

# Spesifikasi Tugas Besar 2: Manajemen Usaha BNMO

## IF2210 Pemrograman Berorientasi Objek

Deadline: 3 Mei 2023 22:10 WIB

### Latar Belakang Persoalan



Sumber Gambar: <https://adventuretime.fandom.com/wiki/Squeez-E-Mart>

*"Toko sebelum direnovasi."*

---

Setelah gagal pada bisnis mereka sebelumnya, Indra dan Doni ingin membuka bisnis baru yaitu sebuah toko DIY. Untuk menopang bisnis mereka ini, mereka ingin menggunakan BNMO sebagai sistem POS (*Point of Sales*). Akan tetapi, BNMO belum memiliki fitur POS yang mereka inginkan. Oleh karena itu, Indra dan Doni ingin meminta kalian, mahasiswa IF' 21, untuk menambahkan fitur baru pada BNMO agar mereka dapat menjalankan bisnis baru mereka dengan lancar.

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>Revision/Changelog</b>	<b>2</b>
<b>Link Penting</b>	<b>3</b>
<b>Tujuan</b>	<b>3</b>
<b>Program secara Abstrak</b>	<b>3</b>
<b>Spesifikasi Sistem</b>	<b>4</b>
GUI	4
IDE	4
Build Automation Tools	4
Ketentuan Teknis	5
<b>Mekanisme &amp; Fitur</b>	<b>7</b>
● Halaman Utama	7
● Menu dan Tab	7
● Customer, Member dan VIP	8
● Sistem Usaha Barang	9
● Sistem Laporan	12
● Halaman Setting	12
● Data Store	12
● Plugin	13
● Komponen Plugin	14
<b>Bonus &amp; Eksplorasi</b>	<b>17</b>
<b>Panduan Pengerjaan</b>	<b>18</b>
<b>Panduan Laporan</b>	<b>19</b>
<b>Asistensi &amp; QnA</b>	<b>20</b>
<b>Pengumpulan</b>	<b>20</b>
<b>Penilaian</b>	<b>21</b>

## Revision/Changelog

1. Belum ada. Pasti akan ada revisi apabila ada pertanyaan dari kalian yang sekiranya akan mengubah spesifikasi.
2. 11 April 2023, 21.20 WIB.
  - a. Penambahan **deadline asistensi.**

## Link Penting

1. QnA: [QnA IF2210 2022/2023](#)
2. Kelompok dan Asisten: [IF2210 OOP - Kelompok](#)
3. Pengumpulan: <https://forms.gle/Yj1jG5DvMEo9ABk48>
4. Format Laporan: [IF2210\\_TB2\\_Laporan\\_XXX](#)
5. Format Dokumen Asistensi: [IF2210\\_TB2\\_Asistensi\\_XXX.docx](#)
6. Spesifikasi Bonus Extended: [Spesifikasi EXTENDED Tugas Besar 2](#)

**Tips: Diskusikan desain program dengan baik bersama kelompok sebelum memulai coding.**

## Tujuan

1. Mahasiswa dapat memahami dan mempraktikkan konsep-konsep dalam *Object oriented programming*.
2. Mahasiswa dapat memahami cara kerja Graphical User Interface.
3. Mahasiswa dapat memakai Wildcard dan Java API: Collection, Reflection dan Threading.
4. Mahasiswa dapat mengembangkan aplikasi berbasis Java di IDE dan menggunakan *build tools* sebagai alat bantu.
5. Mahasiswa dapat mengerti dan mempraktikkan *design pattern*.
6. Jika mengerjakan bonus, Mahasiswa diharapkan dapat bereksplorasi dan mengerti pemakaian *library-library* Java pada umumnya seperti Lombok, Null Safety Annotation, Unit Testing, JDBC, Thread Pooling, Connection Pooling, dan ORM.

## Program secara Abstrak

Program yang harus Anda buat pada Tugas Besar 2 ini adalah program POS (*Point of Sales*), yaitu program yang membantu sebuah toko untuk melakukan dan mencatat transaksi yang berhubungan dengan usaha mereka.

