Kelas : BU11  
Kelompok : 2

Anggota:

William Dominick Lim / 2702290486

Evannail Jeremy / 2702301691

Wilsen ardian budianto / 2702287333

Willie Winardi / 2702297233

**Class: Product**



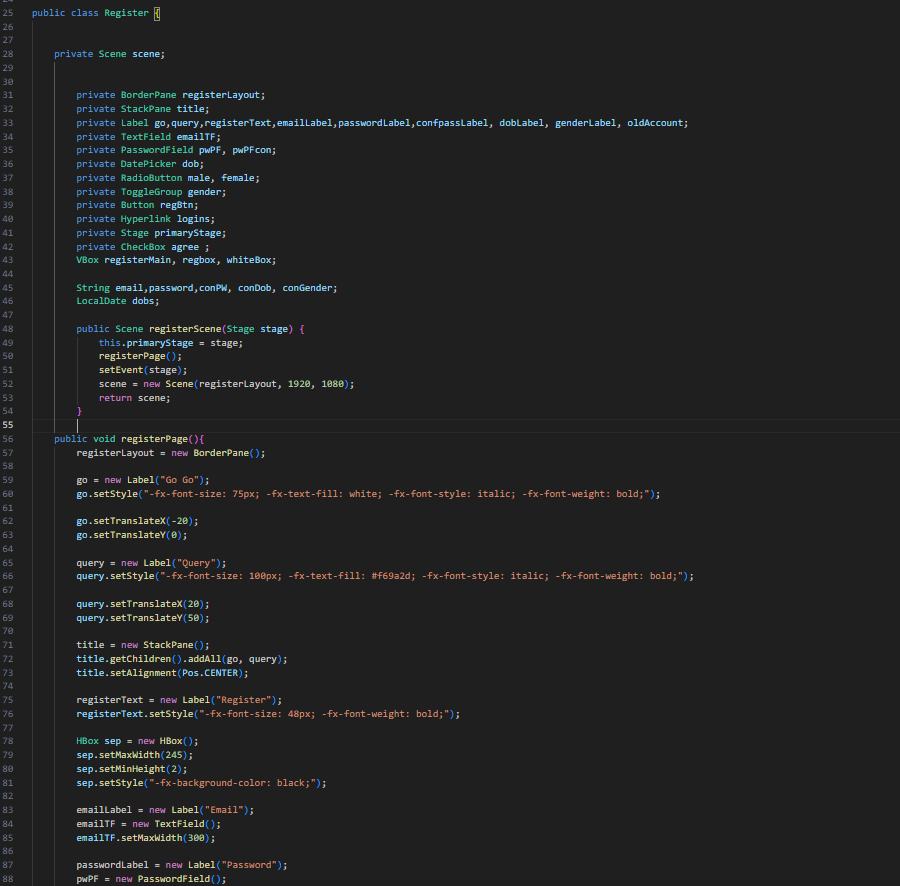
Atribut:

* private int id = Id unit dari product
* Private String name = Nama produk
* Private String category = Kategori produk
* Private double price = Harga produk
* Private String desc = Deskripsi produk
* Private int stock = Jumlah stok produk yang tersedia

Konstruktor = Menginisialisasi objek Product dengan nilai-nilai yang diberikan.

Metode getter = Mengembalikan atribut yang ada seperti return id, return name, return category, return price, return desc, return stock

**Class: Register**



Komponen Utama:

UI Components:

* Label: Digunakan untuk menampilkan teks, seperti judul halaman dan label untuk setiap kolom input (email, password, dll).
* TextField, PasswordField: Kolom input untuk email dan kata sandi.
* DatePicker: Untuk memilih tanggal lahir pengguna.
* RadioButton, ToggleGroup: Untuk memilih jenis kelamin (Laki-laki atau Perempuan).
* Button: Tombol untuk memulai proses pendaftaran.
* Hyperlink: Link untuk mengarahkan pengguna ke halaman login jika mereka sudah memiliki akun.
* CheckBox: Untuk menyetujui syarat dan ketentuan pendaftaran.
* Layout = BorderPane, StackPane, HBox, VBox = Layout komponen antarmuka untuk menyusun elemen-elemen UI di layar.
* Data = String email, password, conPW, conDob, conGender: Variabel untuk menampung data input pengguna.
* LocalDate dobs = Untuk menyimpan tanggal lahir yang dipilih oleh pengguna.

Metode registerScene(Stage stage):

* Menginisialisasi dan menyiapkan tampilan halaman pendaftaran.
* Menyusun halaman dengan memanggil metode registerPage() untuk merancang antarmuka dan setEvent(stage) untuk mengonfigurasi event handler.
* Mengembalikan objek Scene untuk ditampilkan pada primaryStage aplikasi.

Metode registerPage():

* Mengonfigurasi elemen-elemen UI untuk halaman pendaftaran:
* Menambahkan label judul "Go Go Query" di bagian atas halaman dengan gaya tertentu.
* Menambahkan label dan kolom input untuk setiap informasi yang dibutuhkan (email, kata sandi, konfirmasi kata sandi, tanggal lahir, jenis kelamin, dll).
* Menambahkan checkbox untuk persetujuan syarat dan ketentuan serta tombol untuk mendaftar.

Metode setEvent(Stage stage):

* Mengonfigurasi pendengar acara untuk tombol "Register" dan hyperlink "Login".
* Tombol "Register" akan memvalidasi input pengguna, memastikan bahwa semua kolom terisi dengan benar.
* Jika input valid, maka proses pendaftaran dilanjutkan. Jika tidak, akan muncul pesan kesalahan.

Alur Kerja:

Membuat Layout Halaman:

* Layout halaman terdiri dari label judul "Go Go" dan "Query" yang ditata menggunakan StackPane, diikuti dengan formulir pendaftaran.
* Formulir pendaftaran memuat kolom input untuk email, kata sandi, konfirmasi kata sandi, tanggal lahir, jenis kelamin, dan checkbox untuk syarat dan ketentuan.
* Setelah pendaftaran berhasil, pengguna akan diarahkan untuk memilih peran mereka (seperti Manager atau Shopper).

Proses Pendaftaran:

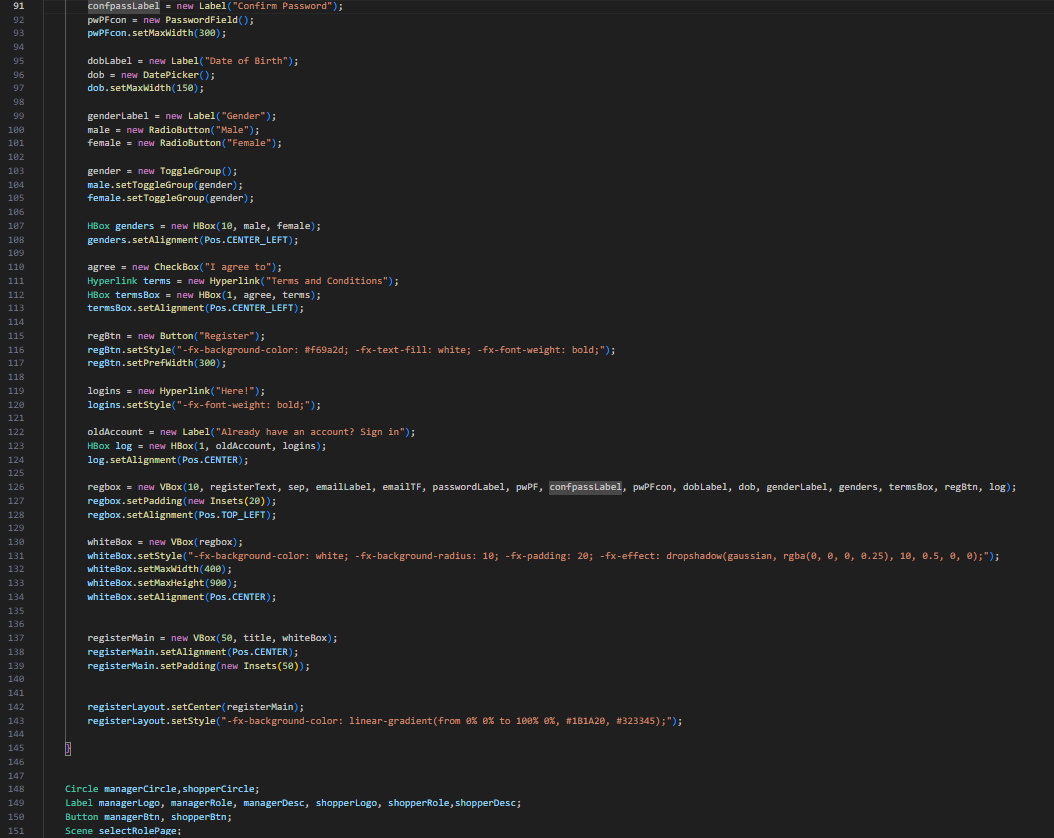
* Pengguna diminta untuk mengisi kolom email, kata sandi, tanggal lahir, dan memilih jenis kelamin, serta menyetujui syarat dan ketentuan.
* Setelah tombol "Register" diklik, aplikasi akan memvalidasi input:
* Format email harus valid.
* Kata sandi harus cukup kuat (misalnya alfanumerik).
* Tanggal lahir harus menunjukkan bahwa pengguna berusia lebih dari 17 tahun.
* Jenis kelamin harus dipilih.
* Checkbox syarat dan ketentuan harus dicentang.
* Jika validasi berhasil, pengguna akan dibawa ke halaman pemilihan peran. Jika gagal, pesan kesalahan akan muncul.

Navigasi ke Halaman Login = Setelah pendaftaran berhasil, pengguna dapat diarahkan ke halaman login untuk masuk menggunakan akun yang baru dibuat.

Penanganan Kesalahan:

Validasi Input:

* Jika ada kesalahan pada format email, kata sandi tidak valid, atau kolom lainnya tidak diisi, maka aplikasi akan menampilkan pesan kesalahan.
* Jika tanggal lahir menunjukkan bahwa pengguna belum cukup umur (kurang dari 17 tahun), maka pesan kesalahan juga akan ditampilkan.
* Jika checkbox syarat dan ketentuan tidak dicentang, pengguna akan diberi tahu untuk menyetujui syarat sebelum melanjutkan.



Konfirmasi Kata Sandi:

* Label confpassLabel: Menampilkan teks "Confirm Password" untuk meminta pengguna memasukkan kata sandi kedua untuk konfirmasi.
* PasswordField pwPFcon: Kolom input untuk kata sandi konfirmasi, dengan lebar maksimum 300 piksel.

Tanggal Lahir:

* Label dobLabel: Menampilkan teks "Date of Birth" untuk meminta pengguna memilih tanggal lahir.
* DatePicker dob: Kolom input untuk memilih tanggal lahir, dengan lebar maksimum 150 piksel.

Jenis Kelamin:

* Label genderLabel: Menampilkan teks "Gender" untuk meminta pengguna memilih jenis kelamin.
* RadioButton male, female: Pilihan jenis kelamin dengan dua opsi, yaitu Laki-laki dan Perempuan.
* ToggleGroup gender: Mengelompokkan kedua radio button tersebut agar hanya satu yang bisa dipilih.
* HBox genders: Menata kedua radio button dalam baris dengan jarak 10 piksel dan menyelaraskan mereka di sebelah kiri.

Syarat dan Ketentuan:

* CheckBox agree: Pengguna harus mencentang kotak ini untuk menyetujui syarat dan ketentuan.
* Hyperlink terms: Menampilkan teks "Terms and Conditions" yang dapat diklik untuk melihat detail syarat dan ketentuan.
* HBox termsBox: Menata checkbox dan hyperlink dalam satu baris dengan jarak 1 piksel, diselaraskan ke kiri.

Tombol Pendaftaran:

* Button regBtn: Tombol untuk mengonfirmasi pendaftaran dengan gaya warna latar belakang oranye dan teks putih.
* Hyperlink logins: Tautan untuk mengarahkan pengguna ke halaman login jika mereka sudah memiliki akun.
* Label oldAccount: Label yang menampilkan teks "Already have an account? Sign in" untuk memberi tahu pengguna yang sudah memiliki akun.
* HBox log: Menata label dan hyperlink login dalam satu baris dengan jarak 1 piksel, diselaraskan ke tengah.

Struktur Layout Halaman Pendaftaran:

* VBox regbox: Menampung semua elemen UI pada halaman pendaftaran (seperti kolom input, tombol, dan checkbox) dalam sebuah kolom vertikal dengan jarak antar elemen 10 piksel.
* VBox whiteBox: Membungkus regbox dengan gaya latar belakang putih, radius sudut 10, dan efek bayangan agar tampak lebih menarik. Dimensinya dibatasi dengan lebar dan tinggi maksimum.
* VBox registerMain: Menampung elemen title dan whiteBox dalam sebuah kolom vertikal untuk pusat halaman, dengan padding 50 piksel.

Penataan Background dan Warna:

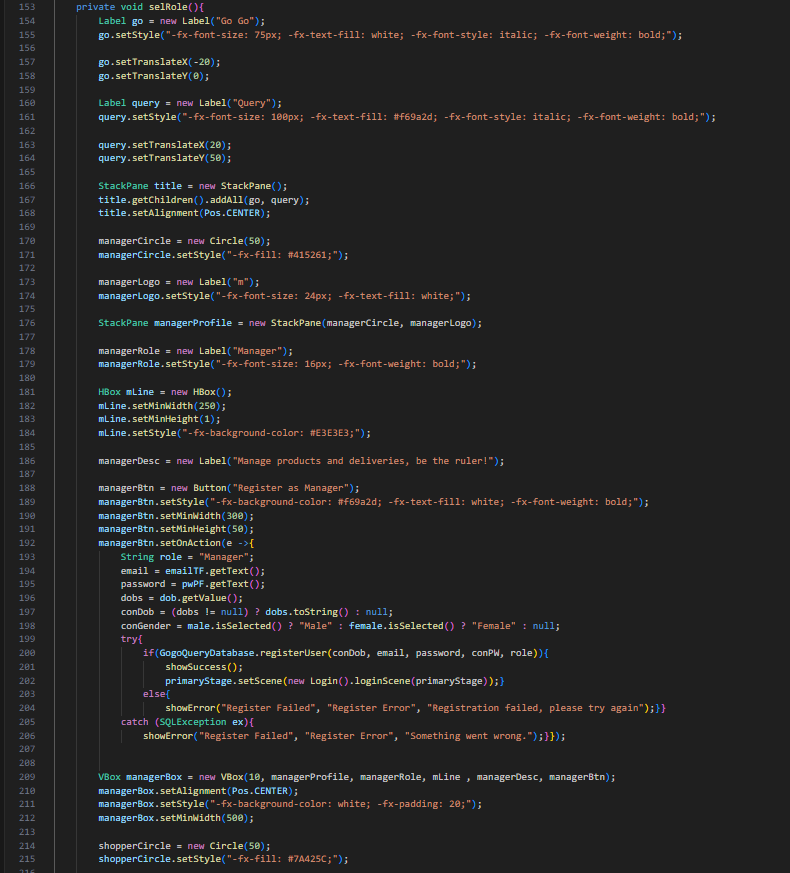
* Background Halaman: Menggunakan linear-gradient dari warna abu-abu gelap ke biru gelap untuk latar belakang halaman pendaftaran, memberikan tampilan yang elegan.

