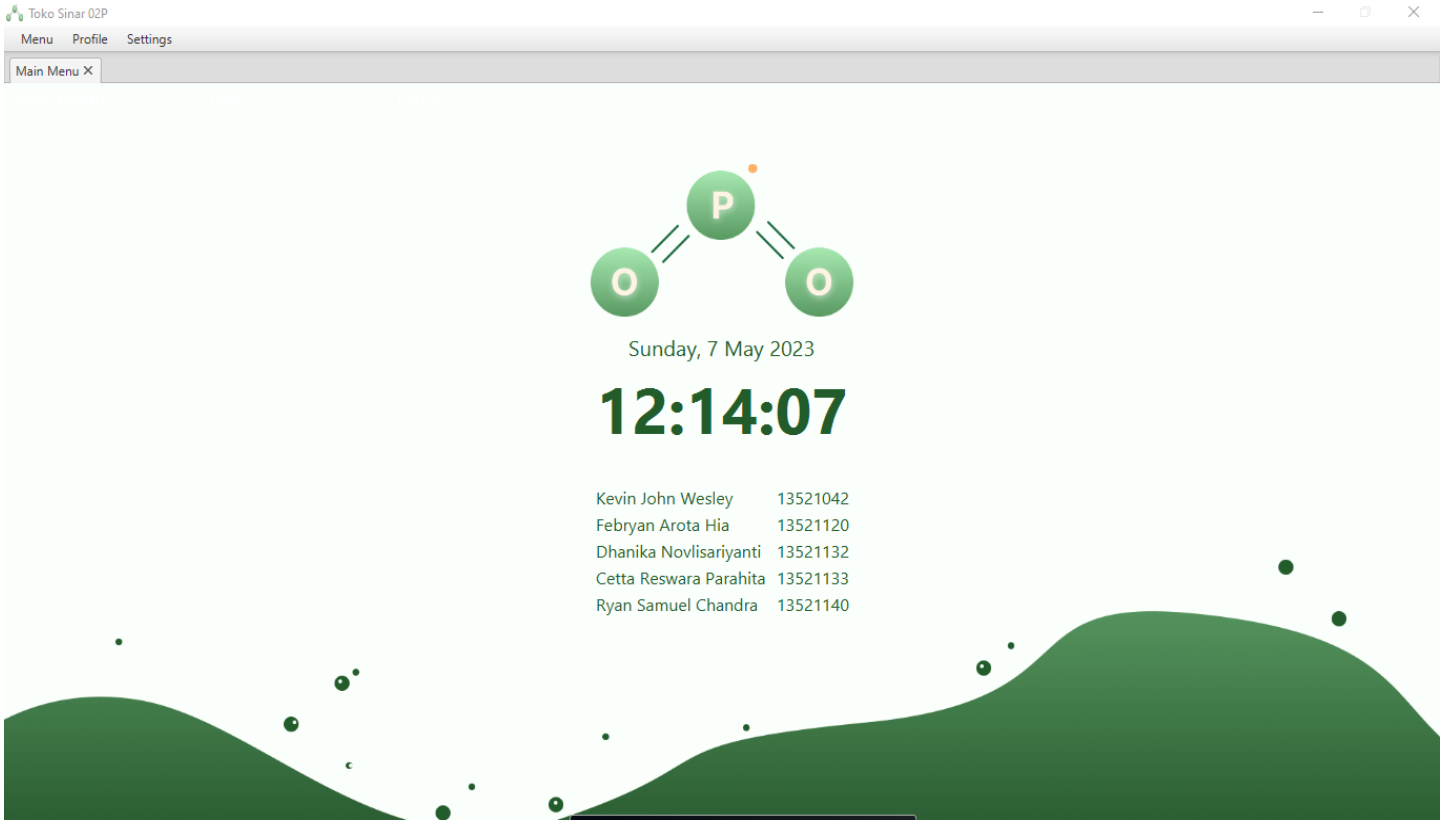
Kode Kelompok : O2P
Nama Kelompok : Oksigen Fosfor
1. 13521042 / Kevin John Wesley Hutabarat
2. 13521120 / Febryan Arota Hia
3. 13521132 / Dhanika Novlisariyanti
4. 13521133 / Cetta Reswara Parahita
5. 13521140 / Ryan Samuel Chandra