Program POS yang akan Anda buat memiliki fitur dasar manajemen inventaris dan manajemen transaksi. Selain itu, program memiliki fitur *membership* agar toko dapat memberikan *reward* kepada pelanggan yang setia, dan juga fitur pembuatan laporan untuk mendukung toko dalam melakukan evaluasi. Program juga bersifat *extensible* dengan menyediakan dukungan *plugin*, sehingga pengguna dapat menambahkan fungsionalitas program dengan mudah.

# Spesifikasi Sistem

## GUI

Dalam pengerjaan Tugas Besar 2, peserta mata kuliah diwajibkan untuk membuat GUI dalam bahasa Java.

Diperbolehkan untuk menggunakan library/tools untuk membuat user interface (misal: Swing, JavaFX, dll). Penggantian versi Java yang digunakan jika perlu menggunakan suatu framework GUI yang hanya dapat digunakan di versi tertentu diperbolehkan, sehingga tidak harus menggunakan JDK 8/Java 8. Jika kalian memakai Java Swing, terdapat komponen-komponen umum yang dapat dilihat cara penggunaan dan visualisasinya pada [tautan berikut](#).

Desain antarmuka dibebaskan kepada setiap kelompok, namun buatlah semenarik mungkin. Anda harus tetap memahami bagian-bagian antarmuka tersebut meskipun menggunakan *tools/library*. Namun jangan lupa untuk menyelesaikan fungsionalitas lengkap dari sistem.

## IDE

Sangat disarankan untuk menggunakan IDE, misalnya IntelliJ IDEA. Dengan email student/std, Anda bisa mendapat license versi Ultimate. Dengan IntelliJ IDEA juga dapat menginstall dan memilih versi JDK serta memilih Build Tools yang ingin dipakai.

## Build Automation Tools

*Build Automation Tools* seperti **maven** dan **gradle** adalah kakas yang paling umum digunakan ketika melakukan pengembangan perangkat lunak berbasis Java. Baik aplikasi *desktop*, *web service* maupun aplikasi berbasis web, kakas ini sangat berguna. Pada Tugas Besar ini, gunakan **maven** atau **gradle** untuk menjadi *build tools* dan *package manager*. Kegunaan paling umum dari *tools* ini adalah untuk menambahkan *library* eksternal dengan mudah. Untuk penggunaannya, silakan dieksplorasi masing-masing kelompok. Penggunaan kakas ini **akan menyulitkan diawal terutama saat mempelajarinya namun akan sangat membantu pengerjaan karena mempercepat proses kompilasi.**

*Catatan: kakas ini harus dipelajari dan digunakan sebelum kelompok anda mengerjakan **spesifikasi**, mengerjakan spesifikasi kakas ini diakhir akan menyulitkan kelompok kalian sendiri. Kalau mau pake library eksternal di Java, ribet banget manual, sedangkan di tugas besar ini kalian bakal banyak pakai library eksternal.*

## Ketentuan Teknis

Berikut adalah hal-hal yang **minimal** wajib diimplementasikan di aplikasi yang Anda buat. Perlu dicatat, yang ditulis disini hanyalah minimal dan sangat besar kemungkinan kelompok kalian akan menambah abstraksi/konsep lainnya untuk membuat sistem ini.

1. Inheritance
2. Composition
3. Interface
  - a. Wajib menggunakan interface Serializable dari Java (lebih lengkap cek bagian Data Store)
  - b. Wajib ada penggunaan interface selain nomor 3a.
4. Polymorphism
5. Method Overriding dan Method Overloading
6. Java API Collection
7. Prinsip [S.O.L.I.D.](#)



### **S**ingle Responsibility Principle

A class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)



### **O**pen / Closed Principle

A software module (it can be a class or method) should be open for extension but closed for modification.



### **L**iskov Substitution Principle

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.



### **I**nterface Segregation Principle

Clients should not be forced to depend upon the interfaces that they do not use.



### **D**ependency Inversion Principle

Program to an interface, not to an implementation.