Variabel untuk Peran:

* Circle managerCircle, shopperCircle: Lingkaran yang mewakili pilihan peran Manager dan Shopper.
* Label managerLogo, shopperLogo: Menampilkan logo untuk peran Manager dan Shopper.
* Label managerRole, shopperRole: Menampilkan nama peran untuk Manager dan Shopper.
* Label managerDesc, shopperDesc: Menampilkan deskripsi singkat tentang peran Manager dan Shopper.

Tombol Pemilihan Peran:

* Button managerBtn, shopperBtn: Tombol untuk memilih peran Manager atau Shopper.
* Scene Pemilihan Peran:
* Scene selectRolePage: Scene yang menampilkan pemilihan peran, setelah pengguna berhasil mendaftar.



Tampilan Judul:

* Label go: Menampilkan teks "Go Go" dengan gaya font besar, berwarna putih, miring, dan tebal.
* Label query: Menampilkan teks "Query" dengan gaya font besar berwarna oranye (#f69a2d), miring, dan tebal.
* StackPane title: Menyusun label go dan query secara tumpuk dan diselaraskan di tengah.

Pemilihan Peran (Manager):

* Lingkaran managerCircle: Lingkaran berwarna abu-abu dengan radius 50 untuk menandakan profil Manager.
* Label managerLogo: Menampilkan huruf "m" sebagai logo untuk peran Manager, dengan gaya font putih dan ukuran 24px.
* StackPane managerProfile: Menampung lingkaran managerCircle dan logo managerLogo dalam sebuah stack.
* Label managerRole: Menampilkan teks "Manager" untuk menunjukkan peran Manager.
* HBox mLine: Garis pemisah dengan lebar minimal 250 piksel dan tinggi 1 piksel, berwarna abu-abu terang (#E3E3E3).
* Label managerDesc: Deskripsi peran Manager, "Manage products and deliveries, be the ruler!".
* Button managerBtn: Tombol untuk mendaftarkan pengguna sebagai Manager dengan teks "Register as Manager", dengan gaya latar belakang oranye (#f69a2d) dan teks putih. Ketika tombol ditekan, aplikasi akan mengambil data pengguna (email, kata sandi, tanggal lahir, dan jenis kelamin), kemudian mencoba mendaftarkan pengguna melalui fungsi GogoQueryDatabase.registerUser(). Jika berhasil, akan muncul pesan sukses dan aplikasi akan beralih ke halaman login. Jika gagal, akan muncul pesan error.

Tampilan Peran Shopper:

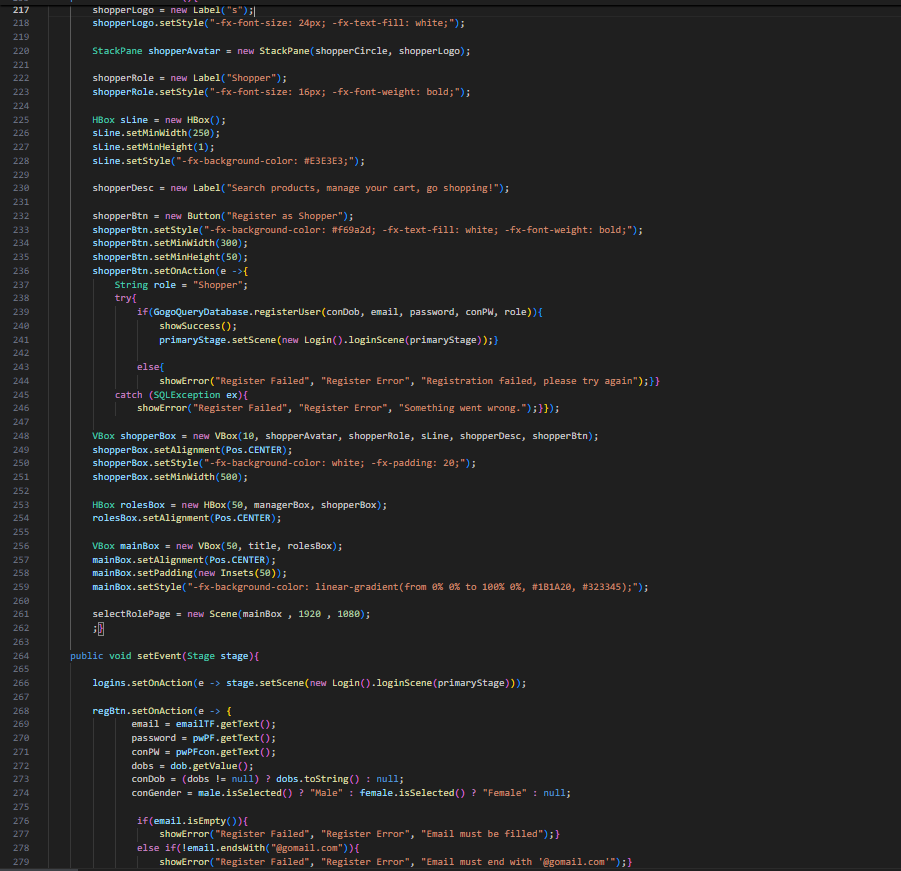
* Lingkaran shopperCircle: Lingkaran berwarna merah muda dengan radius 50 untuk menandakan profil Shopper.
* Label shopperLogo: Menampilkan huruf "s" sebagai logo untuk peran Shopper, dengan gaya font putih dan ukuran 24px.
* StackPane shopperProfile: Menampung lingkaran shopperCircle dan logo shopperLogo dalam sebuah stack.
* Label shopperRole: Menampilkan teks "Shopper" untuk menunjukkan peran Shopper.
* HBox sLine: Garis pemisah dengan lebar minimal 250 piksel dan tinggi 1 piksel, berwarna abu-abu terang (#E3E3E3).
* Label shopperDesc: Deskripsi peran Shopper, "Browse products, and make purchases!".
* Button shopperBtn: Tombol untuk mendaftarkan pengguna sebagai Shopper dengan teks "Register as Shopper", dengan gaya latar belakang merah (#7A425C) dan teks putih. Ketika tombol ditekan, aplikasi akan mengambil data pengguna (email, kata sandi, tanggal lahir, dan jenis kelamin), kemudian mencoba mendaftarkan pengguna melalui fungsi GogoQueryDatabase.registerUser(). Jika berhasil, akan muncul pesan sukses dan aplikasi akan beralih ke halaman login. Jika gagal, akan muncul pesan error.

Struktur Layout:

* VBox managerBox: Menampung semua elemen terkait peran Manager (profil, deskripsi, tombol) dalam sebuah kolom vertikal dengan jarak antar elemen 10 piksel. Warna latar belakang kotak ini adalah putih dengan padding 20 piksel dan lebar minimal 500 piksel.
* VBox shopperBox: Menampung semua elemen terkait peran Shopper dalam kolom vertikal yang serupa dengan managerBox.
* StackPane roleSelection: Menyusun kotak managerBox dan shopperBox di tengah halaman untuk memungkinkan pengguna memilih peran mereka.

Pemrosesan Pendaftaran Pengguna:

* Ketika tombol "Register as Manager" atau "Register as Shopper" ditekan, aplikasi akan mengumpulkan informasi dari formulir pendaftaran yang telah diisi sebelumnya (seperti email, kata sandi, tanggal lahir, dan jenis kelamin), kemudian mencoba mendaftarkan pengguna menggunakan metode GogoQueryDatabase.registerUser().
* Metode registerUser(): Menyimpan data pengguna ke dalam database. Jika pendaftaran berhasil, pengguna akan diarahkan ke halaman login. Jika gagal, pesan kesalahan akan ditampilkan.



Komponen Utama:

Pemilihan Peran Shopper:

* Lingkaran shopperCircle: Lingkaran berwarna merah muda (#7A425C) dengan radius 50 yang menunjukkan profil pengguna untuk peran Shopper.
* Label shopperLogo: Menampilkan teks "s" sebagai logo untuk peran Shopper, dengan gaya font putih dan ukuran 24px.
* StackPane shopperAvatar: Menyusun shopperCircle dan shopperLogo dalam satu stack untuk membentuk avatar Shopper.
* Label shopperRole: Menampilkan teks "Shopper" yang menandakan peran yang dipilih oleh pengguna, dengan gaya font 16px dan tebal.
* HBox sLine: Garis pemisah horizontal dengan lebar 250px dan tinggi 1px yang memberikan pemisahan antar elemen, berwarna abu-abu terang (#E3E3E3).
* Label shopperDesc: Deskripsi singkat tentang peran Shopper: "Search products, manage your cart, go shopping!".
* Button shopperBtn: Tombol untuk mendaftarkan pengguna sebagai Shopper. Ketika tombol ini ditekan, aplikasi akan mengambil data dari formulir pendaftaran, seperti email, kata sandi, tanggal lahir, dan jenis kelamin, kemudian mencoba untuk mendaftarkan pengguna ke dalam basis data dengan peran "Shopper". Jika pendaftaran berhasil, pengguna diarahkan ke halaman login. Jika gagal, muncul pesan kesalahan.

Tampilan Peran Manager:

* Lingkaran managerCircle dan Label managerLogo untuk menunjukkan profil dan logo peran Manager, sama seperti pada bagian sebelumnya untuk Shopper.
* Label managerRole dan Label managerDesc menampilkan informasi tentang peran Manager, yaitu "Manage products and deliveries, be the ruler!".
* Button managerBtn: Tombol pendaftaran untuk peran Manager. Ketika ditekan, aplikasi akan memproses pendaftaran untuk peran Manager dan mengarahkan pengguna ke halaman login jika pendaftaran berhasil, atau menampilkan pesan kesalahan jika gagal.

Tata Letak dan Penataan Visual:

* VBox managerBox: Mengorganisir elemen-elemen terkait peran Manager, dengan jarak antar elemen 10px dan padding 20px.
* VBox shopperBox: Mengorganisir elemen-elemen terkait peran Shopper dengan tampilan yang serupa dengan managerBox.
* HBox rolesBox: Menyusun dua kotak (untuk peran Manager dan Shopper) secara horizontal, dengan jarak antar elemen 50px dan diselaraskan di tengah.
* VBox mainBox: Menyusun elemen-elemen utama, yaitu title dan rolesBox, dalam kolom vertikal dengan padding 50px di sekitar konten dan latar belakang linear gradient yang gelap.

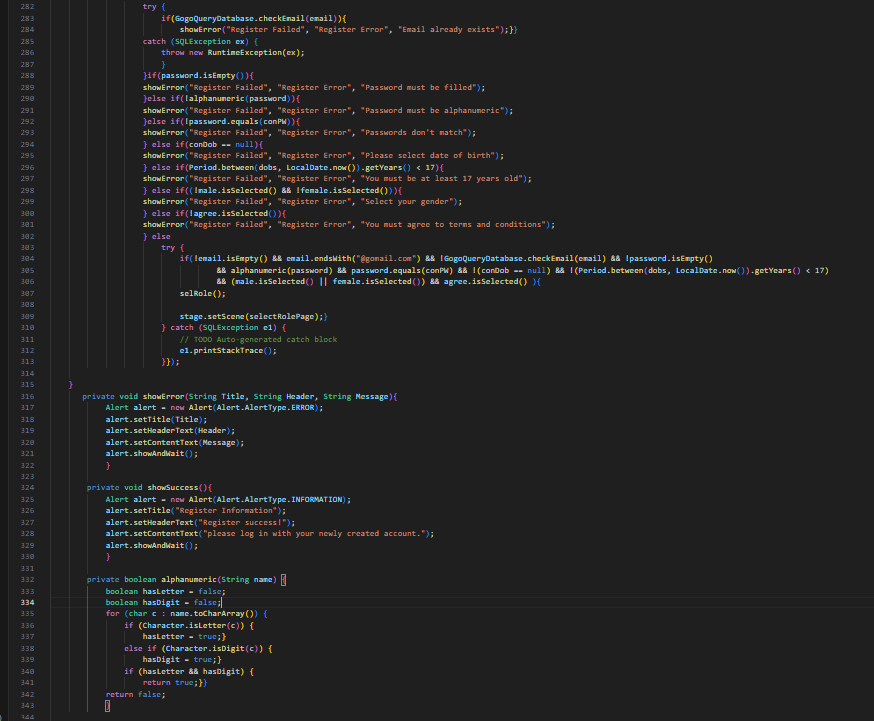
Penanganan Event (Action Handlers):

* Tombol "Register as Shopper" (shopperBtn) = Ketika tombol ini ditekan, data dari form pendaftaran diambil (email, kata sandi, tanggal lahir, jenis kelamin), dan pendaftaran dilakukan menggunakan metode GogoQueryDatabase.registerUser(). Jika pendaftaran berhasil, pengguna diarahkan ke halaman login. Jika gagal, pesan kesalahan ditampilkan.
* Tombol "Register as Manager" (managerBtn) = Prosesnya mirip dengan tombol Shopper, namun dengan peran "Manager". Data diambil dan disubmit ke metode GogoQueryDatabase.registerUser() untuk mendaftarkan pengguna sebagai Manager.
* Tombol "Already have an account? Sign in" (logins) = Jika pengguna sudah memiliki akun, mereka dapat mengklik hyperlink ini untuk beralih ke halaman login.

Validasi Email = Saat tombol pendaftaran ditekan, email harus diisi dan harus memiliki akhiran "@gomail.com". Jika tidak, pesan kesalahan akan ditampilkan.

Validasi Kata Sandi = Jika kata sandi tidak sesuai atau kedua kata sandi (konfirmasi dan utama) tidak cocok, maka pendaftaran tidak akan dilanjutkan dan pesan kesalahan akan muncul.

Validasi Jenis Kelamin dan Tanggal Lahir = Jenis kelamin harus dipilih antara "Male" dan "Female". Tanggal lahir juga harus dipilih.



Validasi Input:

Email:

* Pengecekan Email Kosong: Jika email kosong, pengguna akan diberikan pesan kesalahan "Email must be filled".
* Format Email: Email harus diakhiri dengan @gomail.com. Jika email tidak memenuhi syarat ini, pesan kesalahan "Email must end with '@gomail.com'" ditampilkan.
* Cek Keberadaan Email: Sebelum melanjutkan pendaftaran, aplikasi memeriksa apakah email sudah terdaftar di database menggunakan GogoQueryDatabase.checkEmail(email). Jika email sudah ada, pesan kesalahan "Email already exists" ditampilkan.

Kata Sandi (Password):

* Kata Sandi Kosong: Jika kata sandi kosong, pengguna akan diberikan pesan kesalahan "Password must be filled".
* Alfanumerik: Kata sandi harus mengandung setidaknya satu huruf dan satu angka. Fungsi alphanumeric(password) digunakan untuk memeriksa ini, dan jika kata sandi tidak memenuhi syarat, pesan kesalahan "Password must be alphanumeric" ditampilkan.
* Kata Sandi Tidak Cocok: Jika kata sandi yang dimasukkan tidak cocok dengan konfirmasi kata sandi, maka pesan kesalahan "Passwords don't match" akan muncul.

