژگی‌های ظاهری مثل شفافیت و همچنین استایل‌شیت‌ها[2] است (مبحث استایل‌دهی و CSS جز اهداف این دستورکار نیست اما می‌توانید با جستجو در منابع آموزشی درباره آن بیشتر بیاموزید).

[1] Container
[2] Style Sheets

منوی Layout شامل ویژگی‌های مربوط به اندازه عنصر است و از طریق آن می‌توان طول، عرض و دیگر ویژگی‌های مشابه عنصر مورد نظر را ویرایش کرد.
منوی Code نیز مربوط به مرتبط‌سازی فایل fxml با Controller است و شامل ویژگی‌هایی چون id و تعریف متدهای مربوط به آن است که در بخش بعدی دستور کار، با آن‌ها آشنا خواهیم شد.

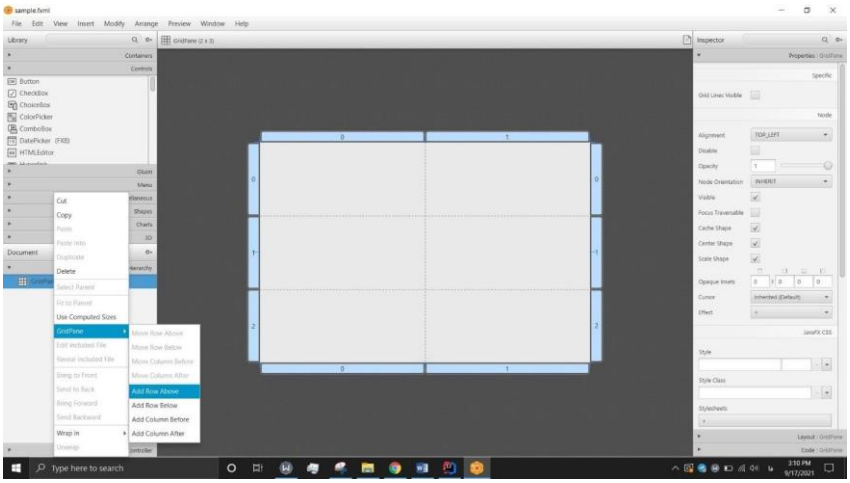
همان‌طور که توضیح داده شد، AnchorPane یک کانتینر ساده است و هیچ تقسیم‌بندی‌ای به بخش‌های کوچک‌تری در آن وجود ندارد، اما گاهی نیاز داریم کانتینرهایی داشته باشیم که به ما در تقسیم فضا کمک کنند، برای این هدف می‌توانیم از BorderPane، GridPane و ... استفاده کنیم.

GridPane، یکی از این کانتینرها است که به طور پیش‌فرض دو ستون و سه سطر دارد، اما می‌توان تعداد آن‌ها را بسته به نیاز، تغییر داد:



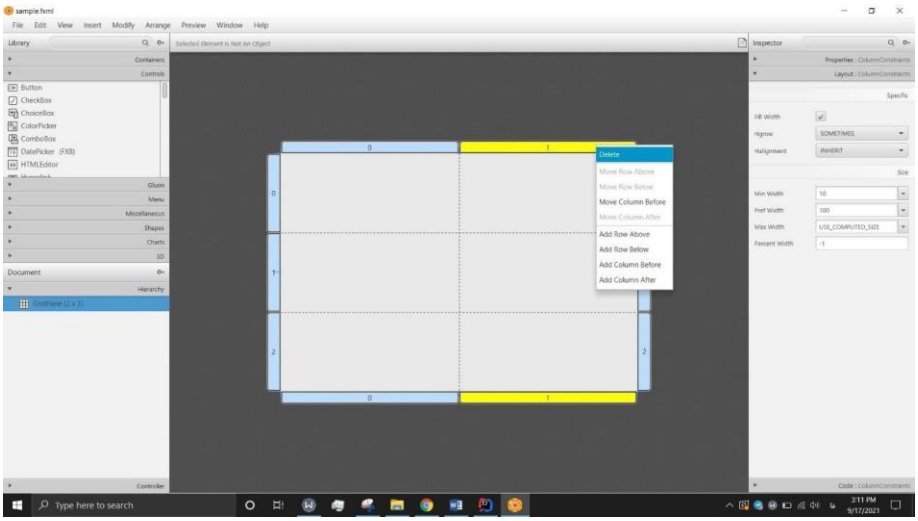
(یک GridPane پیش‌فرض)