8. Design Pattern
  - a. [Adapter](#): rekomendasi penggunaan di bagian data store.
  - b. 4 Design Pattern Lain (Total jadi 5 sama Adapter diatas)  
Catalog: <https://refactoring.guru/design-patterns/catalog>
9. Reflection
  - a. Wajib dipakai pada sistem plugin (lebih lengkap cek bagian plugin).
  - b. Wajib ada penggunaan lain selain nomor 9b.
10. Threading
  - a. Wajib: Jam Digital pada Halaman Utama

- b. Nilai tambah diberikan jika ada penggunaan lain selain jam digital di halaman utama dan yang ditulis dispesifikasi.

Hal-hal yang harus diperhatikan:

- **Hindari menggunakan konsep OOP diatas hanya karena diwajibkan saja! Gunakanlah konsep OOP yang sesuai dengan kasusnya!**
- DRY (Don't Repeat Yourself), tidak memiliki kode duplikat, pindahkan ke fungsi/*class*.
- Memiliki struktur kelas yang mudah dipahami.
- Dekomposisi yang baik dan implementasi yang tidak terlalu kompleks (sebuah method tidak terlalu panjang). Pecah-pecah dan buat method baru agar tidak terlalu kompleks.
- Maksimalkan SOLID terutama pada *open-closed principle* dan *dependency inversion*. Karena tugas besar ini memiliki sistem plugin.
- Usahakan memaksimalkan penggunaan konsep OOP, terutama konsep *polymorphism*, *inheritance*, *composition* dan sebagainya.

**Assistant's Note:**

Pada Tugas Besar 1, **masih ada** kelompok yang paradigmanya prosedural, membuat kelas hanya menjadi penamaan modul saja, sisanya logikanya masih terlalu banyak if else dan kurang memanfaatkan *polymorphism*, bahkan *subclass* sangat minimal dan tidak tepat penggunaannya. Pastikan semua aspek OOP dipakai sesuai dengan *best case* yang kalian pelajari ya.

Sebagai *insight*, kalau kode kalian desainnya bagus dan tentunya rapih, kode yang panjang bisa menjadi *compact* dan pendek dan bisa diubah-ubah ke logik yang lain. Pada Tugas Besar 1, **ada** kelompok yang kodenya lebih pendek dan rapi namun selesai sampai bonus. Bahkan, kodenya relatif lebih pendek\* dibandingkan dengan kelompok yang tidak selesai sampai bonus. *\*dikonfirmasi tidak curang karena memakai file yang dikumpulkan di form.*

*Peringatan ini jelas karena kalian mengerjakan tugas besar mata kuliah pemrograman berorientasi objek bukan sekedar menyelesaikan spesifikasi.*

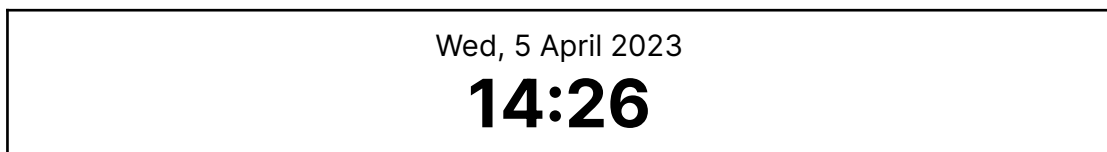
## Mekanisme & Fitur

Program yang Anda buat harus memenuhi beberapa mekanisme. Berikut merupakan mekanisme yang harus Anda implementasikan di program Anda:

- **Halaman Utama**

Pada halaman utama, terdapat sebuah logo aplikasi kalian, NIM, Nama dari setiap anggota kelompok kalian dan Jam Digital.

Jam digital memiliki format sebagai berikut.