Tanggal Lahir (Date of Birth):

* Tanggal Lahir Tidak Dipilih: Jika pengguna tidak memilih tanggal lahir, maka pesan kesalahan "Please select date of birth" ditampilkan.
* Usia di Bawah 17 Tahun: Pengguna harus berusia minimal 17 tahun untuk mendaftar. Jika usia kurang dari 17 tahun (dihitung menggunakan Period.between(dobs, LocalDate.now()).getYears()), maka pesan kesalahan "You must be at least 17 years old" ditampilkan.

Jenis Kelamin (Gender):

* Jenis Kelamin Tidak Dipilih: Jika pengguna tidak memilih jenis kelamin (baik male maupun female), maka pesan kesalahan "Select your gender" ditampilkan.
* Persetujuan Syarat dan Ketentuan:
* Persetujuan Tidak Diberikan: Pengguna harus mencentang kotak persetujuan terhadap syarat dan ketentuan. Jika tidak, pesan kesalahan "You must agree to terms and conditions" ditampilkan.

Proses Pendaftaran (Jika Semua Valid):

* Jika semua data valid (email, kata sandi, tanggal lahir, jenis kelamin, dan persetujuan), maka aplikasi akan melanjutkan ke halaman pemilihan peran dengan memanggil metode selRole(). Halaman ini menampilkan pilihan untuk mendaftar sebagai Manager atau Shopper.
* Pengguna akan diarahkan ke halaman pemilihan peran (selectRolePage), yang memungkinkan pengguna untuk memilih peran mereka dan melanjutkan pendaftaran.

Penanganan Kesalahan Pendaftaran:

* Metode showError(): Digunakan untuk menampilkan pesan kesalahan dalam bentuk dialog (alert) jika ada masalah selama proses pendaftaran. Dialog ini berisi judul, header, dan pesan kesalahan.
* Metode showSuccess(): Digunakan untuk menampilkan pesan sukses setelah pendaftaran berhasil. Dialog ini memberi tahu pengguna bahwa pendaftaran berhasil dan meminta mereka untuk masuk dengan akun yang baru dibuat.

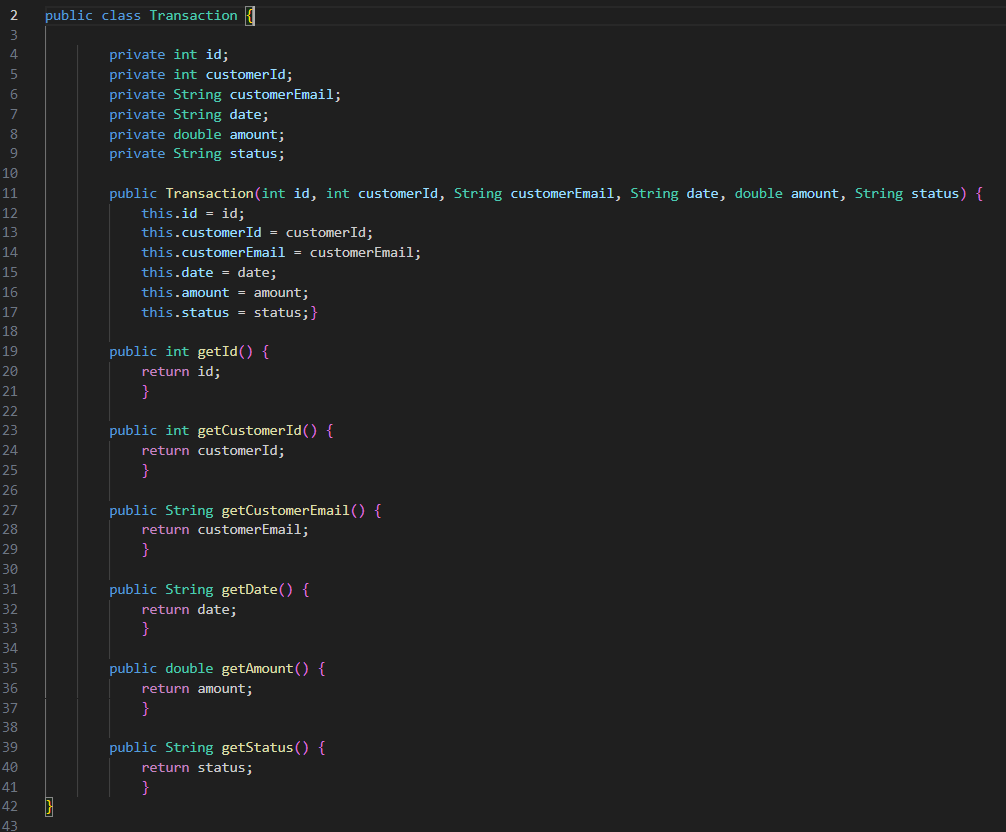
Fungsi alphanumeric():

* Tujuan: Memeriksa apakah kata sandi yang dimasukkan mengandung minimal satu huruf dan satu angka. Fungsi ini digunakan untuk memastikan kekuatan kata sandi.
* Cara Kerja: Fungsi ini mengiterasi setiap karakter dalam string kata sandi dan memeriksa apakah karakter tersebut adalah huruf atau angka. Jika ada minimal satu huruf dan satu angka, fungsi ini mengembalikan nilai true, jika tidak, mengembalikan false.

Penanganan Exception (SQLException):

* SQLException: Jika terjadi kesalahan dalam interaksi dengan basis data (misalnya, saat memeriksa keberadaan email), aplikasi menangkap SQLException dan menampilkan pesan kesalahan umum.

**Class: Transaction**



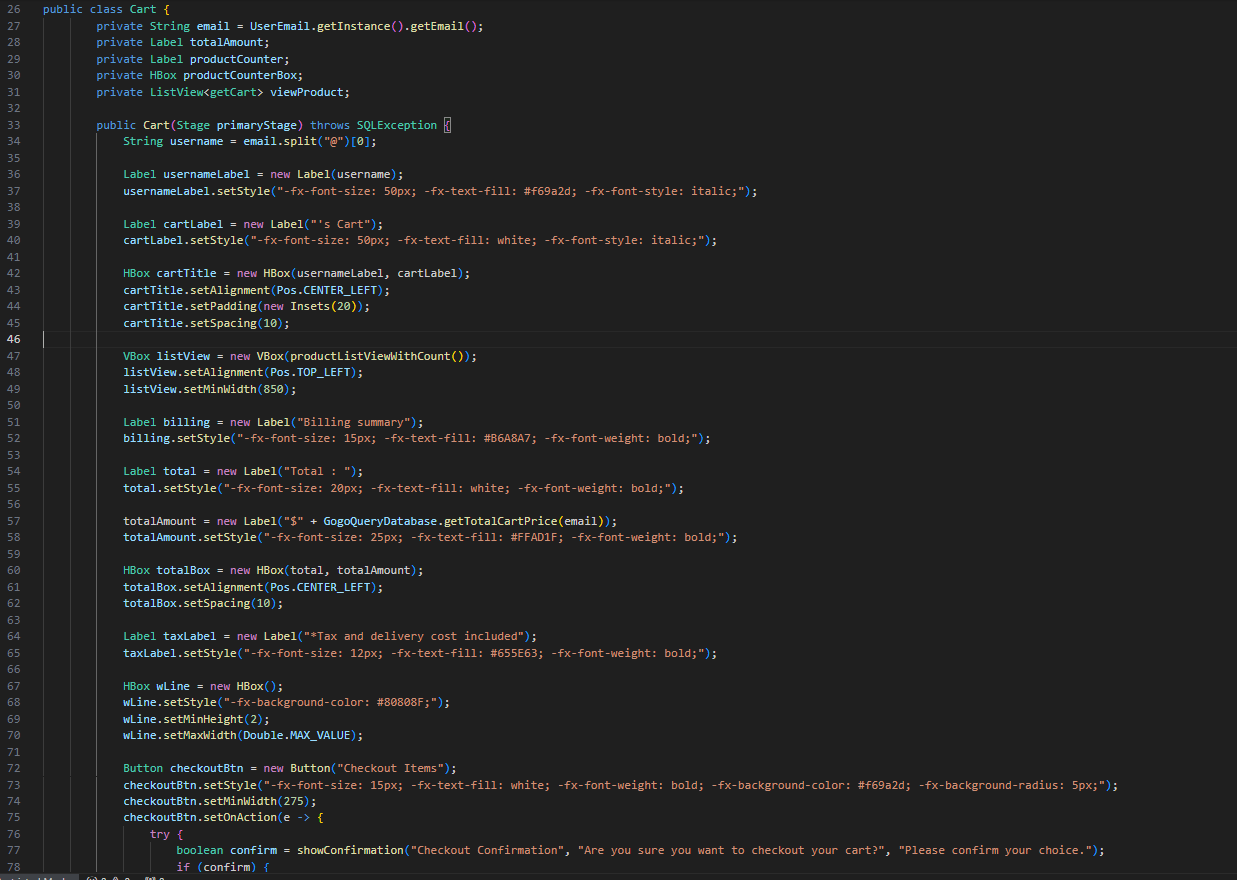
Atribut:

* private int id = Id unit dari transaksi
* Private int customerId = Id pelanggan terkait transaksi
* Private String customerEmail = Email pelanggan yang melakukan transaksi
* Private String date = Tanggal transaksi
* Private double amount = Jumlah nominal transaksi
* Private String status = Status transaksi

Konstruktor = Untuk menginisialisasi objek transaction dengan nilai-nilai yang diberikan

Metode getter = Mengembalikan atribut yang ada seperti return id, return customerId, customerEmail, return date, return amount, return status

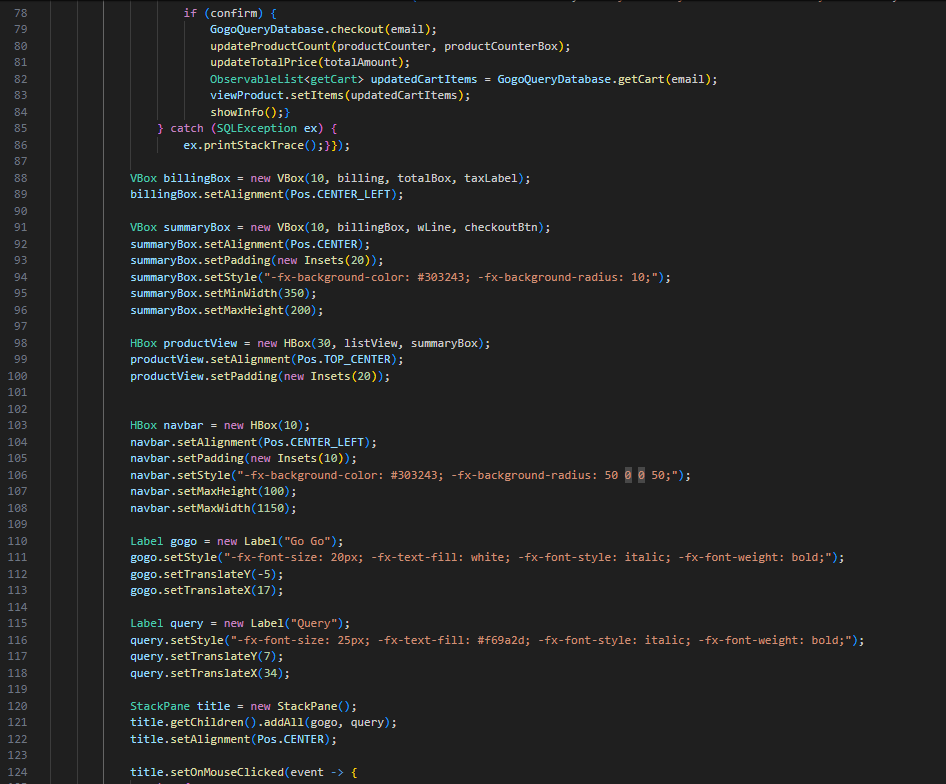
**Class: Cart**



public Cart(Stage primaryStage) = Inisialisasi komponen UI, judul, navbar, dan konten halaman

Atribut:

* String email = Menyimpan alamat email pengguna yang sedang aktif.
* Label totalAmount = Label yang digunakan untuk menampilkan total harga dari semua item yang ada di keranjang belanja. Nilai total ini diperoleh dengan memanggil metode getTotalCartPrice(email) dari kelas GogoQueryDatabase.
* Label productCounter = Label yang nantinya berfungsi untuk menampilkan jumlah total produk dalam keranjang.
* HBox productCounterBox = Komponen horizontal yang berisi elemen-elemen terkait jumlah produk dalam keranjang.
* ListView<getCart> viewProduct = Menampilkan daftar produk dalam keranjang. Data diperoleh melalui metode productListViewWithCount().



Proses Checkout:

* Melakukan konfirmasi checkout dengan memanggil fungsi GogoQueryDatabase.checkout.
* Memperbarui jumlah produk dan total harga keranjang.
* Mengambil data keranjang terbaru dari database menggunakan getCart.
* Menampilkan data keranjang yang diperbarui ke dalam viewProduct.
* Menangani kesalahan SQL menggunakan blok try-catch.

Layout:

* VBox digunakan untuk menampilkan billingBox, garis pemisah wline, dan tombol checkout checkoutBtn.
* Properti visual seperti warna latar belakang (#302243), padding, dan radius sudut diatur menggunakan CSS inline.
* Alignment disetel ke tengah menggunakan Pos.CENTER.

Navbar:

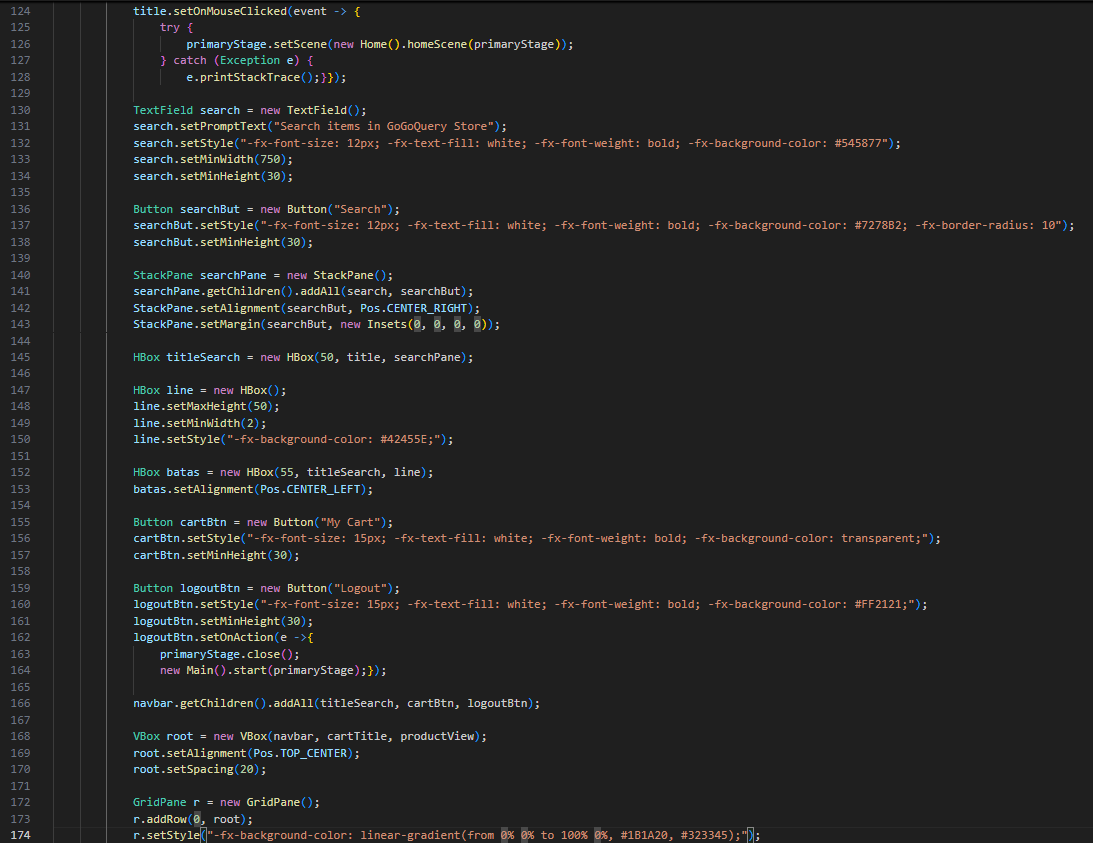
* HBox digunakan sebagai navbar untuk aplikasi.
* Properti visual seperti radius sudut asimetri (50 0 50 50) dan warna latar belakang diatur.
* Lebar dan tinggi maksimum navbar ditetapkan.