برای این کار، می‌توان بر روی GridPane در بخش Document کلیک راست کرده و از منوی ظاهر شده، با انتخاب گزینه‌های Add Row و Add Column، سطر و یا ستون جدید اضافه کرد:



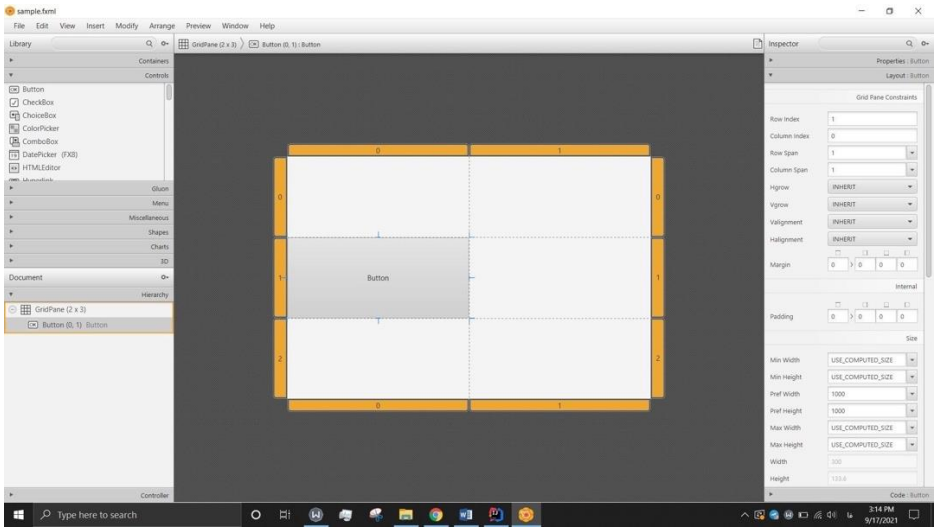
(اضافه کردن سطر و یا ستون جدید)

همچنین با انتخاب هر سطر و یا ستون و کلیک راست کردن بر روی آن و انتخاب گزینه‌ی Delete، می‌توان هر یک از آن‌ها را حذف کرد:



(حذف یک ستون)

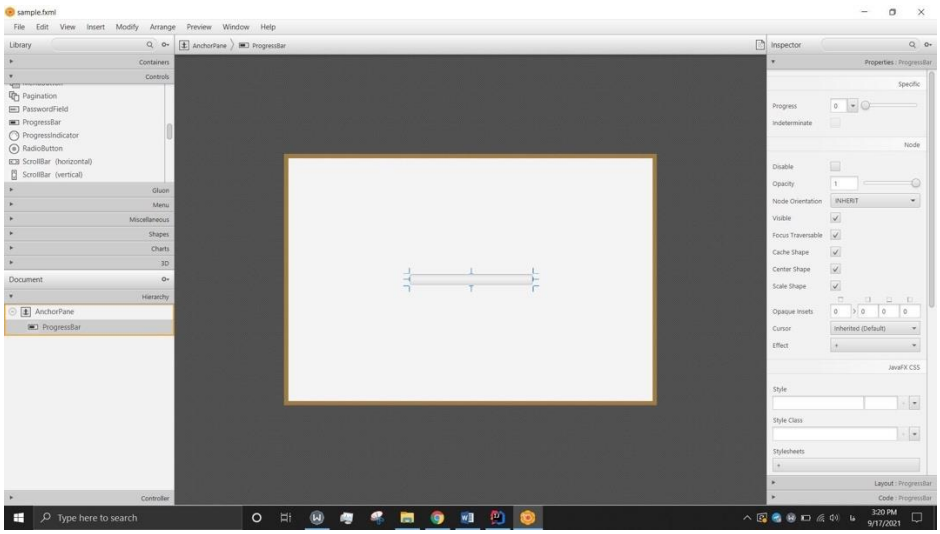
همان‌طور که در شکل بالا مشخص است، بعد از اضافه کردن GridPane، فضای scene به چند بخش تقسیم شده که عناصر درون هر بخش، نمی‌توانند از بخش مربوط به خود خارج شوند و بیشترین اندازه‌ای که می‌توانند داشته باشند، به اندازه فضایی است که به آن‌ها داده شده است:



(محدود بودن اندازه‌ی اجزای درون GridPane)