Asisten Pembimbing : Aditya Bimawan

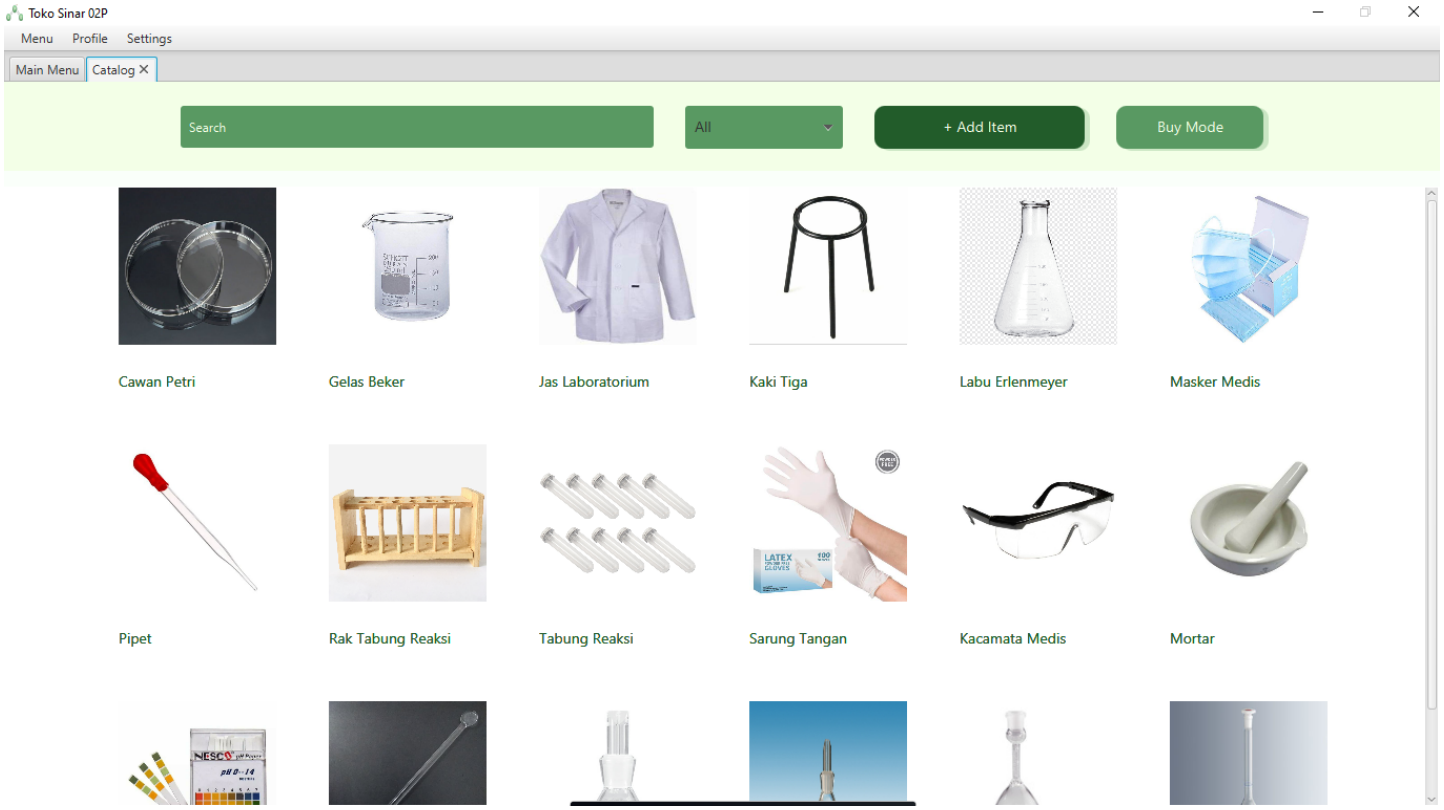
1. Deskripsi Umum Aplikasi

Aplikasi ini adalah aplikasi POS (*Point of Sales*), yaitu aplikasi yang dapat membantu sebuah toko dalam manajemen transaksi yang berhubungan dengan usaha toko tersebut. Di sini kelompok kami membuat aplikasi POS untuk toko yang menjual perlengkapan laboratorium, seperti alat, bahan laboratorium, dan perlengkapan medis. Di dalam aplikasi ini, pelanggan terbagi menjadi tiga jenis, yaitu *customer*, member, dan VIP. Member dan VIP adalah pelanggan yang memiliki *membership*. Seorang *customer* dapat mengaktivasi *membership* ini untuk mendapatkan *reward* berupa poin. Aplikasi ini dilengkapi dengan fitur katalog, yang memudahkan pengguna untuk memilih barang dan kuantitasnya. Setiap barang yang dipilih akan disimpan ke dalam Bill. Saat Bill sudah dibayar, pengguna dapat melakukan *checkout* untuk menyimpan riwayat pembelian ke dalam Fixed Bill.



Gambar 1.1 Tampilan Main Menu

. Semua barang yang sudah dibeli disimpan dalam laporan, yang memudahkan toko untuk melakukan evaluasi mengenai transaksi yang sudah terjadi di toko. Laporan tersebut juga dapat diprint ke dalam *file* .pdf agar lebih mudah dibaca dan dapat digunakan di luar aplikasi. Aplikasi ini dilengkapi dengan *data store*, yang memungkinkan seluruh *history* mengenai pelanggan dan barang di toko dapat disimpan dan digunakan setiap kali aplikasi dijalankan. Aplikasi ini dapat me-*load* plugin, yaitu *plugin chart* yang dapat digunakan untuk visualisasi data dan memudahkan penarikan kesimpulan, dan plugin sistem yang dapat mengubah mata uang, juga menambahkan sistem potongan harga, pajak, dan biaya layanan pada menu pembayaran.



Gambar 1.2 Tampilan Katalog

2. Kakas GUI: JavaFX

Kakas GUI yang digunakan pada tugas besar ini adalah JavaFX. JavaFX merupakan *library* yang digunakan untuk men-*develop* aplikasi *desktop, mobile,* dan sistem *embedded* yang dibangun menggunakan Java. *Life cycle* dari aplikasi ini akan jalan ketika *main program* dijalankan. Jika aplikasi ditutup, maka aplikasi akan berhenti. Cara menggunakan *library* JavaFX adalah setiap halaman akan dibuat sebuah Java Class. Setiap halaman memiliki *button, dropdown,* dan *textfield*-nya masing-masing. Untuk menggunakan komponen tersebut, dibutuhkan *import library* JavaFX. Setelah itu dapat langsung digunakan komponen-komponennya untuk membuat