Judul Halaman:

* Label Go Go dan Query berfungsi sebagai judul halaman.
* Properti teks (ukuran font, warna, gaya) diatur menggunakan CSS inline.
* Posisi teks diubah menggunakan setTranslateX dan setTranslateY.

StackPane buat judul:

* StackPane digunakan untuk menumpuk label gogo dan query.
* Elemen-elemen anak ditambahkan ke dalam title dan diatur ke posisi tengah.



Event klik pada judul(title.setOnMouseClicked(event -> {):

* Ketika judul title diklik, aplikasi akan berpindah ke scene utama.
* Menggunakan try-catch untuk menangani kesalahan saat pemindahan scene.

Input pencarian produk(TextField search = new TextField();):

* TextField digunakan sebagai input untuk pencarian produk.
* search.setPromptText digunakan untuk menampilkan tulisan "Search items in GoGoQuery Store"
* search.setMinWidth(750); digunakan untuk mengatur minimal lebar
* search.setMinHeight(30); digunakan untuk mengatur minimal tinggi

Tombol Search(Button searchBut = new Button("Search");):

* Tombol Search untuk memulai pencarian produk.
* CSS inline digunakan untuk mengatur gaya tombol, termasuk warna latar belakang dan radius border seperti searchBut.setStyle, searchBut.setMinWidth(80); dan searchBut.setMinHeight(30);

StackPane untuk pencarian(StackPane searchPane = new StackPane();):

* StackPane digunakan untuk menempatkan tombol Search di samping TextField.
* Posisi tombol disesuaikan ke kanan menggunakan Pos.CENTER\_RIGHT.

HBox untuk Judul dan Pencarian(HBox titleSearch = new HBox(50, title, searchPane);) = HBox menyusun elemen title dan searchPane dengan jarak 50px di antara keduanya.

Garis Pemisah(HBox line = new HBox();):

* Garis horizontal digunakan sebagai pemisah visual antar elemen.
* Mengatur tinggi dan warna garis menggunakan CSS inline seperti line.setStyle, line.setMaxHeight(50); dan line.setMinHeight(2);

Tombol “My Cart” dan “Logout”(Button cartBtn = new Button("My Cart"); dan Button logoutBtn = new Button("Logout");):

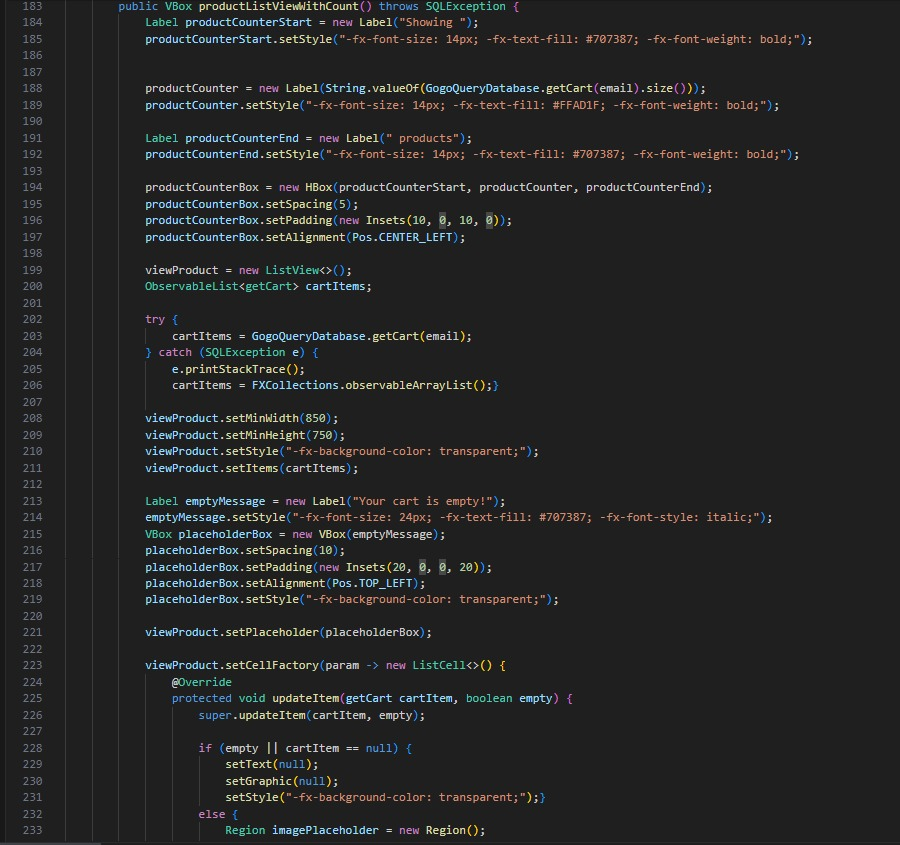
* Tombol My Cart untuk menampilkan keranjang belanja.
* Tombol Logout menutup scene saat ini dan memulai ulang aplikasi menggunakan Main.

Penyusunan Navbar dan Layout Utama(HBox navbar = new HBox(55, titleSearch, cartBtn, logoutBtn); dan VBoxroot = new VBox(navbar, cartView, productView);):

* HBox digunakan untuk menyusun titleSearch, My Cart, dan Logout secara horizontal.
* VBox menyusun elemen navbar di atas elemen lainnya dengan jarak 20px.

GridPane buat latar belakang(GridPane r = new GridPane();):

* GridPane digunakan sebagai kontainer utama dengan latar belakang gradasi warna.
* root ditambahkan sebagai elemen baris pertama.



Label buat header counter(Label productCounterStart = new Label("Showing ");):

* Membuat label dengan teks "Showing ".
* Mengatur tampilan teks menggunakan style (ukuran font, warna teks, dan ketebalan font).

Menampilkan Jumlah Item dalam Keranjang(productCounter=new Label(String.valueOf(GogoQueryDatabase.getCart(email).size()));):

* Mengambil jumlah item dari database melalui getCart(email).size() dan menampilkan hasil dalam label.
* Mengatur *style* untuk tampilan jumlah item (warna teks oranye).

Deklarasi Label Akhir Counter(Label productCounterEnd = new Label(" products");) = Menambahkan label teks "products"

Menyusun Label Counter ke dalam HBox(productCounterBox = new HBox(productCounterStart, productCounter, productCounterEnd);):

* Menggabungkan tiga label (Start, Counter, dan End) ke dalam sebuah HBox.
* Menambahkan spasi horizontal antar elemen sebesar 5.
* Menambahkan padding (atas, kanan, bawah, kiri).
* Mengatur posisi teks ke kiri-tengah (CENTER\_LEFT).

Inisialisasi ListView untuk Produk(viewProduct = new ListView<>();):

* Membuat ListView yang akan menampilkan daftar produk.
* Mendeklarasikan ObservableList untuk menyimpan data produk.

Mengambil Data Keranjang dari Database(try {cartItems = GogoQueryDatabase.getCart(email);):

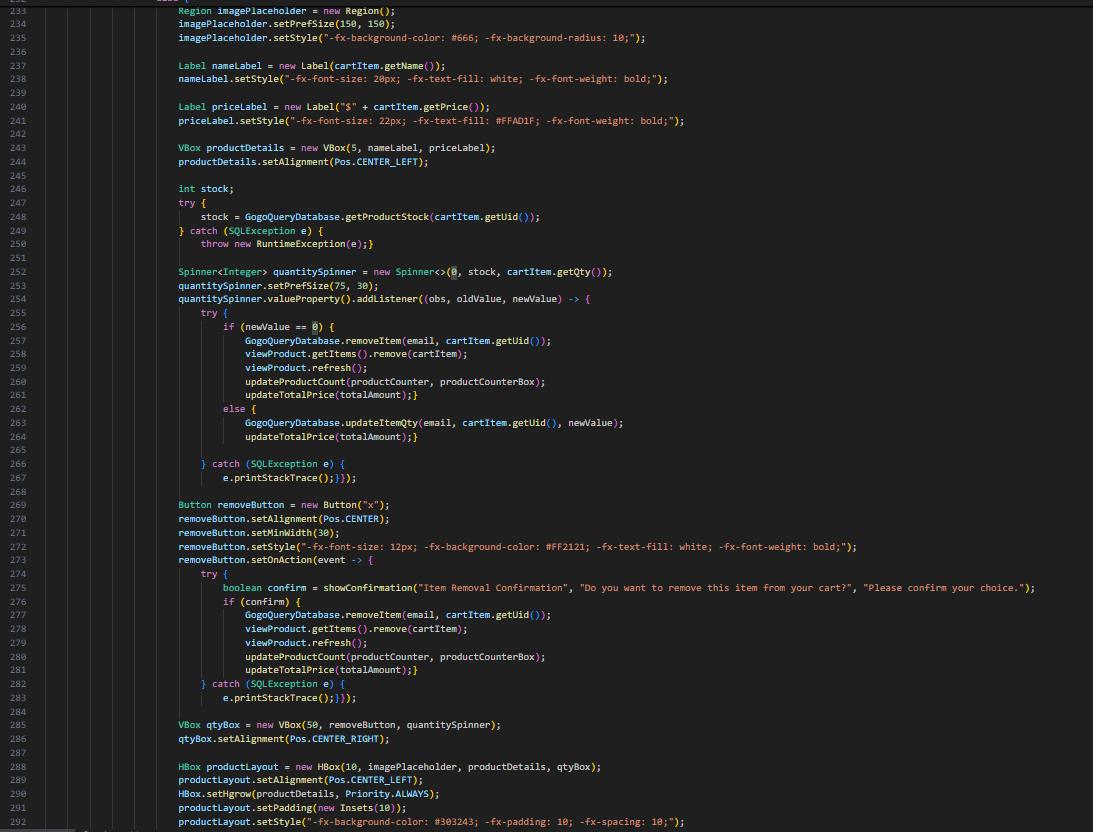
* Memulai blok try-catch untuk menangani error SQL.
* Mengambil data keranjang dengan method getCart(email).
* Jika terjadi kesalahan SQL, cetak stack trace untuk debugging.
* Jika gagal mengambil data, inisialisasi cartItems sebagai list kosong.

Menambahkan Placeholder Jika Keranjang Kosong(Label emptyMessage = new Label("Your cart is empty!");):

* Membuat pesan teks "Your cart is empty!".
* Menambahkan gaya teks (ukuran font, warna teks, dan italic).
* Membuat VBox untuk memuat pesan placeholder.
* Menambahkan spasi vertikal sebesar 10 antar elemen dalam VBox.
* Menambahkan padding untuk posisi pesan.
* Menyusun posisi placeholder di bagian atas-kiri.
* Membuat background placeholder transparan.

Custom ListCell untuk ListView(viewProduct.setCellFactory(param -> new ListCell<>() ):

* Mengatur setCellFactory untuk kustomisasi tampilan setiap item.
* Override method updateItem untuk memproses tiap item.
* Jika item kosong, hilangkan teks dan grafik, serta set background transparan.
* Menyiapkan placeholder untuk elemen Region (gambar atau elemen UI lain).



Placeholder Gambar(Region imagePlaceholder = new Region();):

* Membuat placeholder menggunakan Region untuk gambar produk.
* Mengatur ukuran dan gaya background gambar placeholder (warna abu-abu dan sudut membulat).

Menyusun Label Nama dan Harga(VBox productDetails = new VBox(5, nameLabel, priceLabel);):

* Menggabungkan nameLabel dan priceLabel ke dalam VBox dengan spasi vertikal 5.
* Menyusun posisi teks ke kiri-tengah (CENTER\_LEFT).

Mengambil Stok Produk dari Database(try {stock = GogoQueryDatabase.getProductStock(cartItem.getUid());):

* Mendeklarasikan variabel stock.
* Mengambil stok produk dari database menggunakan getProductStock.
* Menangani error SQL, jika terjadi maka dilempar sebagai RuntimeException.

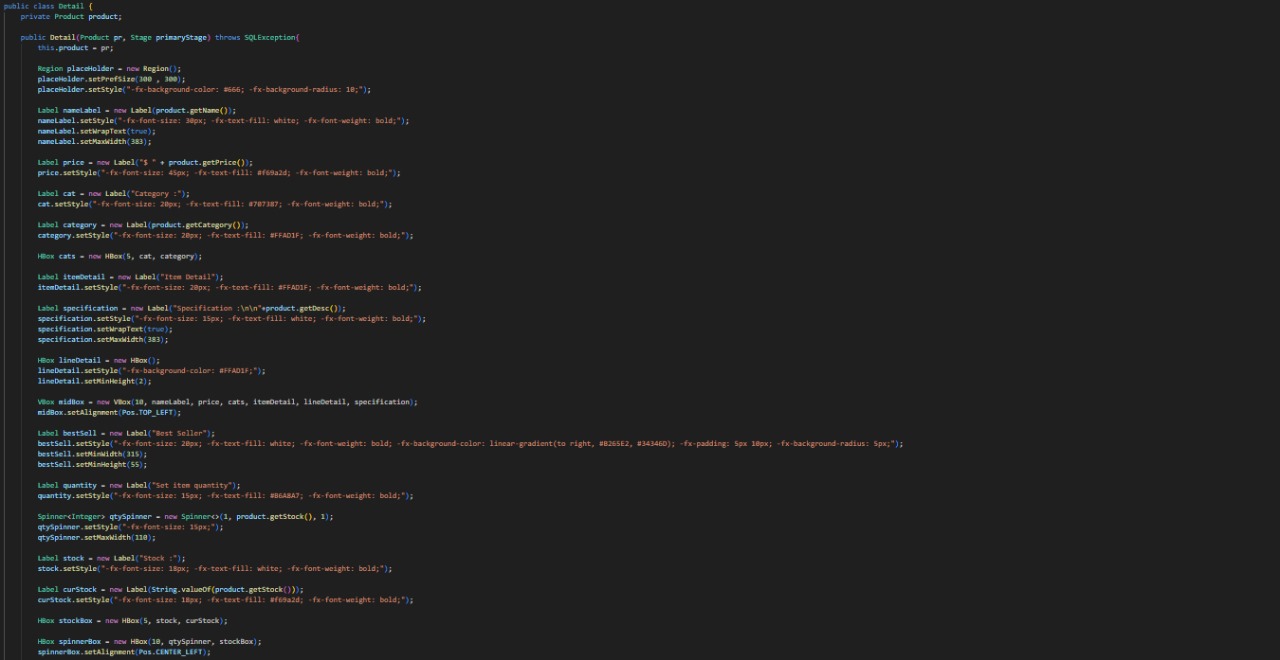
Spinner untuk Kuantitas Produk(Spinner<Integer> quantitySpinner = new Spinner<>(0, stock, cartItem.getQty());):

* Membuat Spinner dengan nilai minimum 0, maksimum stock, dan nilai awal cartItem.getQty().
* Mengatur ukuran spinner.
* Menambahkan event listener yang dipanggil ketika nilai spinner berubah:
* Jika nilai newValue == 0, item dihapus dari database dan UI.
* Jika nilai diperbarui, maka data di-update di database.