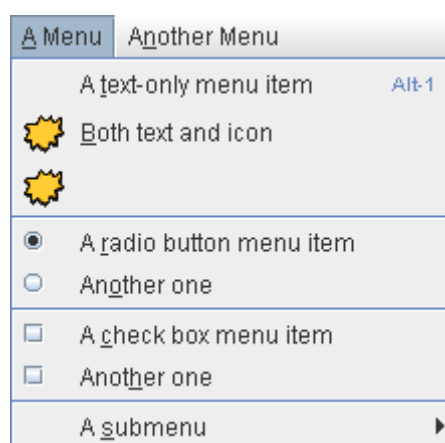


Karena jam digital yang diinginkan merupakan penunjuk waktu secara *real-time*, maka jam digital tersebut harus diimplementasikan menggunakan *threading* supaya tidak memblok menu utama dan menyebabkan *freeze*.

Jika kalian ingin menyalurkan kreativitas di halaman ini, maka diperbolehkan dengan syarat tidak ada kekurangan dari spek yang diminta. Dengan kata lain kalian boleh menambah tetapi tidak boleh mengurangi. Contoh kreativitas: logo aplikasi tidak statik, seperti logo beranimasi sederhana dengan *threading*.

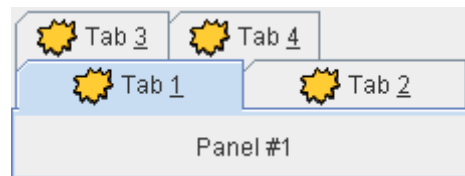
- **Menu dan Tab**

Untuk setiap halaman yang tersedia, dimuat dalam sebuah menu sebagai berikut. Sesuaikan halaman yang tersedia dengan contoh gambar berikut.



Sumber Gambar: [Web MIT](https://web.mit.edu)

Dan untuk setiap halaman yang sudah dibuka akan dibuat menjadi tab yang bisa ditutup. Secara umum, bentuknya akan seperti *browser web*. Untuk visualisasi agar lebih jelas, lihat gambar berikut.



Sumber Gambar: [Web MIT](#)

## ● Customer, Member dan VIP

Customer adalah pelanggan biasa yang tidak memiliki keistimewaan. Pelanggan ini hanya memiliki id yang akan bertambah setiap pesanan yang dibuat. Tidak ada nama karena customer ini tidak terdaftar.

Sedangkan Member adalah pelanggan yang memiliki id, nama, dan nomor telepon yang sudah terdaftar di dalam sistem. Keistimewaan dari Member adalah setiap pesanan yang dibuat dapat menghasilkan poin. Poin didapatkan 1% dari harga total yang dibayarkan, misal Rp100,000 mendapat 1000 poin. Poin dapat digunakan untuk diskon, dengan aturan 1 poin = 1 rupiah diskon dan hanya bisa dipakai pada transaksi berikutnya. Customer hanya dapat menjadi Member dengan mendaftar dengan syarat sudah pernah transaksi.

Terdapat status lain yaitu VIP. VIP adalah pelanggan yang mirip dengan Member, tetapi keistimewaannya lebih baik lagi karena mendapatkan *fixed rate* diskon sebesar 10% dan memiliki sistem poin yang aturannya sama dengan Member. Customer juga bisa menjadi VIP dengan syarat yang sama dengan member.

### 1. Halaman pendaftaran Member

Terdapat menu agar BNMO dapat mendaftarkan Member. Transaksi terakhir yang dibuat oleh customer dengan *id* tertentu akan dipindahkan menjadi transaksi pertama di Member. Halaman pendaftaran Member dan VIP hanya meminta beberapa hal, seperti nama dan nomor telepon untuk dimasukkan ke dalam sistem.

### 2. Halaman *update* Member/VIP

Terdapat juga halaman *update* data member/VIP. BNMO dapat melakukan *update* data member untuk semua *field* selain id. BNMO dapat mengubah status member ke VIP dan sebaliknya.

### 3. Deaktivasi Akun Membership



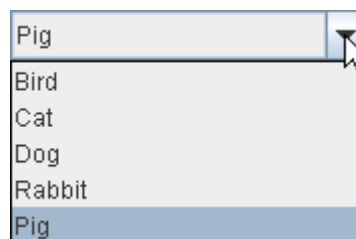
BNMO tidak dapat menghapus membership siapapun namun dapat mendeaktivasi kan suatu membership. Akun membership yang dimatikan berarti akun tersebut tidak bisa memakai poin, tidak mendapat poin pada transaksi berikutnya, poin sebelumnya masih tersimpan dan apabila membershipnya adalah VIP, tidak bisa mendapat 10%. Transaksi histori pada akun yang di deaktivasi tetap berjalan normal.