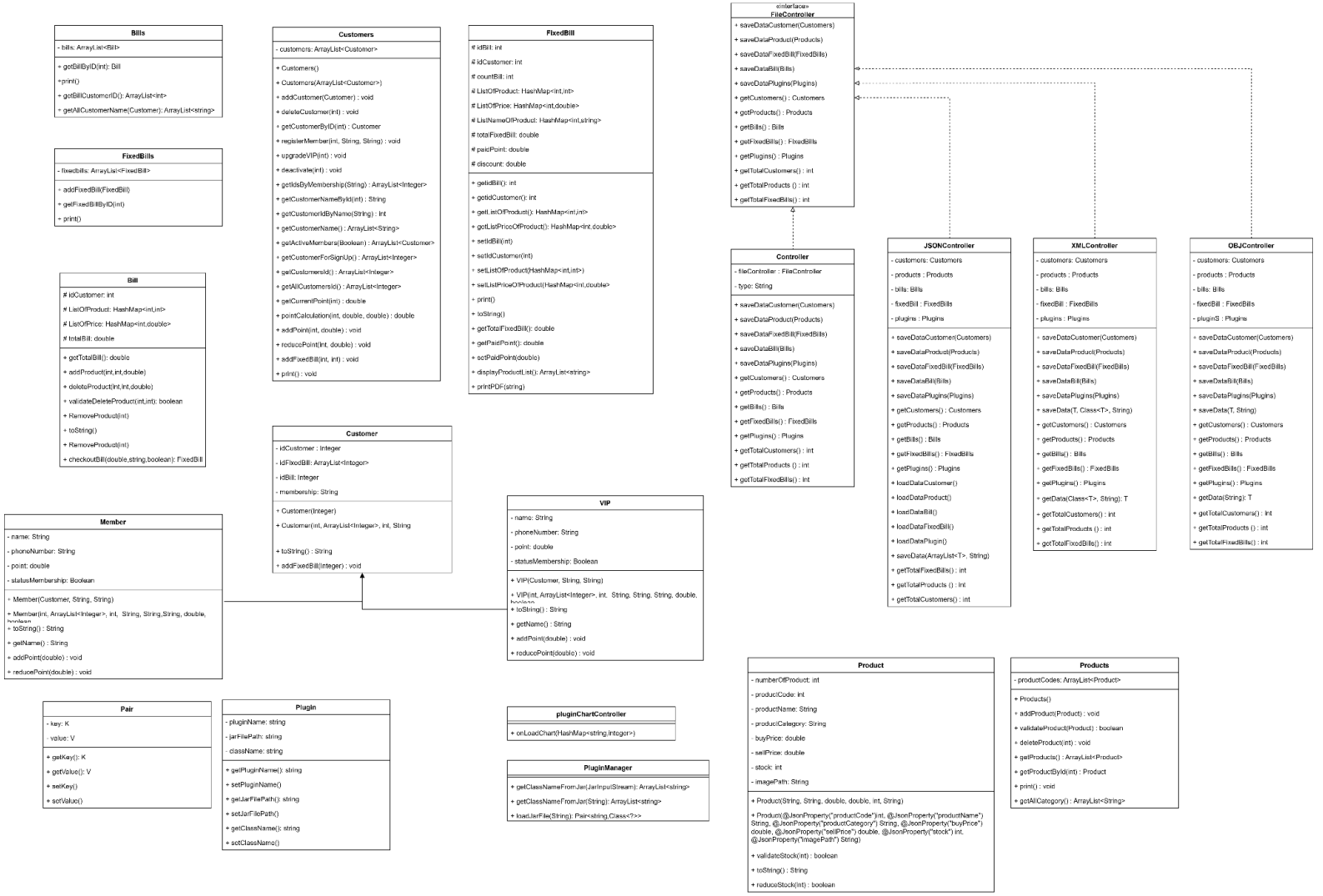
aplikasi. Komponen utama agar GUI dapat muncul pada JavaFX adalah Stage dan Primary Stage. Pada Main.java, akan terdapat Primary Stage yang memiliki semua komponen dari tiap halaman, lalu *stage* tersebut akan diset agar semua komponen dan halaman dapat muncul dalam satu kesatuan aplikasi.Komponen yang disediakan:

1. Controls, seperti *button*, label, *text field*, *text area*, *table view*, dan lain-lain
2. Layout, seperti HBox, VBox, GridPane, ScrollPane, dan lain-lain
3. Scene Graph, seperti Container, Shape, Image, dan lain-lain
4. CSS Styling
5. Animasi dan Transisi
6. Grafik 2D dan 3D
7. Multimedia

3. Plugin & Class Loader

Plugin memiliki kelas sendiri yaitu Plugin yang memiliki *field* nama kelas *plugin*, *file path* dari *plugin* tersebut, serta nama *plugin*. *Plugin* akan di-*load* menggunakan *plugin manager* yang akan me-*load class* dari *plugin* tersebut. Ketika berhasil di-*load*, *plugin* akan diinstansiasi dengan kelas Plugin agar dapat disimpan pada *data store* agar ketika program dijalankan ulang, *plugin* tetap ada. *Plugin* yang di-*load* akan di-*parsing* sesuai *package* dari *plugin* tersebut sehingga dihasilkan com/org.example.*.class. Lalu *class-class* akan di-*load* agar dapat dijalankan oleh program utama jika *plugin* mengimplementasi API yang disediakan oleh program utama.

4. Class Diagram



Program yang telah dibangun memiliki 18 kelas secara keseluruhan. Kelas-kelas tersebut sudah dapat mewakili objek-objek yang terdapat dalam aplikasi POS sesuai dengan spesifikasi yang diberikan. Kelas utama yang mewakili bagian logis dari program, adalah Customer (beserta turunannya), Bill, FixedBill, dan Product. Agar memudahkan operasi teknis yang diinginkan, Bill dikumpulkan dalam Bills, FixedBill dikumpulkan dalam FixedBills, Customer dikumpulkan dalam Customers, dan Product dikumpulkan dalam Products. Dengan demikian, penanganan terhadap objek-objek atomik dapat dilakukan secara kolektif.