Button Hapus Item(Button removeButton = new Button("X");):

* Membuat tombol "X" untuk menghapus item.
* Menyusun posisi tombol ke tengah.
* Menentukan lebar minimum tombol.
* Mengatur gaya tombol (warna merah, teks putih, dan tebal).
* Menambahkan aksi event ketika tombol diklik:
* Menampilkan konfirmasi dialog.
* Jika konfirmasi "Yes", item dihapus dari database dan UI diperbarui.

**Class: Detail**

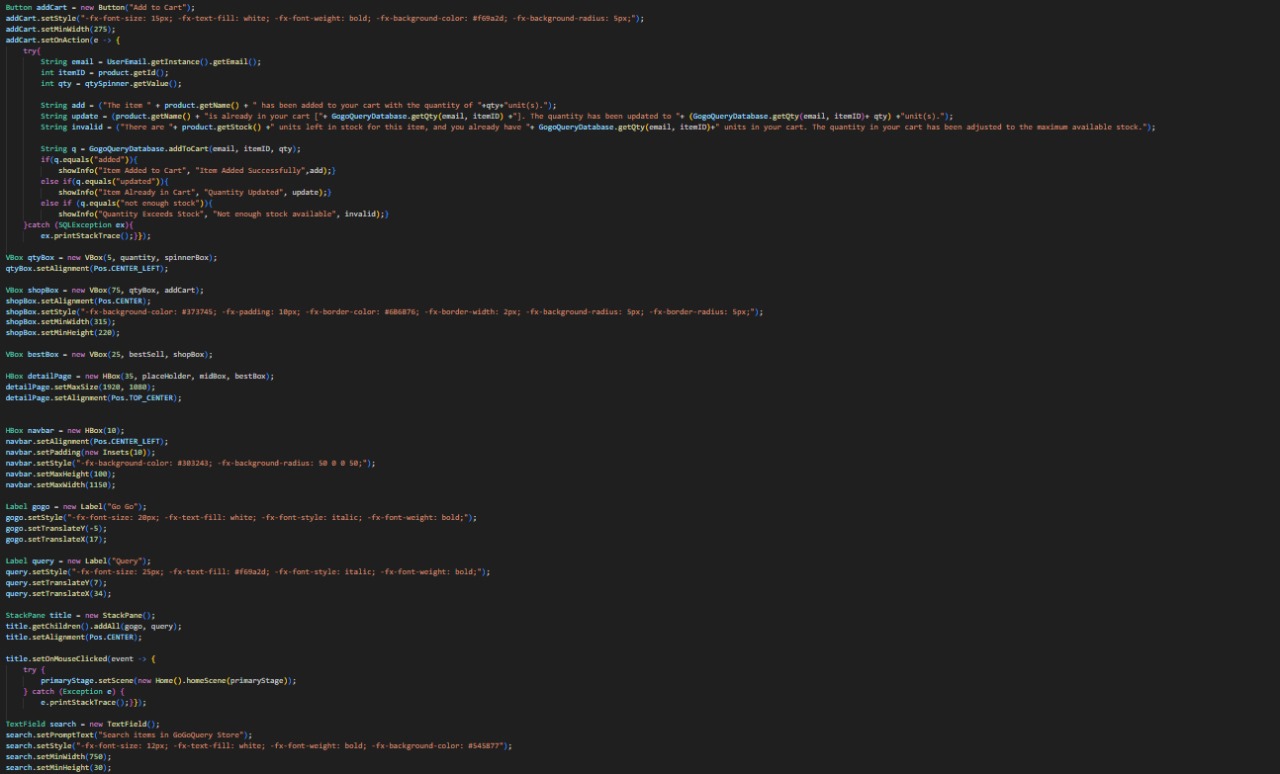


public String getProductName():  
Metode ini mengembalikan nilai dari atribut product berupa nama produk. Atribut product ini berisi informasi terkait produk yang akan ditampilkan pada aplikasi, dan metode ini digunakan untuk mengambil nama produk tersebut. Misalnya, nama produk akan digunakan untuk ditampilkan pada antarmuka pengguna agar pengguna dapat melihat produk yang dimaksud.

public double getProductPrice():  
Metode ini berfungsi untuk mengembalikan nilai harga produk yang ada dalam atribut product. Nilai harga ini biasanya digunakan untuk menunjukkan biaya atau harga jual produk pada aplikasi. Metode ini memungkinkan aplikasi untuk mengambil informasi harga dan menampilkannya secara dinamis sesuai dengan produk yang sedang dilihat atau dipilih oleh pengguna.

public String getProductCategory():  
Metode ini digunakan untuk mengambil informasi kategori produk dari atribut product . Produk sering kali dikategorikan berdasarkan jenis atau tipe tertentu, seperti elektronik, pakaian, atau barang lainnya. Dengan menggunakan metode ini, aplikasi dapat mengakses kategori produk untuk ditampilkan pada antarmuka pengguna, yang membantu pengguna untuk mengidentifikasi atau memfilter produk berdasarkan kategori yang relevan.

public int getProductStock():  
Metode ini digunakan untuk mendapatkan jumlah stok yang tersedia untuk produk tertentu. Atribut product menyimpan informasi mengenai stok produk yang tersisa, dan metode ini memberikan cara bagi aplikasi untuk mengakses nilai tersebut. Informasi stok ini biasanya penting dalam aplikasi untuk menampilkan ketersediaan barang dan membantu pengguna mengetahui apakah produk masih tersedia atau sudah habis.



String email = UserEmail.getInstance().getEmail();

Kode ini digunakan untuk mengambil email pengguna yang sedang login atau aktif dari instance kelas UserEmail. Dengan memanggil metode getEmail() dari instance UserEmail, sistem dapat mengetahui identitas pengguna yang sedang menggunakan aplikasi. Email ini nantinya akan digunakan untuk mengaitkan data keranjang belanja dengan akun pengguna tersebut.

int itemID = product.getId();

Kode ini berfungsi untuk mengambil ID produk yang dipilih. Metode getId() dipanggil dari objek product, yang berisi informasi produk yang sedang dipilih oleh pengguna. ID produk ini digunakan sebagai identifikasi unik dalam basis data untuk memastikan produk yang tepat ditambahkan ke keranjang belanja.

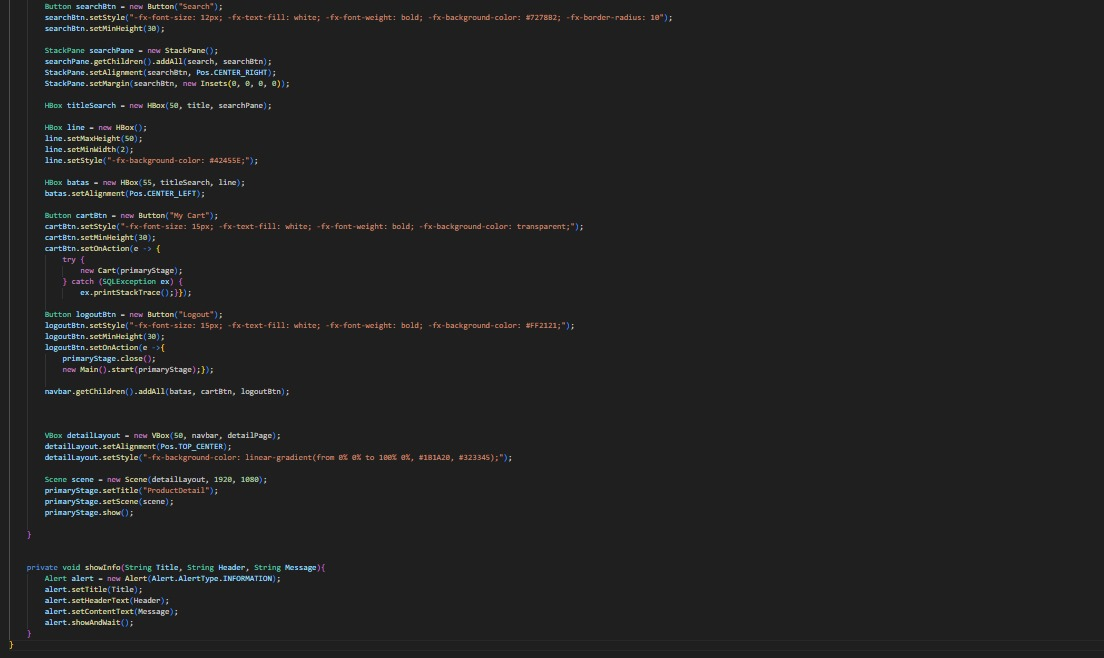
int qty = qtySpinner.getValue();

Kode ini digunakan untuk mengambil jumlah unit produk yang dipilih dari komponen spinner. Komponen spinner memungkinkan pengguna untuk memasukkan atau memilih jumlah produk yang ingin mereka beli. Nilai yang diambil dari metode getValue() akan digunakan sebagai kuantitas yang dimasukkan ke dalam keranjang.

GodQueryDatabase.addToCart(email, itemID, qty);

Kode ini berfungsi untuk menambahkan produk ke keranjang belanja. Metode addToCart() dipanggil dari kelas GodQueryDatabase dan memerlukan tiga parameter:

1. email: Email pengguna yang sedang aktif.
2. itemID: ID produk yang akan ditambahkan ke keranjang.
3. qty: Jumlah unit produk yang dipilih pengguna.



Button searchBtn = new Button("Search");

Kode ini membuat tombol bernama searchBtn dengan label "Search". Tombol ini digunakan untuk memulai aksi pencarian berdasarkan input pengguna.

StackPane searchPane = new StackPane();

Kode ini membuat objek StackPane, sebuah layout di JavaFX yang memungkinkan elemen-elemen ditumpuk satu sama lain. Tombol searchBtn kemudian ditambahkan ke dalam StackPane.

Button cartBtn = new Button("My Cart");

Kode ini membuat tombol bernama cartBtn dengan label "My Cart". Tombol ini berfungsi untuk menampilkan halaman keranjang belanja pengguna.

cartBtn.setOnAction(e -> { ... });

Kode ini menentukan aksi yang akan dijalankan ketika tombol "My Cart" ditekan:

1. Membuat objek Stage baru yang menampilkan halaman keranjang.
2. Jika terjadi kesalahan (exception), maka akan dicetak melalui printStackTrace().

Button logoutBtn = new Button("Logout");

Kode ini membuat tombol logoutBtn dengan label "Logout". Tombol ini digunakan agar pengguna bisa keluar dari aplikasi atau sesi yang sedang aktif.

logoutBtn.setOnAction(e -> { primaryStage.close(); });

Ketika tombol "Logout" ditekan, aksi ini akan menutup aplikasi dengan memanggil primaryStage.close().

VBox DetailLayout= new VBox(50, navbar, DetailPage);

Kode ini membuat layout vertikal (VBox) dengan jarak antar elemen sebesar 50 piksel. Elemen yang ditambahkan ke dalam VBox adalah navbar (berisi tombol navigasi) dan DetailPage.

private void showInfo(String Title, String Header, String Message)

Kode ini mendefinisikan metode showInfo untuk menampilkan notifikasi informasi menggunakan komponen Alert di JavaFX.

1. alert.setTitle(Title): Menetapkan judul notifikasi.
2. alert.setHeaderText(Header): Menetapkan teks header.
3. alert.setContentText(Message): Menetapkan isi pesan yang akan ditampilkan.
4. alert.showAndWait(): Menampilkan dialog notifikasi dan menunggu aksi pengguna.

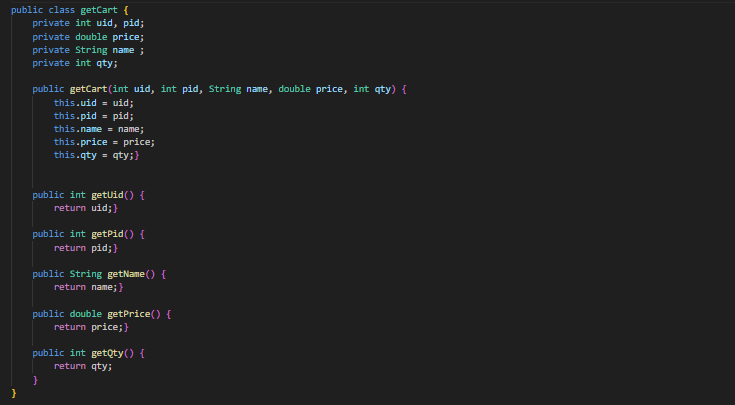
Tampilan Umum

Kode ini membangun antarmuka pengguna yang mencakup:

1. Tombol Search untuk aksi pencarian.
2. Tombol My Cart untuk menampilkan keranjang belanja.
3. Tombol Logout untuk keluar dari aplikasi.
4. Layout VBox untuk menyusun elemen-elemen vertikal seperti navigasi dan konten halaman.

Fitur notifikasi melalui metode showInfo memastikan pengguna mendapat umpan balik informasi saat melakukan aksi tertentu.

**Class: getCart**



Atribut:

uid (int): Mungkin merepresentasikan unique identifier atau ID unik dari pengguna.

pid (int): Kemungkinan merepresentasikan product ID atau ID unik dari produk.

name (String): Menyimpan nama produk.

price (double): Menyimpan harga produk.

qty (int): Menyimpan jumlah produk yang dibeli.

Konstruktor:

public getCart(int uid, int pid, String name, double price, int qty): Ini adalah konstruktor kelas getCart. Konstruktor ini digunakan untuk menginisialisasi objek getCart dengan nilai-nilai yang diberikan sebagai parameter. Nilai-nilai parameter ini kemudian ditugaskan ke variabel instance (atribut) objek.

Metode Getter:

public int getUid(): Metode ini mengembalikan nilai dari atribut uid.