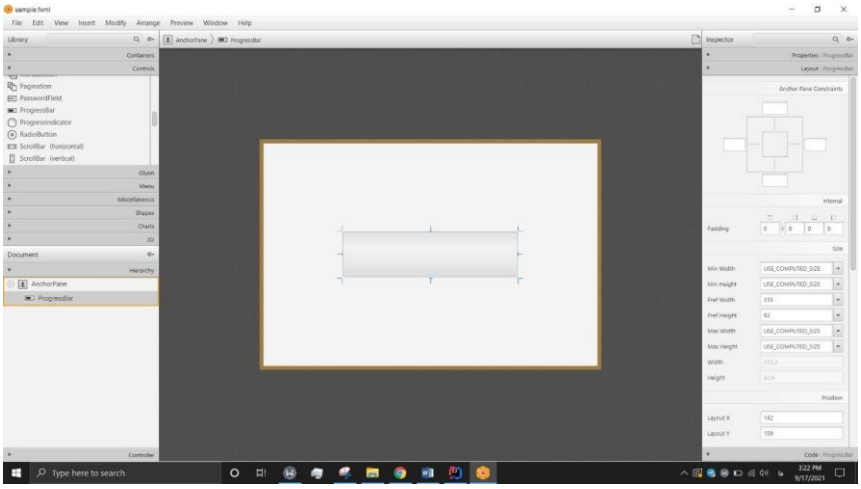
عناصر Controls

دسته‌ای دیگر از اجزای JavaFX، عناصر Controls هستند که شامل اجزایی چون Button، Label، Hyperlink و ... است. در این دستورکار می‌خواهیم با ProgressBar آشنا شویم:



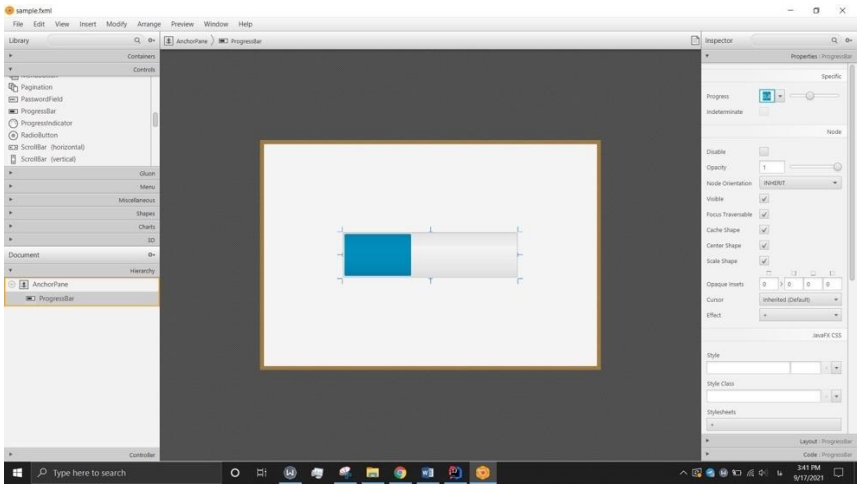
(نمونه‌ی یک ProgressBar)

می‌توان اندازه‌ی ProgressBar را با کشیدن گوشه‌های آبی‌رنگ اطراف آن و یا منوی Layout تغییر داد:



(تغییر اندازه‌ی ProgressBar)

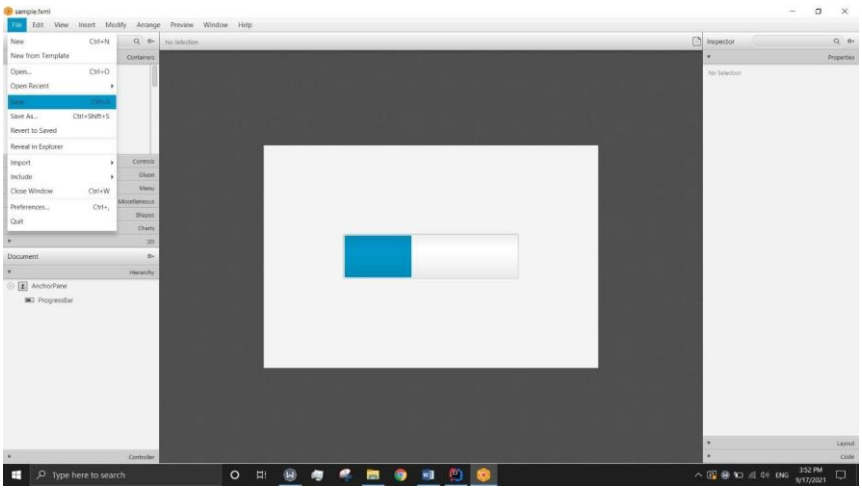
همچنین مشابه عناصر قبلی، از منوی Properties می‌توان ویژگی‌های کلی و ظاهری را تعیین کرد، برای مثال می‌توان از بخش Specific، مقدار پیشفرضی (بین صفر و یک) برای میزان پیشروی ProgressBar در نظرگرفت:



(تغییر مقدار پیش‌فرض ProgressBar)

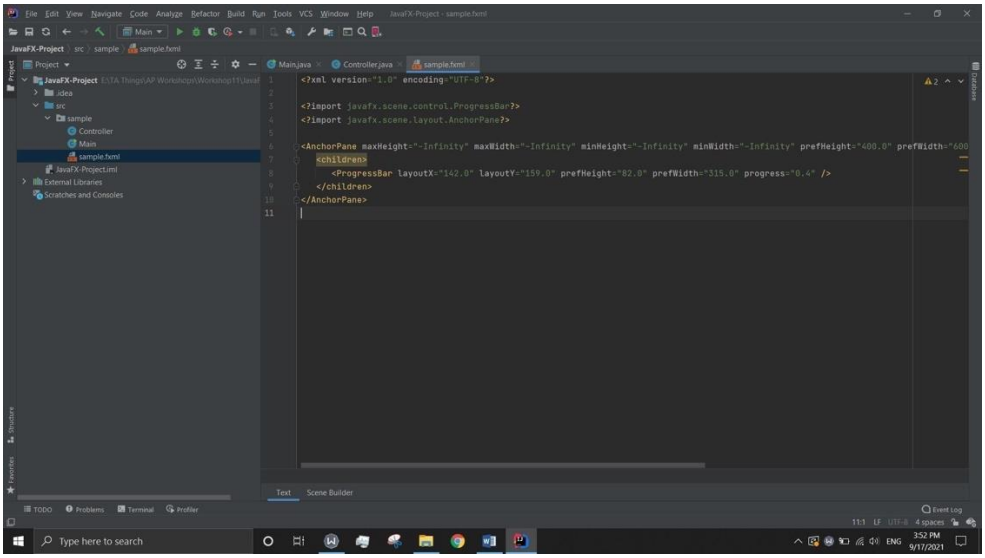
برنامه‌ریزی برای تغییر میزان پیشروی ProgressBar در طول اجرای برنامه، در کلاس کنترلر مربوط به آن انجام می‌شود که در ادامه با آن‌ها آشنا خواهیم شد.

هر زمان که scene را به شکل مورد نظر خود طراحی و ویرایش کردیم، می‌توان از منوی File، گزینه Save را انتخاب کرده تا تغییرات بر روی فایل fxml اعمال شود:



(ذخیره کردن scene)

همان‌طور که در تصویر زیر می‌بینید، همه‌ی عناصر مورد استفاده با جزئیات تعیین شده در فایل fxml مورد نظر ثبت شده‌اند:



(فایل fxml تولید شده)

آشنایی با رویدادها[1]

معرفی رویداد

رویداد، یک اطلاعیه درباره‌ی جزئیات تغییرات رخ داده می‌باشد. در واقع هر نوع ارتباط بین کاربر و برنامه یک رویداد هست. برای مثال فشردن کیبورد، حرکت موس و یا کلیک کردن دکمه‌ای در برنامه و...، همگی نوعی رویداد هستند. در برخی موارد، حتی کاربر هم در این ریوداد به طور کامل دخیل نمی‌باشد، مانند ساعتی که پس مدتی فعال می‌شود.

انواع رویدادها

رویدادها به طور کلی به دو دسته‌ی foreground و background تقسیم می‌شوند.

نوع اول مربوط به اتفاقاتی است که در ارتباط مستقیم با کاربر رخ می‌دهد، مانند کشیدن و انداختن فایل یا کلیک کردن با موس و نگه داشتن یا رها کردن موس، انتخاب کردن گرینه‌ای از لیست و ... . نوع دوم اما در ارتباط مستقیم با کاربر نیست و بلکه در اثر اعمال انجام شده توسط کاربر رخ می‌دهد، مانند تمام شدن زمان یک تایمر، ارورهای رخ داده، قطع و یا وصل شدن اینترنت و ... .