Customer adalah kelas yang menangani data-data pelanggan. Kelas Customer memiliki dua kelas turunan yang diperuntukkan bagi customer yang secara khusus telah mendapatkan status *membership*, yaitu kelas Member dan VIP. Masing-masing memiliki keuntungan yang berbeda, terkait dengan potongan harga. Kelas Customer sendiri memiliki sebuah atribut bertipe *string* yang dapat mengklasifikasikan *membership* seorang pelanggan, baik sebagai member, VIP, maupun bukan keduanya. Sedangkan pada kelas turunannya, ditampung atribut *boolean* (statusMembership) yang menandakan apakah *membership* seorang pelanggan masih valid / aktif / berlaku. Kelas Bill adalah kelas yang menangani transaksi sementara dari pelanggan. Kelas ini menampung semua barang beserta kuantitasnya sebelum pelanggan melakukan *checkout*. Kelas Bill ini digunakan agar pelanggan dapat melihat barang apa saja yang sudah diambil. Setelah *checkout*, riwayat transaksi dimasukkan ke *fixed bill* yang ditangani oleh kelas FixedBill. Dalam kelas FixedBill, terdapat juga fitur yang memungkinkan sebuah *fixed bill* dapat dicetak untuk pelanggan. Semua riwayat transaksi yang diterima oleh pengguna disimpan ke dalam laporan, yang ditangani oleh kelas SalesReport. Dengan kelas ini, pengguna dapat melihat *insight* dari riwayat transaksi dan mencetaknya ke *file* berformat PDF. Perlu diperhatikan juga bahwa seorang Customer hanya memiliki satu *bill* pada satu waktu tertentu, tetapi Customer yang sama dapat memiliki lebih dari satu *fixed bill* sekaligus (dicatat semua ID-nya dalam sebuah ArrayList<Integer>). Banyak FixedBill menandakan banyak transaksi yang telah dilakukan oleh Customer tersebut di toko.

Kelas Product menangani produk-produk yang dijual di dalam toko, mulai dari manajemen stok, mekanisme pengambilan produk, harga produk dan lain sebagainya. Di dalam program terdapat sebuah *interface,* yaitu FileController, yang memuat fungsi dan prosedur untuk mengatur penanganan *file* untuk *data store. Interface* ini diimplementasi oleh empat kelas, yaitu kelas Controller, JSONController, XMLController, dan OBJController. Masing-masing dari kelas tersebut menangani tipe *file* yang berbeda sesuai dengan namanya. Di dalam program juga terdapat kelas Plugin dan PluginManager, yang menangani bagaimana sebuah *plugin* akan di-*load* oleh program utama.

5. Konsep OOP

5.1. Inheritance

- Kelas Member (src/main/java/com/o2pjualan/Classes/Member.java) merupakan anak dari kelas Customer (src/main/java/com/o2pjualan/Classes/Customer.java)
- Kelas VIP (src/main/java/com/o2pjualan/Classes/VIP.java) merupakan anak dari kelas Customer (src/main/java/com/o2pjualan/Classes/Customer.java)
- Kelas catalogMenu (src/main/java/com/o2pjualan/GUI/catalogMenu.java) merupakan anak dari kelas Tab
- Kelas mainMenu (src/main/java/com/o2pjualan/GUI/mainMenu.java) merupakan anak dari kelas Tab
- Kelas Main (src/main/java/com/o2pjualan/Main.java) merupakan anak dari kelas Application

5.2. Composition

- Kelas Bills (src/main/java/com/o2pjualan/Classes/Bills.java) memiliki banyak kelas Bill (src/main/java/com/o2pjualan/Classes/Bill.java)
- Kelas Customers (src/main/java/com/o2pjualan/Classes/Customers.java) memiliki banyak kelas Customer (src/main/java/com/o2pjualan/Classes/Customer.java)
- Kelas FixedBills (src/main/java/com/o2pjualan/Classes/FixedBills.java) mengandung banyak kelas FixedBill (src/main/java/com/o2pjualan/Classes/FixedBill.java)
- Kelas Plugin (src/main/java/com/o2pjualan/Classes/Plugin.java) mengandung tepat sebuah kelas *wildcard* yang bernama “pluginClass”, yang harus diisi dengan kelas yang mengimplementasikan *plugin*.
- Kelas Products (src/main/java/com/o2pjualan/Classes/Products.java) memiliki banyak kelas Product (src/main/java/com/o2pjualan/Classes/Product.java)

5.3. Interface

- Interface FileController (src/main/java/com/o2pjualan/Classes/FileController.java)
- Interface pluginChartController (src/main/java/com/o2pjualan/Classes/pluginChartController.java)

5.4. Method Overriding dan Method Overloading

- Method saveDataCustomer(Customers customers) pada kelas Controller (src/main/java/com/o2pjualan/Classes/Controller.java)
- Method getBills() pada kelas OBJController (src/main/java/com/o2pjualan/Classes/OBJController.java)
- Method getTotalCustomers() pada kelas XMLController (src/main/java/com/o2pjualan/Classes/XMLController.java)
- Method toString() pada kelas Bill (src/main/java/com/o2pjualan/Classes/Bill.java)
- Method getClassNameForJar pada kelas PluginManager (src/main/java/com/o2pjualan/Classes/PluginManager.java) dapat dipanggil dengan parameter *string* maupun JarInputStream

5.5. Polymorphism

- *Method* changeExt(String ext) (src/main/java/com/o2pjualan/Classes/FileManager.java) controller dibuat dengan parameter ext. Pada *method* ini *controller* akan memiliki *attribute* FileController sesuai dengan class *extension*-nya.