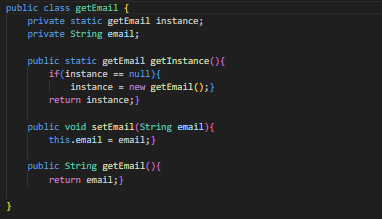
public int getPid(): Metode ini mengembalikan nilai dari atribut pid.

public String getName(): Metode ini mengembalikan nilai dari atribut name.

public double getPrice(): Metode ini mengembalikan nilai dari atribut price.

public int getQty(): Metode ini mengembalikan nilai dari atribut qty.

**Class:getEmail**



private static getEmail instance;: Variabel instance dari tipe getEmail sendiri. Ini adalah variabel statis, artinya hanya ada satu salinan untuk seluruh objek dari kelas getEmail. Variabel ini akan menyimpan satu-satunya objek getEmail yang akan dibuat.

private String email;: Variabel email untuk menyimpan alamat email.

Metode getInstance():

Metode ini bersifat static sehingga dapat dipanggil tanpa perlu membuat objek dari kelas getEmail.

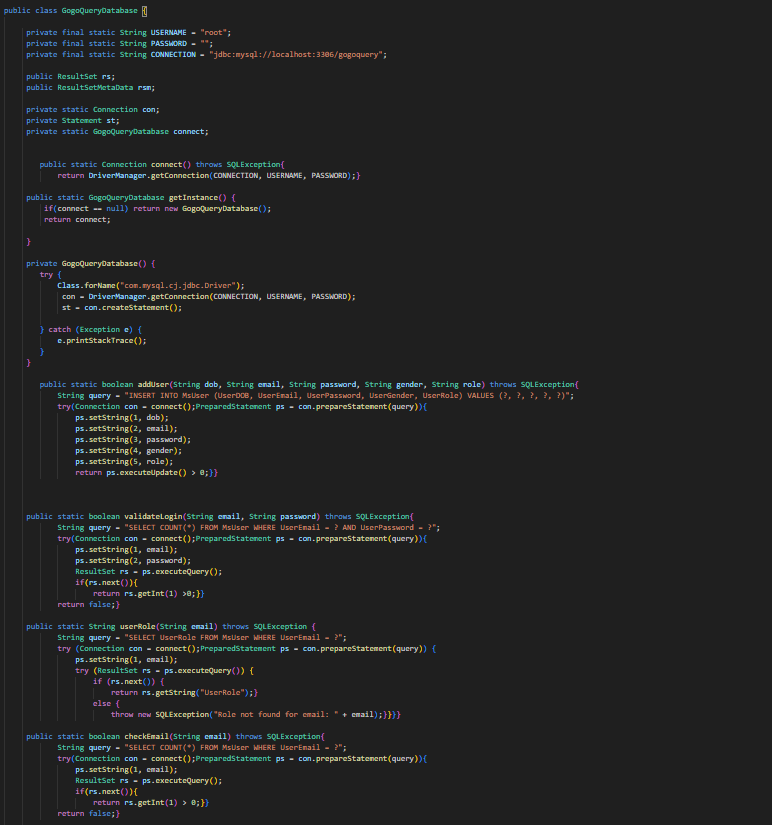
Periksa Instans: Jika instance masih null (belum ada objek yang dibuat), maka buat objek baru dari kelas getEmail dan simpan ke dalam instance.

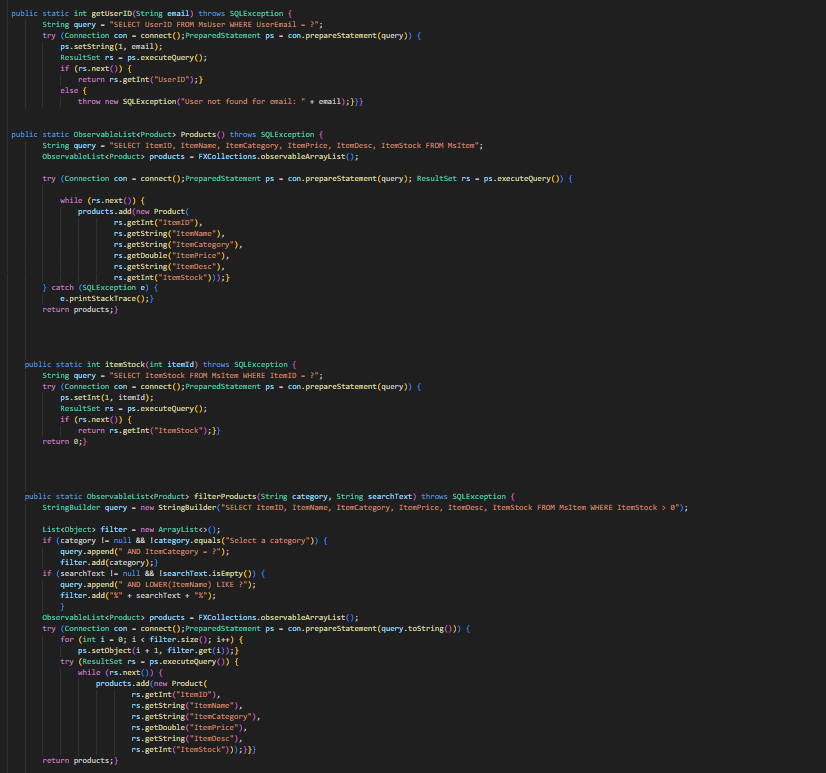
Kembalikan Instans: Kembalikan instance yang sudah ada atau yang baru dibuat.

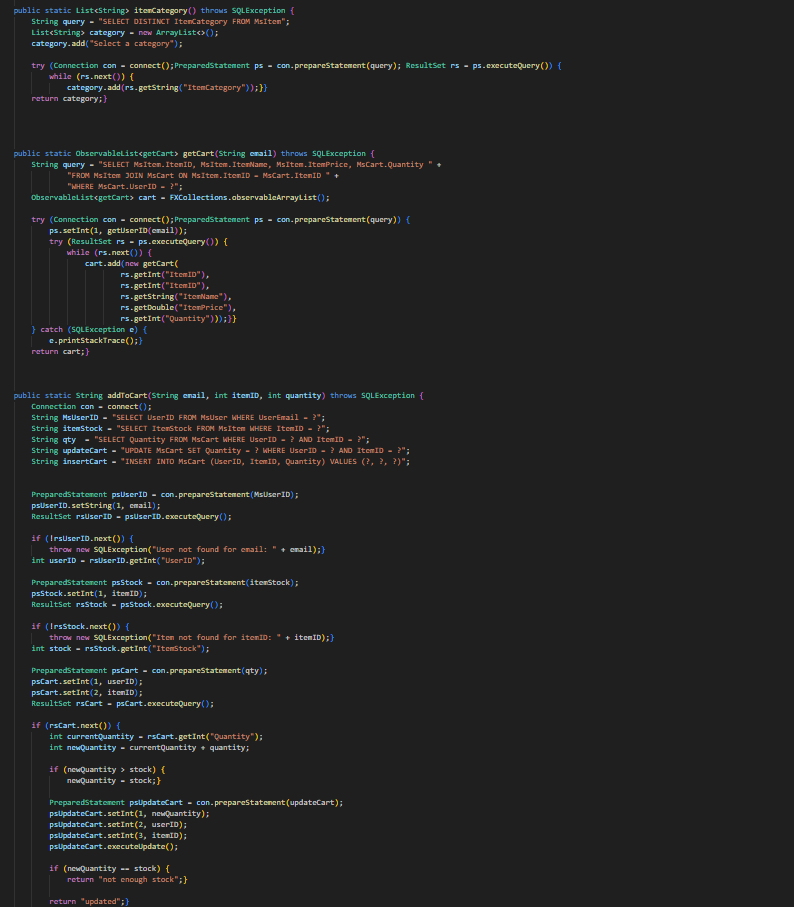
Metode setEmail() dan getEmail():

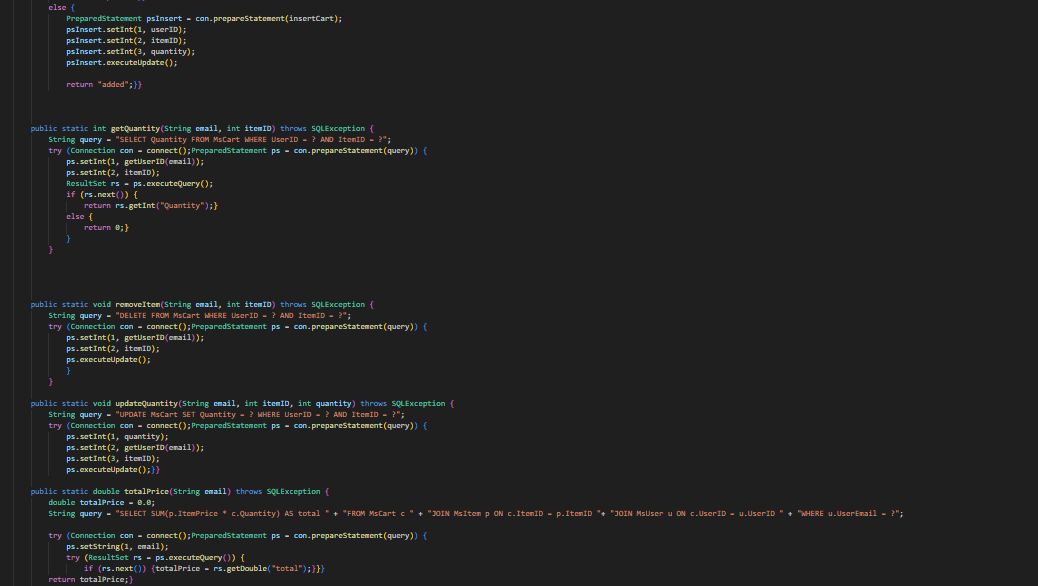
Metode setEmail() digunakan untuk mengatur nilai email.

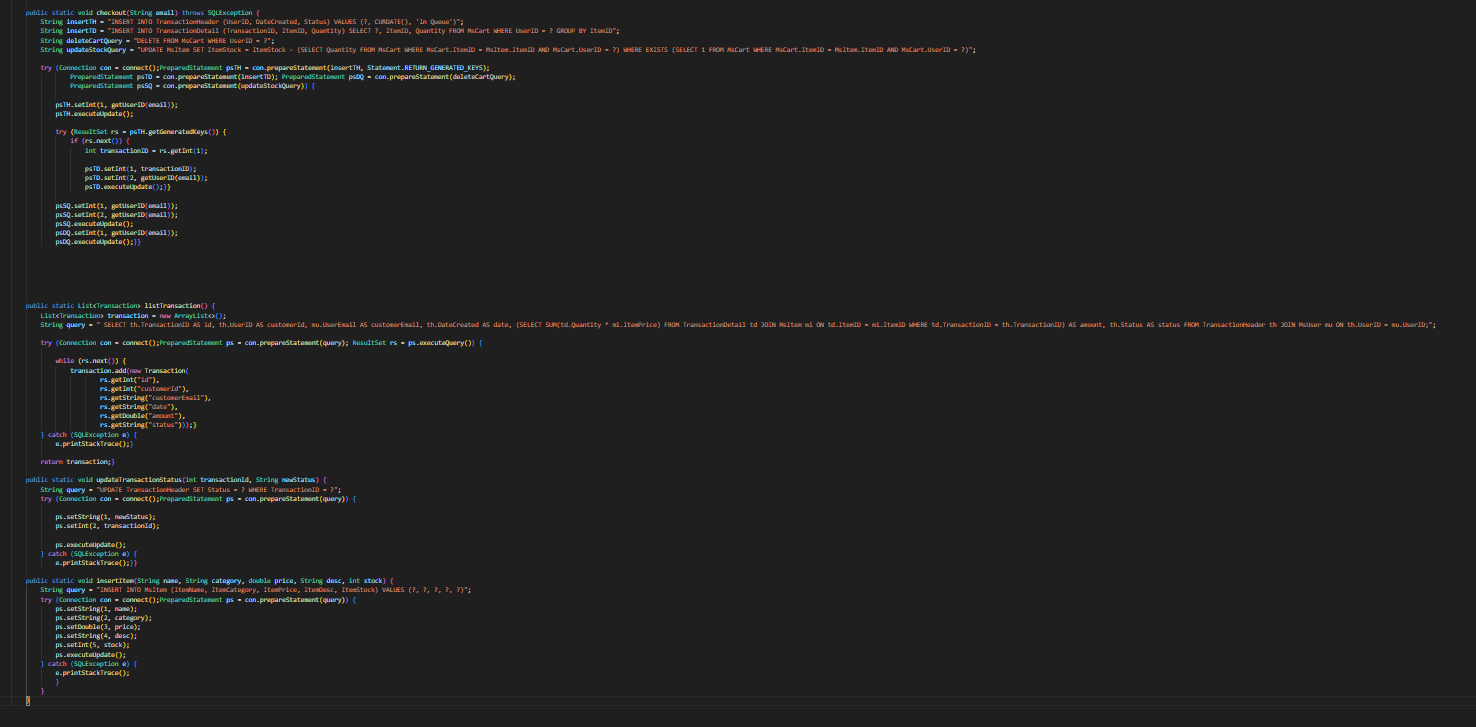
Metode getEmail() digunakan untuk mendapatkan nilai email.

**Class: GogoQueryDatabase**  
.









1. Koneksi Database

Konstanta Koneksi:

USERNAME, PASSWORD, dan CONNECTION digunakan untuk menghubungkan ke database MySQL dengan JDBC.

DriverManager.getConnection() membuat koneksi ke database.

Singleton Pattern:

Metode getInstance() memastikan hanya ada satu instance kelas GogoQueryDatabase.

2. Metode CRUD (Create, Read, Update, Delete)

Manajemen Pengguna

addUser:

Menambahkan pengguna baru ke tabel MsUser dengan data seperti email, password, tanggal lahir, gender, dan role.

validateLogin:

Memeriksa apakah kombinasi email dan password valid di tabel MsUser.

checkEmail:

Memeriksa apakah email sudah ada di database.

userRole:

Mengambil peran (UserRole) pengguna berdasarkan email.

getUserID:

Mengambil UserID berdasarkan email pengguna.

Manajemen Produk

Products:

Mengambil semua produk dari tabel MsItem.

filterProducts:

Menyaring produk berdasarkan kategori dan teks pencarian.

itemCategory:

Mengambil semua kategori produk yang unik dari tabel MsItem.

itemStock:

Mengambil jumlah stok produk berdasarkan ItemID.

Manajemen Keranjang (Cart)

getCart:

Mengambil daftar produk dalam keranjang pengguna tertentu.

addToCart:

Menambahkan atau memperbarui kuantitas produk dalam keranjang.

getQuantity:

Mengambil kuantitas produk tertentu dalam keranjang.

removeItem:

Menghapus item tertentu dari keranjang.

updateQuantity:

Memperbarui jumlah kuantitas produk dalam keranjang.

totalPrice:

Menghitung total harga semua item dalam keranjang.

Checkout dan Transaksi

checkout:

Membuat transaksi baru dengan menyalin data dari keranjang ke tabel transaksi dan mengurangi stok produk.