انواع رویدادهای JavaFx

در پکیج javafx.event، طیف وسیعی از رویدادهای قابل هندل کردن وجود دارد که به برخی از آن‌ها می‌پردازیم:

- Mouse event: این رویدادها در ارتباط با تغییرات موس هستند، نظیر فشردن (نگه داشتن)، رها کردن، کلیک کردن، جابجایی، وارد محدوده‌ای شدن و ... که در کلاس MouseEvent قرار دارند.
- Key event: این گروه، مربوط به تغییرات دکمه‌های کیبورد مانند فشردن (نگه داشتن)، رها کردن و کلیک کردن می‌باشد که در کلاس KeyEvent قرار دارند.
- Drag event: این گروه، مربوط به رویدادهای کشیدن (drag) است. مانند کشیدن به داخل محدوده‌ای یا خارج شدن از آن، رها کردن در نقطه‌ای و ... که در کلاس DragEvent قرار دارند.

[1] Events

هندل[1] کردن رویدادها

برای هندل کردن رویداد اتفاق افتاده برای یک شئ، باید نوع رویداد و eventhandlerای که کد مد نظر برای هندل کردن را دارد به متد addEventHandler شئ مد نظر پاس دهیم. این کار به دو شکل زیر انجام می‌شود:

```
@FXML
public void initialize(){
    myNode.addEventHandler(eventType, new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent mouseEvent) {
//              code to handle the event
        }
    });
}
```

(روش اول)

```
EventHandler<MouseEvent> eventHandler = new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent mouseEvent) {
//              code to handle the event
    }
};

@FXML
public void initialize(){
    myNode.addEventHandler(eventType, eventHandler);
}
```

(روش دوم)

معمولاً از روش اول استفاده می‌شود، چون برای هندل کردن رویدادهای دو شئ متفاوت، کدهای متفاوتی نیاز است و خیلی قابلیت استفاده‌ی مجدد ندارد.

[1] Handle

هندل کردن رویداد با Convenience Methods

در برخی از کنترل‌ها در JavaFx، برای هر رویدادی یک EventHandler وجود دارد که با استفاده از ستر آن، می‌توان برای آن رویداد، یک Hanlder اضافه کرد. به این متدها Convenience Methods گفته می‌شود:

```
@FXML
public void initialize(){
    myNode.setOnMouseClicked(new EventHandler<MouseEvent>() {
        @Override
        public void handle(MouseEvent mouseEvent) {
//              code to handle event
        }
    });
}
```

(نمونه‌ای از یک Convenience Method)

همچنین برای مطالعه‌ی بیشتر درباره‌ی رویدادهای JavaFx، می‌توانید به JavaFX events و oracle events handling مراجعه کنید.

نکته: در اصل رویدادها علاوه بر انواعی که دارند، حاوی اطلاعات بیشتری مانند مبدأ و هدف نیز هستند که برای بررسی این موارد می‌توانید به این لینک مراجعه کنید.

معرفی معماری MVC

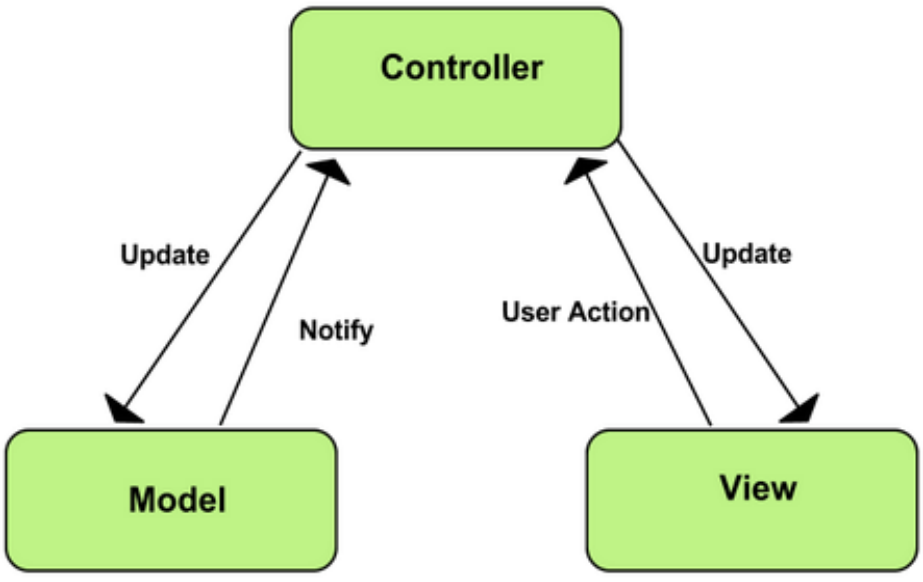
Model-View-Controller Pattern یا به اختصار MVC یک الگوی طراحی است که برای جداسازی بخش‌های مختلف برنامه استفاده می‌شود. به این شکل، مدیریت بخش‌های مختلف برنامه آسان‌تر می‌شود و به طور کلی، توسعه‌ی برنامه سریع‌تر خواهد بود.