5.6. Java API Collection

- Kelas Bill (src/main/java/com/o2pjualan/Classes/Bill.java) memiliki Hashmap
- Kelas Customers (src/main/java/com/o2pjualan/Classes/Customers.java) memiliki ArrayList<Customer>
- Constructor pada kelas Product (src/main/java/com/o2pjualan/Classes/Product.java) memanfaatkan java API *stream filter*

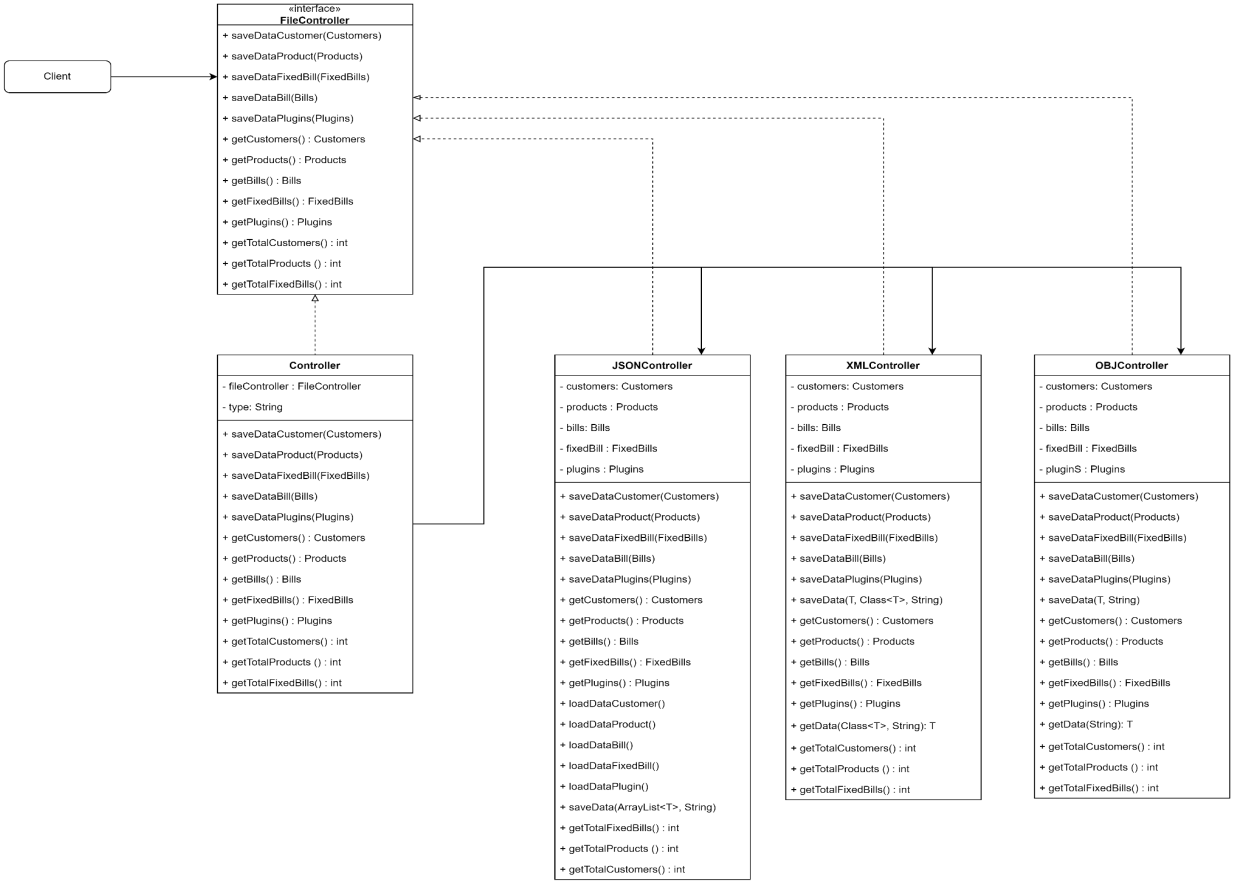
5.7. SOLID

- **S**: Kelas Customer (src/main/java/com/o2pjualan/Classes/Customer.java) memenuhi prinsip ini karena hanya memiliki tanggung jawab untuk melakukan penyimpanan data customer seperti nama, dan nomor telepon, serta menyediakan metode get_details() untuk mengambil rincian pelanggan.
- **O** : Kelas Controller (src/main/java/com/o2pjualan/Classes/Controller.java) memenuhi prinsip ini karena kelas ini tidak perlu dimodifikasi lagi apabila ingin menambahkan controller file (XMLController, JSONController, OBJController) baru.
- **L**: Kelas VIP dan Member, merupakan subclass dari Customer. Kelas ini memenuhi konsep *Liskov Substitution Principle* karena Member dan VIP dan digunakan atau menggantikan kelas Customer ketika diperlukan pada berbagai konteks.