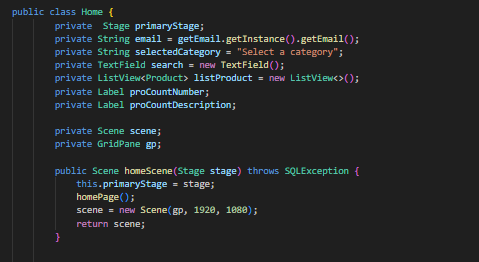
listTransaction:

Mengambil daftar transaksi pengguna termasuk total harga dan status transaksi.

updateTransactionStatus:

Memperbarui status transaksi tertentu.

**Class: Home**



primaryStage: Variabel tipe Stage ini kemungkinan digunakan untuk merepresentasikan jendela utama aplikasi.

email: Variabel tipe String ini menyimpan alamat email pengguna. Nilai awal diambil dari hasil pemanggilan metode getEmail.getInstance().getEmail(), yang mengindikasikan adanya kelas getEmail dengan metode getInstance() dan getEmail() untuk mendapatkan informasi email.

selectedCategory: Variabel tipe String ini menyimpan kategori yang dipilih pengguna. Nilai awalnya adalah "Select a category".

search: Variabel tipe TextField ini kemungkinan digunakan sebagai kotak pencarian.

listProduct: Variabel tipe ListView<Product> ini digunakan untuk menampilkan daftar produk.

proCountNumber dan proCountDescription: Variabel tipe Label ini mungkin digunakan untuk menampilkan jumlah produk dan deskripsi terkait.

scene: Variabel tipe Scene ini merepresentasikan tampilan visual dari kelas Home.

gp: Variabel tipe GridPane ini kemungkinan digunakan sebagai layout untuk mengatur tata letak elemen-elemen UI dalam scene.

Metode ini menerima parameter stage tipe Stage dan mengembalikan objek Scene.

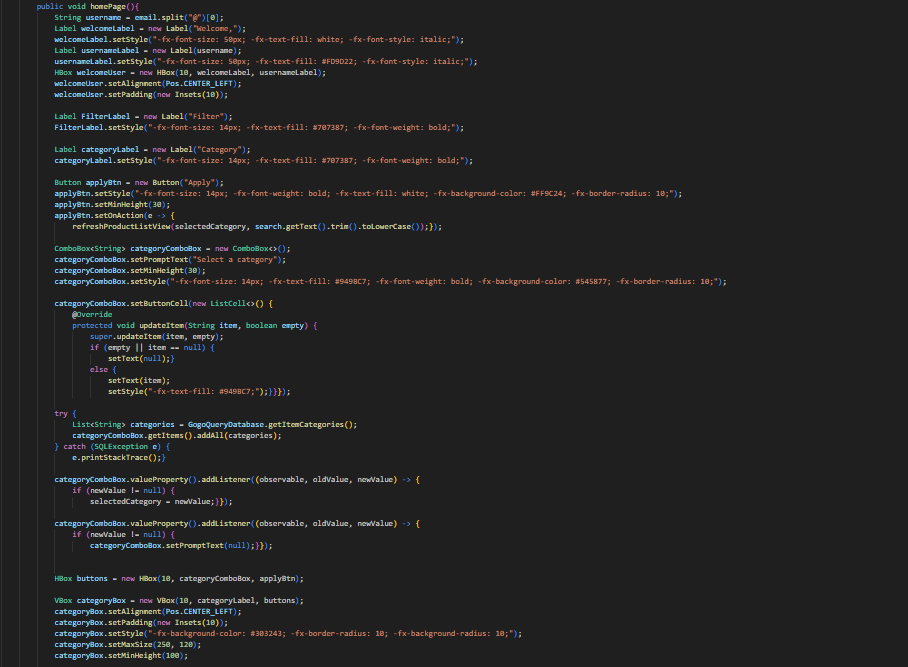
Di dalam metode ini:

this.primaryStage = stage;: Menugaskan nilai stage ke variabel primaryStage.

homePage();: Memanggil metode homePage() yang belum ditampilkan dalam kode ini. Metode ini kemungkinan berisi logika untuk menginisialisasi tampilan halaman utama.

scene = new Scene(gp, 1920, 1080);: Membuat objek Scene baru dengan ukuran 1920x1080 piksel dan menugaskannya ke variabel scene.

return scene;: Mengembalikan objek Scene yang telah dibuat.



Deklarasi dan Inisialisasi Label:

welcomeLabel, usernameLabel, filterLabel, categoryLabel: Membuat label-label dengan teks dan gaya tertentu. Gaya ini mengatur font, warna, ukuran, dan berat font.

Tombol applyBtn:

Membuat tombol dengan teks "Apply".

Menambahkan event handler setOnAction yang akan menjalankan fungsi refreshProductListView ketika tombol ditekan. Fungsi ini kemungkinan akan me-refresh tampilan daftar produk berdasarkan kategori dan kata kunci pencarian yang dipilih.

ComboBox categoryComboBox:

Membuat kotak kombo untuk memilih kategori.

Mengatur tampilan dan gaya kotak kombo.

Menambahkan listener untuk mendeteksi perubahan nilai yang dipilih.

Mengisi opsi dalam kotak kombo dengan data kategori yang diambil dari database menggunakan GogoQueryDatabase.getItemCategories().

Layout:

HBox buttons: Membuat kotak horizontal untuk menampung categoryComboBox dan applyBtn.

VBox categoryBox: Membuat kotak vertikal untuk menampung categoryLabel, buttons, dan mengatur gaya serta ukurannya.



Inisialisasi ListView:

ListProduct listProduct = new ListView<>();

Membuat objek ListView baru yang akan digunakan untuk menampilkan daftar produk.

Pengaturan Gaya:

Mengatur berbagai properti gaya seperti warna latar belakang, padding, border, dan ukuran untuk ListView.

Cell Factory:

listProduct.setCellFactory(param -> new ProductCell());

Menentukan tampilan setiap item dalam ListView dengan menggunakan kelas ProductCell.

Kelas ProductCell:

Kelas ini merepresentasikan tampilan dari setiap item (produk) dalam ListView.

Elemen UI: Membuat label untuk menampilkan nama produk, harga, stok, dan mungkin gambar produk.

Layout: Mengatur tata letak elemen-elemen UI menggunakan layout seperti VBox dan HBox.

Styling: Mengatur gaya untuk setiap elemen UI.

Event Handler: Menambahkan event handler untuk merespons ketika pengguna mengklik item.

Mengisi Data:

List<Product> availableProducts = SogoQueryDatabase.getProducts();

Mengambil data produk dari database menggunakan SogoQueryDatabase.getProducts().

listProduct.getItems().addAll(availableProducts);

Menambahkan data produk ke dalam ListView.



Inisialisasi ListView:

ListProduct listProduct = new ListView<>();

Membuat objek ListView baru untuk menampilkan daftar produk.

Pengaturan Gaya:

Banyak baris kode yang digunakan untuk mengatur gaya tampilan berbagai komponen seperti label, tombol, dan kotak pencarian menggunakan CSS. Ini termasuk pengaturan font, warna, ukuran, padding, dan border.

Tata Letak:

Menggunakan HBox dan VBox untuk mengatur tata letak komponen-komponen UI secara horizontal dan vertikal.

GridPane digunakan untuk mengatur tata letak keseluruhan halaman.

Kotak Pencarian:

Membuat TextField untuk input pencarian dan Button untuk melakukan pencarian.

Menambahkan event listener pada tombol pencarian untuk memicu aksi pencarian.

Keranjang Belanja:

Membuat tombol "My Cart" untuk mengakses keranjang belanja.

Menambahkan event listener pada tombol untuk membuka halaman keranjang belanja.

Navigasi:

Membuat tombol "Logout" untuk keluar dari aplikasi.

Menambahkan event listener pada tombol untuk menutup jendela aplikasi.



refreshProductListView: Fungsi ini menerima dua parameter, yaitu category (kategori yang dipilih) dan searchText (kata kunci pencarian).

SogoQueryDatabase.filterProducts: Fungsi ini diasumsikan merupakan fungsi yang berinteraksi dengan database untuk mengambil data produk berdasarkan kategori dan kata kunci pencarian.

filteredProducts: Menyimpan hasil pencarian produk yang didapatkan dari database.

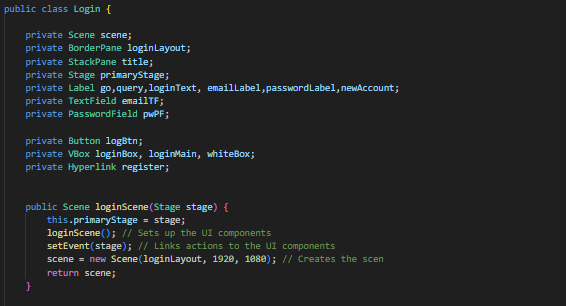
descriptionFlow: Objek TextFlow digunakan untuk membuat tampilan teks yang dinamis, di mana kita bisa mengatur gaya dan warna teks secara individual.

Kondisi-kondisi if-else: Digunakan untuk menentukan teks deskripsi yang akan ditampilkan berdasarkan kombinasi kategori dan kata kunci pencarian yang dimasukkan pengguna.

coloredText dan defaultText: Fungsi-fungsi ini digunakan untuk membuat objek Text dengan gaya yang berbeda (warna berbeda) untuk kata kunci pencarian dan kata kategori.

proCountNumber dan proCountDescription: Elemen UI yang digunakan untuk menampilkan jumlah produk yang ditemukan dan deskripsi hasil pencarian.

**Class: Login**



Deklarasi Variabel:   
scene: Ini adalah wadah untuk semua elemen tampilan login kita. Bayangkan seperti kanvas kosong yang akan kita isi dengan berbagai komponen.

loginLayout: Ini adalah tata letak utama yang mengatur posisi semua komponen dalam tampilan login. Kita bisa membayangkannya seperti sebuah bingkai foto yang dibagi menjadi beberapa bagian.

primaryStage: Ini adalah jendela utama aplikasi kita. Semua elemen tampilan akan ditampilkan di dalam jendela ini.

title: Ini kemungkinan digunakan untuk membuat judul atau header pada tampilan login.

go, query, loginText, dll: Ini adalah label-label yang menampilkan teks seperti "GoGo Query", "Login", "Email", dan sebagainya.

emailTF: Ini adalah kotak teks tempat pengguna memasukkan alamat email.

pwPF: Ini adalah kotak teks khusus untuk memasukkan password (inputannya akan tersembunyi).

logBtn: Ini adalah tombol "Login" yang akan memicu aksi login ketika ditekan.

loginBox, loginMain, whiteBox: Ini adalah wadah-wadah yang digunakan untuk mengelompokkan komponen-komponen lain secara vertikal.

register: Ini adalah link yang mengarah ke halaman pendaftaran.

Metode loginScene:

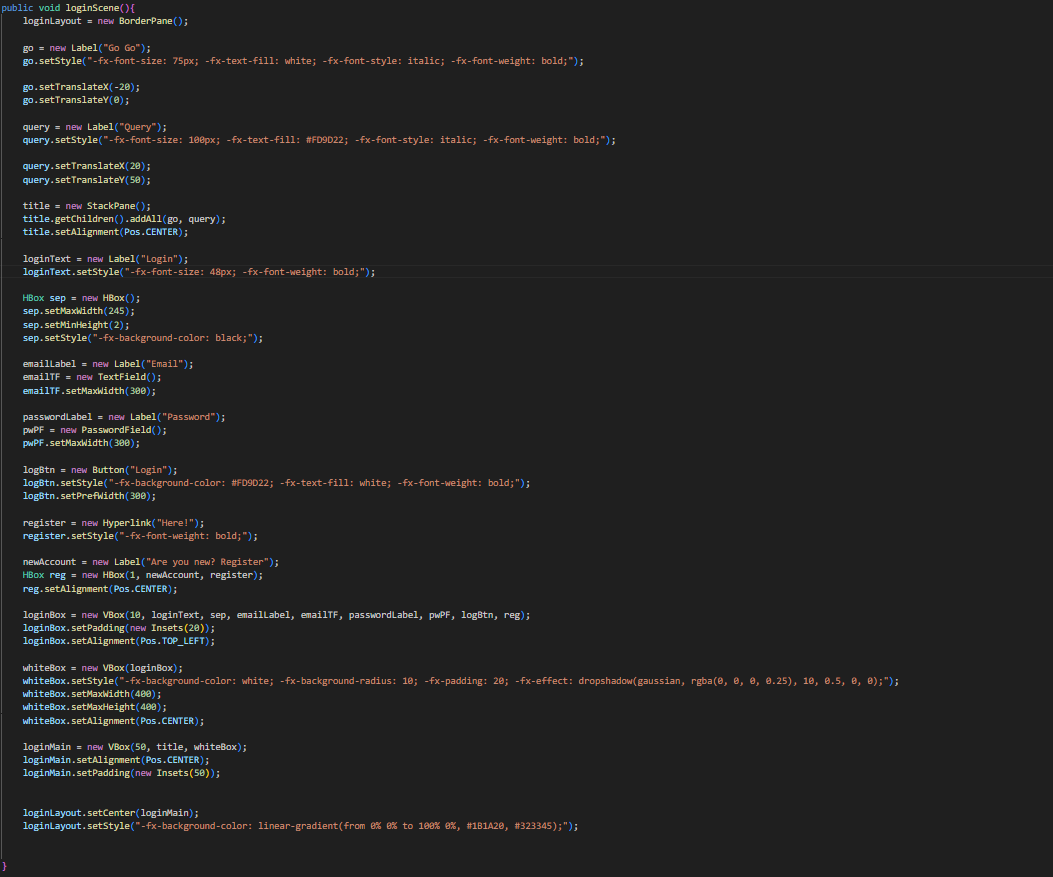
Inisialisasi primaryStage: Menyimpan referensi ke jendela utama dalam variabel primaryStage.

Memanggil loginScene(): Metode ini kemungkinan berisi logika untuk membuat dan mengatur tampilan login.

Memanggil setEvent(): Metode ini kemungkinan berisi logika untuk menghubungkan event (misalnya, ketika tombol "Login" ditekan) dengan aksi yang akan dilakukan.

Membuat Scene: Membuat objek Scene baru dan mengatur ukurannya.

Mengembalikan Scene: Mengembalikan objek Scene yang telah dibuat.



loginLayout = new BorderPane();: Membuat sebuah BorderPane baru yang akan menjadi wadah utama untuk semua komponen.

go = new Label("GoGo Query");: Membuat label dengan teks "GoGo Query" dan mengatur gaya font-nya.

query = new Label("Query");: Membuat label dengan teks "Query" dan mengatur gaya font-nya.

title = new StackPane();: Membuat StackPane untuk menampung label "GoGo Query" dan "Query".

loginText = new Label("Login");: Membuat label dengan teks "Login".

emailLabel = new Label("Email");: Membuat label dengan teks "Email".

emailTF = new TextField();: Membuat kotak teks untuk input email.

passwordLabel = new Label("Password");: Membuat label dengan teks "Password".

pwPF = new PasswordField();: Membuat kotak teks untuk input password (inputannya akan tersembunyi).

logBtn = new Button("Login");: Membuat tombol "Login" dan mengatur gaya tampilannya.