در این الگو، طراحی برنامه به سه بخش Model، View و Controller تقسیم می‌شود.

Model شامل تمام اطلاعات و داده‌های برنامه می‌باشد. در واقع می‌توان گفت منطق مربوط به پردازش و دسترسی داده در Model پیاده‌سازی شده است.

View تنها یک نمایش از داده‌های Model می‌باشد و بخشی از برنامه است که کاربر به طور مستقیم با آن ارتباط دارد. در View به خودی خود هیچ منطقی وجود ندارد.

Controller در واقع پلی میان Model و View است. Controller بررسی و کنترل کردن تغییرات Model و بروزرسانی View براساس تغییرات Model را بر عهده دارد. معمولا Model و View با هم ارتباط مستقیمی ندارند و این Controller است که میان این دو ارتباط برقرار می‌کند:



(ساختار معماری MVC)

باید توجه کرد که JavaFX از الگوی طراحی MVC استفاده می‌کند، به همین دلیل استفاده درست از این الگوی طراحی در پروژه‌های JavaFX اهمیت بسزایی دارد.

در یک پروژه‌ی JavaFX، فایل‌های fxml و همچنین المان‌هایی که با استفاده از کد در جریان برنامه ایجاد می‌شوند نقش View را ایفا می‌کنند. هر فایل fxml دارای یک فایل کنترلر می‌باشد. واضح است که این فایل‌ها نقش Controller را ایفا می‌کنند. در یک فایل کنترلر امکان ایجاد تغییرات در View وجود دارد. همچنین event handling یکی از وظایف کنترلر می‌باشد.

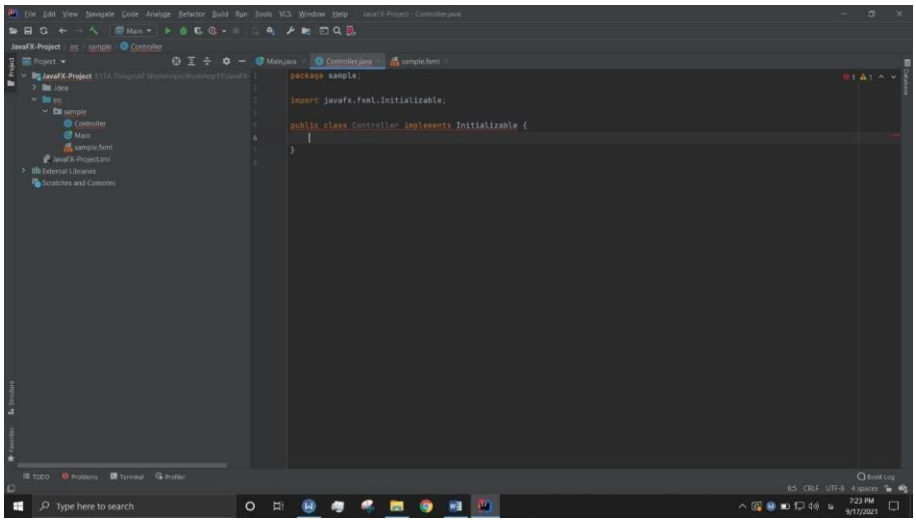
یکی از اشتباهات رایج در استفاده از JavaFX، پیاده‌سازی منطق برنامه (البته نه view logic) در کنترلرها است. توجه کنید که کنترلر تنها وظیفه‌ی ایجاد ارتباط میان Model و View را دارد و هرگونه مسئولیت اضافی مانند کار با داده‌ها و منطق برنامه به عهده‌ی Model است.