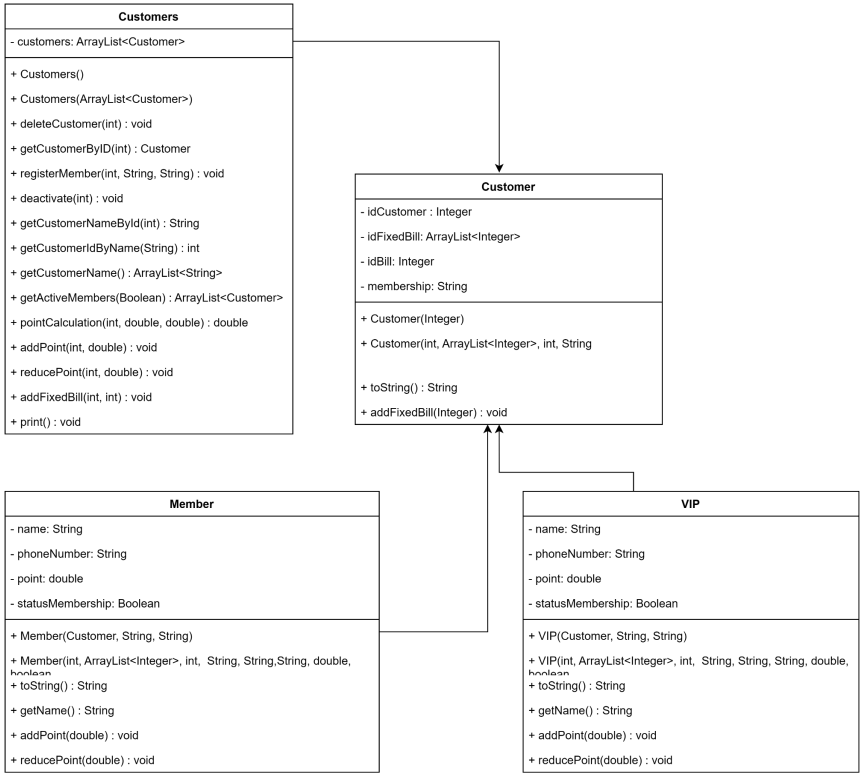
- **I**: Kelas JSONController, XMLController, dan OBJController meng-*implement* kelas FileController yang merupakan *interface*. *Interface* FileController men-*define method* yang digunakan untuk mendapatkan dan menyimpan data. Lalu kelas-kelas yang meng-*implement interface* FileController meng-*implement* sesuai dengan format ekstensi *file* masing-masing.
- **D**: Kelas Controller berfungsi sebagai kelas yang merepresentasikan modul bagi kelas-kelas JSONController, XMLController, OBJController sehingga kelas tersebut akan dependen terhadap kelas Controller. Sehingga kelas Controller dependen terhadap FileController.

5.8. Design Pattern

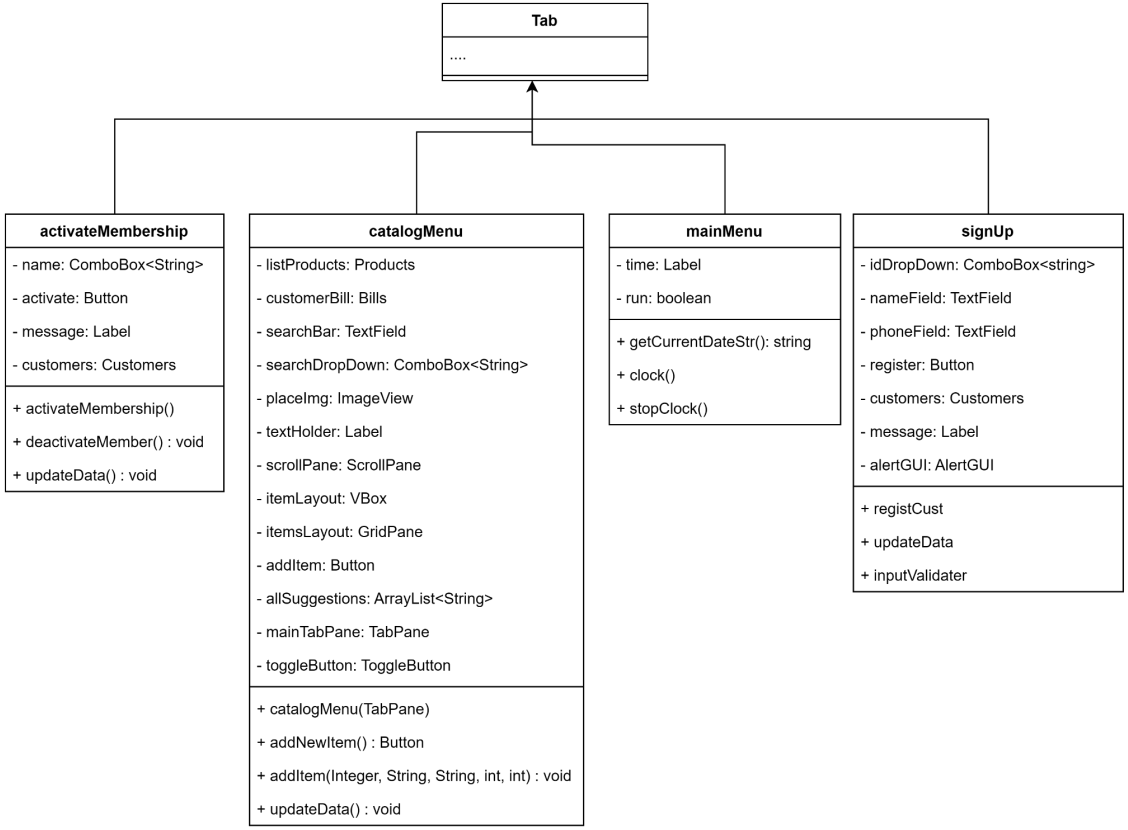
Adapter : Kelas Controller (src/main/java/com/o2pjualan/Classes/Controller.java) sebagai *adapter* yang meng-*implement interface* FileController. Kelas Controller digunakan untuk membaca dan mengolah keseluruhan *data store*. Kelas Controller akan menggunakan Class JSONController, XMLController, dan OBJController untuk mengolah data sesuai dengan *extension* data yang diterima. Kelebihan menggunakan *design pattern* ini adalah *user* tidak perlu lagi memberi tahu *extension* data apa yang akan dibaca.