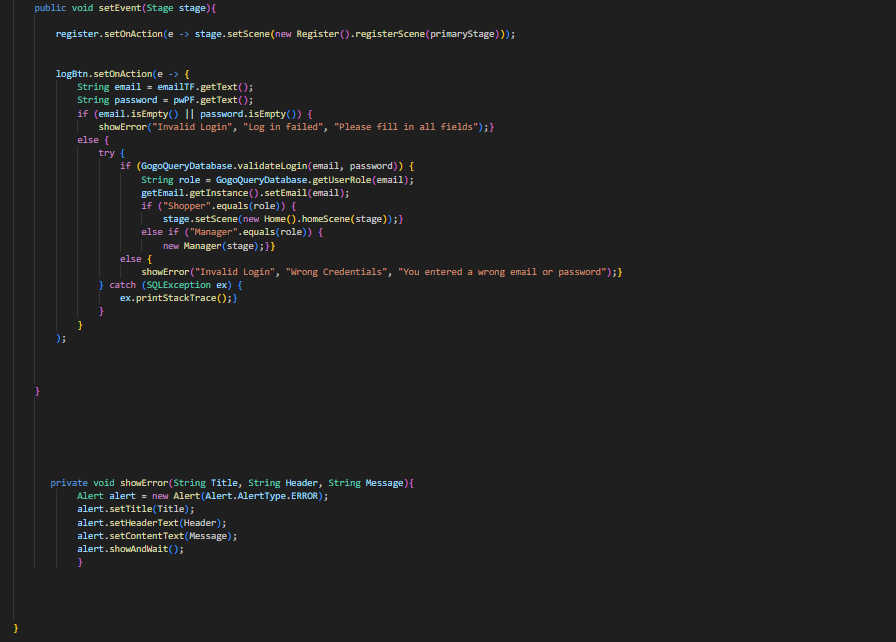
register = new Hyperlink("Register");: Membuat hyperlink dengan teks "Register".

loginBox = new VBox(10, loginText, sep, emailLabel, emailTF, passwordLabel, pwPF, logBtn, reg);: Mengelompokkan beberapa komponen dalam sebuah VBox dengan jarak 10 pixel antar komponen.

whiteBox = new VBox();: Membuat VBox untuk menampung loginBox dan memberikan gaya background dan shadow.

loginMain = new VBox(50, title, whiteBox);: Mengelompokkan title dan whiteBox dalam sebuah VBox dengan jarak 50 pixel antar komponen.

loginLayout.setCenter(loginMain);: Menempatkan loginMain di bagian tengah dari BorderPane.



register.setOnAction: Bagian ini mengatur aksi untuk tombol registrasi. Ketika tombol ini ditekan, aplikasi akan berpindah ke scene registrasi.

loginBtn.setOnAction: Bagian ini mengatur aksi untuk tombol login. Ketika tombol ini ditekan, kode di dalamnya akan dieksekusi.

Validasi Input:

if (email.isEmpty() || password.isEmpty()) : Jika email atau password kosong, maka akan muncul pesan error menggunakan fungsi showError.

Koneksi ke Database dan Validasi:

GogoQueryDatabase.validateLogin(email, password): Fungsi ini (yang asumsinya ada di kelas GogoQueryDatabase) akan mengecek ke database apakah kombinasi email dan password valid.

String role = GogoQueryDatabase.getUserRole(email): Fungsi ini akan mengambil peran pengguna berdasarkan email.

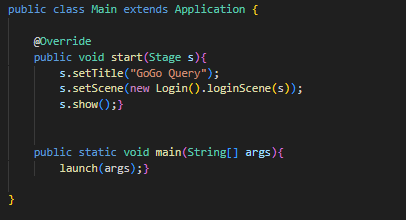
Pengaturan Scene:

Jika login berhasil dan pengguna adalah Shopper, maka aplikasi akan berpindah ke scene Home.

Jika login berhasil dan pengguna adalah Manager, maka akan dibuat objek baru dari kelas Manager. Jika login gagal, akan muncul pesan error.

showError: Fungsi ini digunakan untuk menampilkan dialog error dengan judul, header, dan pesan yang sesuai.

**Class: Main**



@Override dan start(Stage s): Metode start ini adalah titik awal aplikasi JavaFX. Ia menerima objek Stage (jendela utama) sebagai parameter.

s.setTitle("GoGo Query"): Memberikan judul "GoGo Query" pada jendela aplikasi.

new Login().loginScene(s): Membuat objek Login baru, memanggil metode loginScene untuk mendapatkan tampilan login, lalu mengatur tampilan tersebut sebagai isi jendela utama.

s.show(): Menampilkan jendela aplikasi di layar.

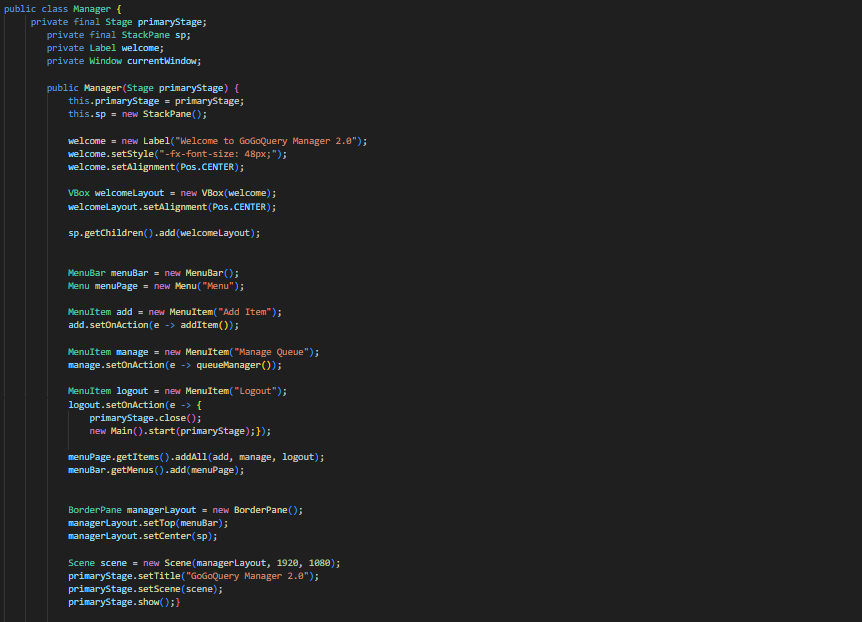
public static void main(String[] args) dan launch(args): Metode main adalah titik masuk utama program Java. launch memulai aplikasi JavaFX.

Metode loginScene:

Metode ini memiliki satu parameter, yaitu Stage stage, yang mereferensikan ke jendela utama aplikasi. Fungsi utama metode ini adalah untuk:

* Inisialisasi primaryStage: Menyimpan referensi ke jendela utama dalam variabel primaryStage.
* Memanggil loginScene(): Metode ini kemungkinan berisi logika untuk membuat dan mengatur tampilan login.
* Memanggil setEvent(): Metode ini kemungkinan berisi logika untuk menghubungkan event (misalnya, ketika tombol "Login" ditekan) dengan aksi yang akan dilakukan.
* Membuat Scene: Membuat objek Scene baru dan mengatur ukurannya.
* Mengembalikan Scene: Mengembalikan objek Scene yang telah dibuat.

**Class: Manager**



Deklarasi Variabel

primaryStage: Variabel ini merepresentasikan jendela utama aplikasi. Ini adalah titik awal untuk semua komponen UI lainnya.

sp: Ini adalah sebuah StackPane yang digunakan sebagai wadah utama untuk mengatur tata letak komponen-komponen lainnya.

currentWindow: Variabel ini kemungkinan digunakan untuk melacak jendela aktif saat ini, meskipun penggunaannya tidak terlihat jelas dalam potongan kode ini.

Konstruktor  
primaryStage: Parameter ini diberikan saat objek Manager diinstansiasi. Ini adalah referensi ke jendela utama yang akan diatur oleh objek Manager ini.

Pembuatan label selamat datang: Dibuat sebuah label dengan teks "Welcome to GoGoQuery Manager 2.0" dan diatur gaya font-nya.

Pengaturan tata letak: Label selamat datang ditempatkan di tengah StackPane menggunakan VBox dan pengaturan setAlignment.

Pembuatan menu bar: Dibuat sebuah menu bar dengan menu "Menu" yang berisi item menu "Add Item", "Manage Queue", dan "Logout".

Pengaturan event handler: Setiap item menu memiliki event handler yang akan dipanggil ketika item tersebut dipilih. Event handler ini kemungkinan akan memicu aksi tertentu, seperti membuka jendela baru atau menutup aplikasi.

Pembuatan BorderPane: BorderPane digunakan untuk mengatur tata letak utama. Menu bar ditempatkan di bagian atas (setTop) dan StackPane (yang berisi label selamat datang) ditempatkan di bagian tengah (setCenter).

Pembuatan scene: Dibuat sebuah Scene baru dengan ukuran 1920x1080 piksel.

Pengaturan scene pada stage: Scene yang baru dibuat ditetapkan sebagai scene untuk primaryStage.

Menampilkan stage: Jendela utama ditampilkan di layar.



Pembuatan Jendela Baru:

Dibuat jendela baru untuk menampilkan form tambah item.

Jendela ini diposisikan di tengah layar.

Pembuatan Label dan Field:

Dibuat berbagai label dan field untuk input data item seperti nama item, kategori, deskripsi, harga, dan kuantitas.

Setiap label dan field diatur gaya dan ukurannya.

Tombol Tambah:

Dibuat tombol "Add Item" dengan gaya tertentu.

Ditambahkan event handler pada tombol untuk menjalankan fungsi addItem ketika tombol ditekan.

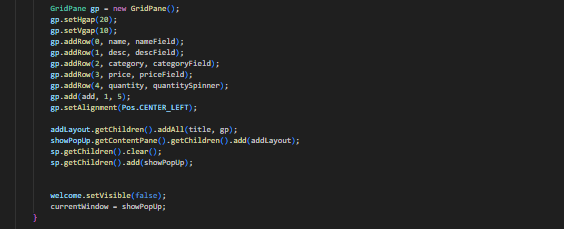
Fungsi addItem:

Fungsi ini mengambil data dari field-field input.

Melakukan validasi data untuk memastikan semua field terisi dan data yang dimasukkan valid (misalnya, harga harus berupa angka).

Jika data valid, maka data akan ditambahkan ke database atau struktur data yang sesuai.

Jika ada kesalahan, maka akan ditampilkan pesan error kepada pengguna.



Membuat GridPane:

GridPane gp = new GridPane(); : Membuat sebuah objek GridPane baru yang akan digunakan sebagai layout utama untuk mengatur posisi komponen-komponen secara grid (baris dan kolom).

Mengatur Jarak Antar Komponen:

gp.setHgap(28); dan gp.setVgap(10); : Mengatur jarak horizontal (28 pixel) dan vertikal (10 pixel) antara setiap komponen dalam grid.

Menambahkan Komponen ke Grid:

gp.addRow(0, name, nameField); dan seterusnya: Menambahkan komponen-komponen (label dan field input) ke dalam grid pada baris dan kolom yang ditentukan. Contohnya, name dan nameField akan ditempatkan pada baris 0 (baris pertama) dan kolom 0 (kolom pertama).

Menyejajarkan Grid:

gp.setAlignment(Pos.CENTER\_LEFT); : Menyejajarkan seluruh isi grid ke kiri tengah.

Menambahkan Grid ke Container Lain:

addLayout.getChildren().addAll(title, gp); : Menambahkan grid (bersama dengan komponen judul title) sebagai anak dari sebuah container yang bernama addLayout.

Menampilkan Popup:

showPopUp.getContentPane().getChildren().add(addLayout); : Menambahkan addLayout ke dalam konten dari sebuah popup yang bernama showPopUp.

sp.getChildren().clear(); : Membersihkan semua anak dari sp (mungkin sebuah StackPane).

sp.getChildren().add(showPopUp); : Menambahkan popup ke dalam sp.

welcome.setVisible(false); : Menyembunyikan komponen welcome.

currentWindow = showPopUp; : Mengatur currentWindow menjadi showPopUp untuk menunjukkan bahwa jendela aktif saat ini adalah popup.



Membuat Jendela Baru:

Window showPopUp = new Window("Queue Manager"); : Membuat sebuah jendela baru dengan judul "Queue Manager".

Mengatur Ukuran Jendela:

showPopUp.setPrefSize(1728, 888); : Mengatur ukuran preferensi jendela.

Membuat Tabel:

TableView<Transaction> table = new TableView<>(); : Membuat sebuah tabel untuk menampilkan data transaksi.

Mendefinisikan Kolom Tabel:

TableColumn<Transaction, Integer> idCol = new TableColumn<>("ID"); dan seterusnya: Mendefinisikan kolom-kolom dalam tabel dengan tipe data yang sesuai (misalnya, Integer untuk ID, String untuk nama pelanggan).

Mengisi Data Tabel:

table.getItems().setAll(GogoQueryDatabase.fetchAllTransactions()); : Mengisi tabel dengan data transaksi yang diambil dari database menggunakan fungsi GogoQueryDatabase.fetchAllTransactions().

Menambahkan Tombol:

Button sendBtn = new Button("Send Package"); : Membuat tombol dengan teks "Send Package".

Menambahkan Event Handler pada Tombol:

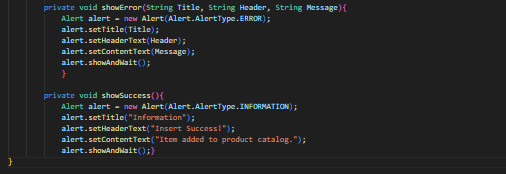
Ketika tombol ditekan, akan dilakukan pengecekan terhadap transaksi yang dipilih. Jika transaksi valid, maka status transaksi akan diubah menjadi "Sent" dan tabel akan di-refresh.

Menyusun Layout:

Komponen-komponen seperti tabel dan tombol disusun dalam sebuah layout (VBox) dan kemudian ditambahkan ke dalam jendela popup.

Menampilkan Popup:

Jendela popup ditampilkan dan jendela sebelumnya disembunyikan.



showError: Menampilkan dialog pesan kesalahan (error).

showSuccess: Menampilkan dialog pesan keberhasilan (sukses).

Membuat Objek Alert:

Kedua fungsi memulai dengan membuat objek Alert baru. Objek ini akan menjadi wadah untuk pesan yang akan ditampilkan.

Tipe alert ditentukan berdasarkan jenis pesan yang akan ditampilkan: AlertType.ERROR untuk pesan kesalahan dan AlertType.INFORMATION untuk pesan informasi.

Mengatur Isi Pesan:

Setelah membuat objek Alert, properti-properti seperti judul, header, dan isi pesan diatur menggunakan metode seperti setTitle, setHeaderText, dan setContentText.

Pada fungsi showError, isi pesan dapat disesuaikan berdasarkan parameter yang diberikan. Sedangkan pada fungsi showSuccess, isi pesan sudah didefinisikan secara hardcode.

Menampilkan Dialog:

Metode showAndWait() digunakan untuk menampilkan dialog pesan. Metode ini akan menghentikan eksekusi program hingga pengguna menutup dialog.