معرفی اینترفیس Initializable

Initializable، اینترفیسی است که به عنوان بخشی از فریم‌ورک JavaFX، می‌تواند به ما در طراحی برنامه‌های گرافیکی با این ابزار کمک کند.

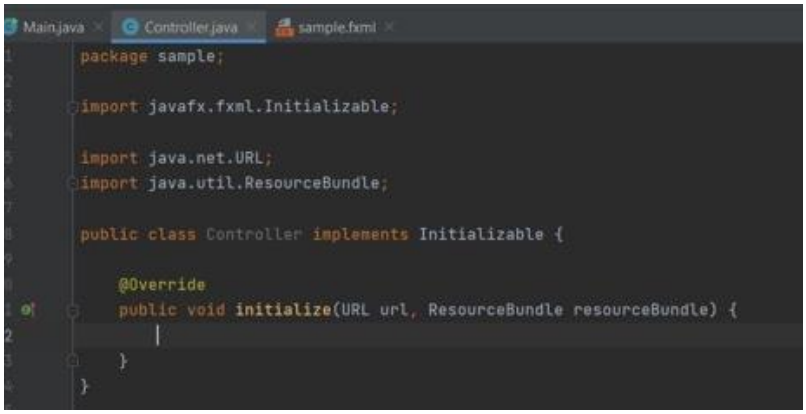
این اینترفیس که در پکیج javafx.fxml قرار دارد، می‌تواند توسط کلاس‌های کنترلرهای مربوط به scene ها پیاده‌سازی شود.

برای استفاده از این اینترفیس، ابتدا باید آن را در کنترلر مورد نظر پیاده‌سازی کنیم:



(پیاده‌سازی اینترفیس Intializable)

همان‌طور که در تصویر بالا مشخص است، IDE وجود خطایی را به شما هشدار می‌دهد. این خطا ناشی از این است که همه کلاس‌هایی که این اینترفیس را پیاده‌سازی می‌کنند، باید متد initialize را override کنند:



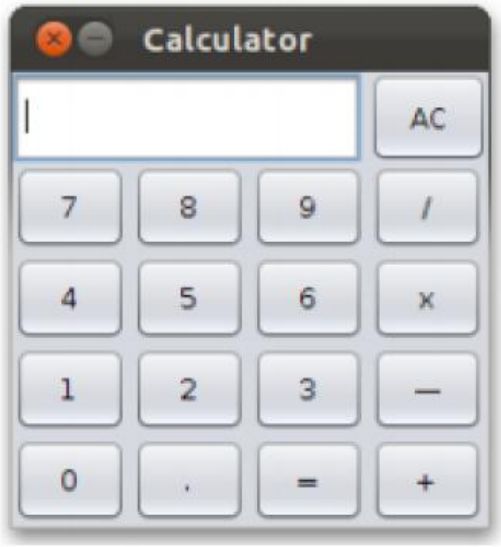
(override کردن متد initialize)

کاربرد این اینترفیس زمانی است که می‌خواهیم برای زمان نمایش scene بر روی stage، برنامه‌ریزی کنیم؛ به طوری که هر زمان نمایش scene بر روی صفحه آغاز شود، در صورتی که کلاس کنترلر آن اینترفیس، Initializable را پیاده‌سازی کرده باشد، متد initialize اجرا می‌شود و می‌توان از آن برای مقداردهی اولیه فیلدها، تعیین ویژگی‌های عناصر گرافیکی و دیگر اعمال منطقی استفاده کرد.

انجام دهید: ماشین‌حساب گرافیکی

در این جلسه قصد داریم به وسیله‌ی JavaFx، یک ماشین‌حساب گرافیکی را پیاده‌سازی کنیم. این ماشین‌حساب باید قابلیت‌های زیر را داشته باشد:

۱. قابلیت محاسبه‌ی چهار عمل اصلی را داشته باشد.
۲. علاوه بر موس، به وسیله کیبورد نیز قابل کنترل باشد (KeyEvent ها را نیز هندل کنید). برای مثال، اگر کاربر دکمه «۱» کیبورد را فِشرد، مانند آن باشد که روی دکمه‌ی ۱ ماشین‌حساب کلیک کرده است.
۳. ماشین حساب، دارای ظاهری کاربر پسند باشد.



(نمونه‌ی یک ماشین‌حساب گرافیکی)

کد خود را بر روی یک ریپازیتوری با نام «AP-Workshop11» قرار دهید.