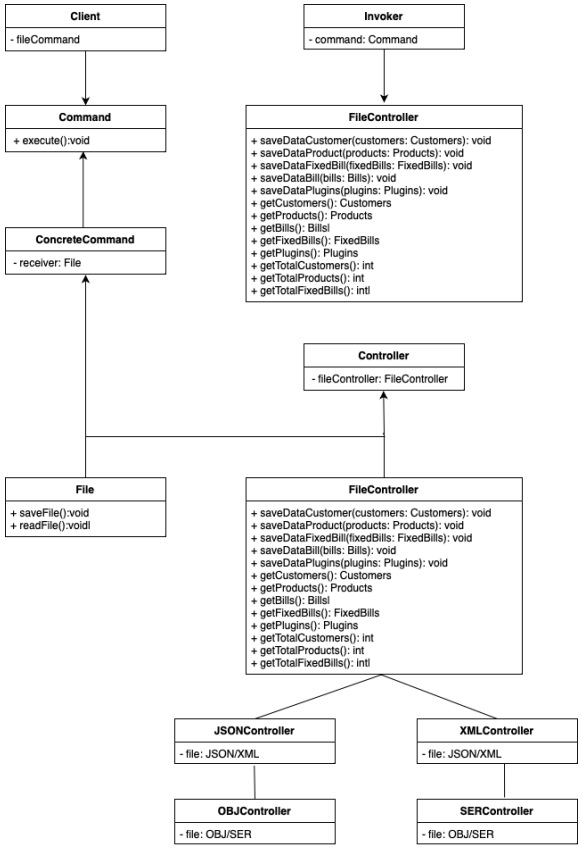
Composite: Pada *design pattern* ini, kelas Customers (src/main/java/com/o2pjualan/Classes/Customer.java) diimplementasikan sebagai *root* dan kelas Customer, Member, dan VIP sebagai *leaf. Design pattern* ini digunakan untuk mengelola Customers sebagai *box* yang berisikan Customer, Member, dan VIP dengan penanganan sesuai dengan implemen masing-masing *leaf*. Kelebihannya ketika ingin mengubah keseluruhan data atau *attribute* pada *leaf* program dapat melakukannya sekaligus melalui *method* yang diimplemen pada *root* (Customers)



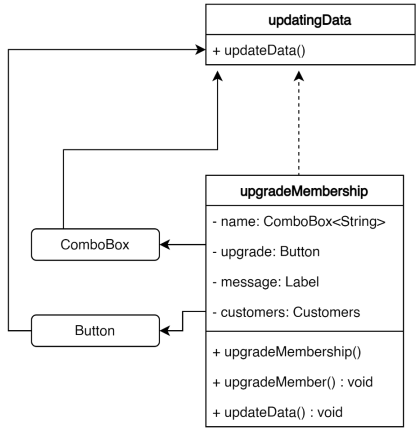
Template Method: Kelas-kelas GUI yang berada pada folder (src/main/java/com/o2pjualan/GUI) menggunakan *design pattern template method* dengan meng-*extend* class Tab yang merupakan *class* dari javafx. Masing-masing *class* meng-*override* fungsi-fungsi dengan menyesuaikan kebutuhannya. Keuntungannya setiap class GUI tidak perlu meng-*override* seluruh *method* class Tab, hanya yang diperlukan saja. Beberapa *method* Tab yang di-*overwritte* adalah setContent(), setOnSelectedChanged(), dll. *Method-method* ini tidak sepenuhnya dijadikan sebagai *method* dari masing-masing *class*, melainkan dipanggil di dalam beberapa *method* sesuai kebutuhan.

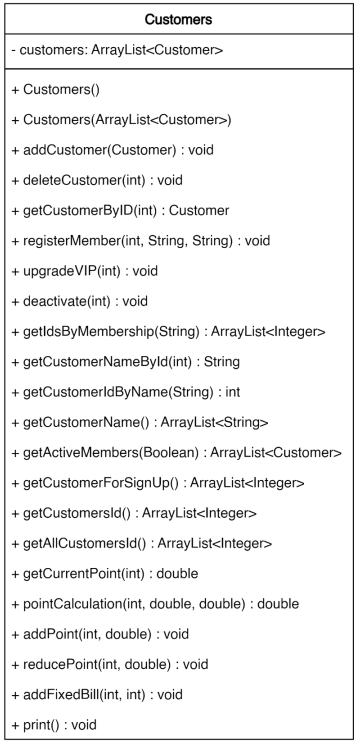


Command: Kelas Controller (src/main/java/com/o2pjualan/Classes/Controller.java) bertanggung jawab untuk memilih implementasi FileController (src/main/java/com/o2pjualan/Classes/FileController.java) yang tepat berdasarkan format *file* yang diinginkan saat dibuat. Implementasi FileController digunakan untuk menyimpan dan mengambil data dari *file* dalam format tertentu. Kelas Controller memiliki beberapa metode yang mengimplementasikan metode dari antarmuka FileController, seperti saveDataCustomer, getCustomers, saveDataProduct, getProducts, dan lainnya. Metode-metode ini digunakan untuk melakukan operasi penyimpanan dan pengambilan data pelanggan, produk, tagihan, dan *plugin* dari *file* sesuai dengan format yang telah ditentukan. Dengan demikian, kelas Controller dapat digunakan sebagai pengontrol untuk berinteraksi dengan *file* dan mengelola data dalam aplikasi.



Observer: Pada program ini kebanyakan kelas GUI mengimplementasi *interface* updatingData. Salah satu contohnya adalah kelas upgradeMembership. Kelas upgradeMembership *implements interface* updatingData. Kelas upgradeMembership digunakan sebagai *observer* dan *listener* pada *field-field* dalam kelas yang memiliki *on change action*. Hubungan observasi antara upgradeMembership terjadi ketika updateData() dari *interface* updatingData dipanggil dan men-*trigger* kelas Customers yang akan meng-*update* datanya.





5.9. Reflection

- *Method* loadJarFile pada (src/main/java/com/o2pjualan/Classes/PluginManager.java) menggunakan *reflection* untuk me-*load class* dari fail JAR.

5.10. Threading

- *Threading* digunakan pada *method clock* di dalam *class* mainMenu (src/main/java/com/o2pjualan/GUI/mainMenu.java). Dibuat *threading* agar *display clock* tidak menghalangi proses lainnya sehingga *display clock* tetap berjalan berdampingan dengan program.
- *Threading* digunakan pada *method print* PDF di dalam *class report* (src/main/java/com/o2pjualan/GUI/report.java) dan BillGUI (src/main/java/com/o2pjualan/GUI/BILLGUI.java). Dibuat *threading* agar ketika mengubah menjadi PDF tidak menghalangi proses *main thread* GUI aplikasi dan aplikasi tetap dapat berjalan.
- Threading saat mengganti ekstension file pada data store settings. Dibuat threading agar aplikasi tidak hang ketika saat pengubahan pengguna yang tidak sengaja menekan hal lain tidak menyebabkan program terminate.

6. Pembagian Tugas

NIM	Nama	Tugas
13521042	Kevin John Wesley Hutabarat	1. Membuat implementasi kelas Bill, FixedBill 2. Laporan
13521120	Febryan Arota Hia	1. Membuat keseluruhan data store berserta controller dan file manager 2. Implementasi data store adapter untuk mengubah ke format lain 3. Bertanggung jawab untuk tampilan dan penggunaan Main Menu, Sign Up, (De)activate Membership, Upgrade Membership, Settings Data Store, Remove Item 4. Laporan
13521132	Dhanika Novlisariyanti	1. Membuat mekanisme plugin 2. Membuat plugin pie chart dan bar chart 3. Bertanggung jawab untuk tampilan dan penggunaan Catalog Menu, Add Item, Edit Item, Report, History, print PDF, plugin 4. Laporan
13521133	Cetta Reswara Parahita	1. Laporan
13521140	Ryan Samuel Chandra	1. Implementasi inisialisasi kelas Customer dan turunannya, Customers, Product, dan Products 2. Laporan

7. Foto Kelompok

Kode Kelompok : O2P
Nama Kelompok : Oksigen Fosfor
1. 13521042 / Kevin John Wesley Hutabarat
2. 13521120 / Febryan Arota Hia
3. 13521132 / Dhanika Novlisariyanti
4. 13521133 / Cetta Reswara Parahita
5. 13521140 / Ryan Samuel Chandra

1. Konten Diskusi

Konten diskusi berisi setiap pertanyaan dan jawaban yang dibahas saat asistensi. Dapat ditulis dalam format poin-per-poin atau paragraf.

Siapkan pertanyaan, karena keaktifan kelompok dinilai.

No.	Pertanyaan	Jawaban
1.	Apakah boleh pakai controller seperti di RPL, Misalnya query untuk getProduct, return list of product saja atau tetap ada kelas productnya?	misalnya mau ngambil product dari datastore, cuma simpan ID-nya aja. Itu menyalahkan aturan OOP. Tapi kan sebenarnya tetap masih ada objek untuk menyimpannya. Kalau mau coba bikin diagramnya dulu kalau mau pake controller, nanti di-review.
2.	Kalau mau ngesave member, apakah harus bikin method di JSON reader atau langsung di class membernya?	Bisa bikin method di interface, bisa juga bikin di JSON reader tapi jadi banyak, harus ada untuk setiap kelas.
3.	Apakah boleh dijelasin lebih detail tentang plugin?	Plugin -> ada program, satu program dicompile, compilenya harus bisa terima satu file jar. Plugin bisa dimasukin ke programnya secara dinamis,

		ga harus dicompile ulang. Misalnya jar implement interface, terus bisa dipanggil methodnya. Jadi compile terpisah.
4.	Untuk GUI, rencananya mau pakai javaFX. Untuk construct GUI apakah boleh pake FXML (sejenis HTML)?	Tanya di QnA aja biar kelompok lain tahu juga.
5.	Dalam fitur search, apakah boleh pake library java yang regex?	Boleh. Tanya di QnA juga.
6.	Laporan penjualan isinya apa aja yang di-print ke PDF?	Penjualannya aja, kalau mau tambahin untungnya juga boleh, formatnya bebas.
7.	customer.XML mau ubah ke JSON, yang XML dihapus, dibiarin aja, atau diubah?	Ga harus dihapus.
8.	Kalau ada atribut yang bisa diturunkan dari atribut lain (misal total penjualan) perlu disimpan atau cukup pakai method saja?	Bebas, sesuai desain aja

2. Tindak Lanjut

Menginisialisasi kelas-kelas, merencanakan dan mengimplementasikan semua keterhubungan antarkelas. Mencoba akses data dari *data store* hingga bisa digunakan dalam keberjalanan aplikasi. Merancang dan merealisasikan tampilan yang menarik untuk setiap fitur dalam aplikasi.

3. Foto Dokumentasi