#### 4. Histori Transaksi

Halaman ini memuat semua histori transaksi dari seorang member/VIP/Customer. Customer pasti hanya ada 1 transaksi karena akan selalu bertambah *id*-nya. Sedangkan, Member atau VIP pasti memiliki 1 transaksi karena mereka harus melakukan transaksi sebelum menjadi member atau VIP.

#### 5. Saat pembayaran

Setiap pembayaran, BNMO memasukkan nama customer, apabila dikosongkan maka akan membuat customer baru dengan *id* bertambah. Apabila nama customer terisi, maka terdapat sebuah hint nama Member/VIP berdasarkan input BNMO. Akun membership yang di deaktivasi akan tetap muncul untuk ditambahkan ke histori transaksinya. Contoh visualisasi adalah sebagai berikut.



Sumber Gambar: [Web MIT](#)

Catatan: Gambar diatas hanya ilustrasi tampilan GUI. Jika menyesuaikan pada spesifikasi, teks pada "Pig" dapat diketik, dan seharusnya Bird, Cat, Dog, Rabbit tidak muncul karena awalan kata-kata tersebut bukan Pig.

## ● Sistem Usaha Barang

Usaha utama BNMO adalah menjual barang. Pada sistem ini terdapat fitur sebagai berikut.

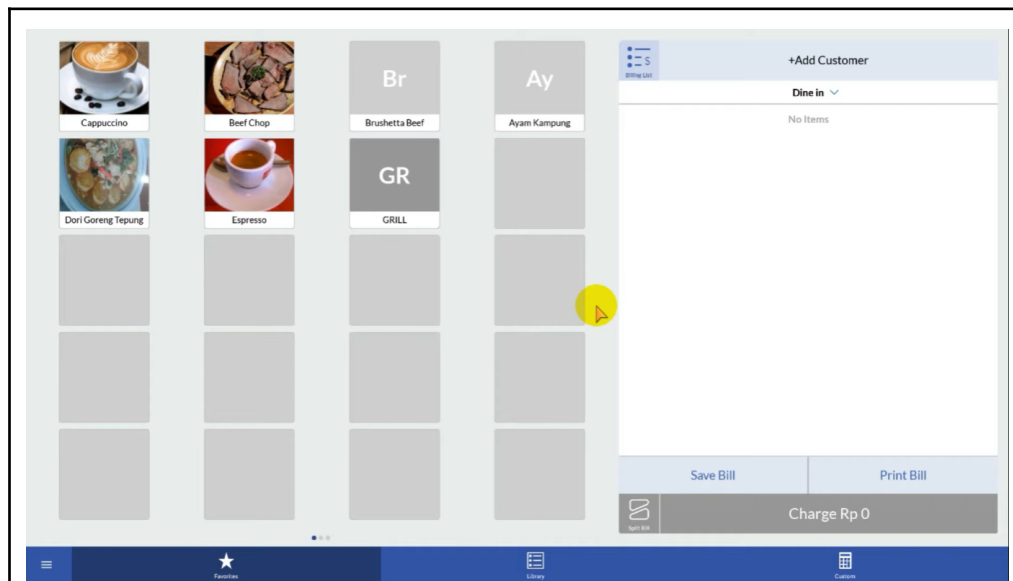
#### 1. Barang & Sistem Inventaris/Gudang

Aplikasi dapat mengatur penambahan, pembacaan dan perubahan barang (atau lebih dikenal dengan operasi CRUD). Untuk gambaran umum, di dalam aplikasi yang kalian kembangkan diperlukan *section*/halaman/tempat dimana pengguna dapat mengatur jenis dan jumlah masing-masing barang yang tersedia, serta

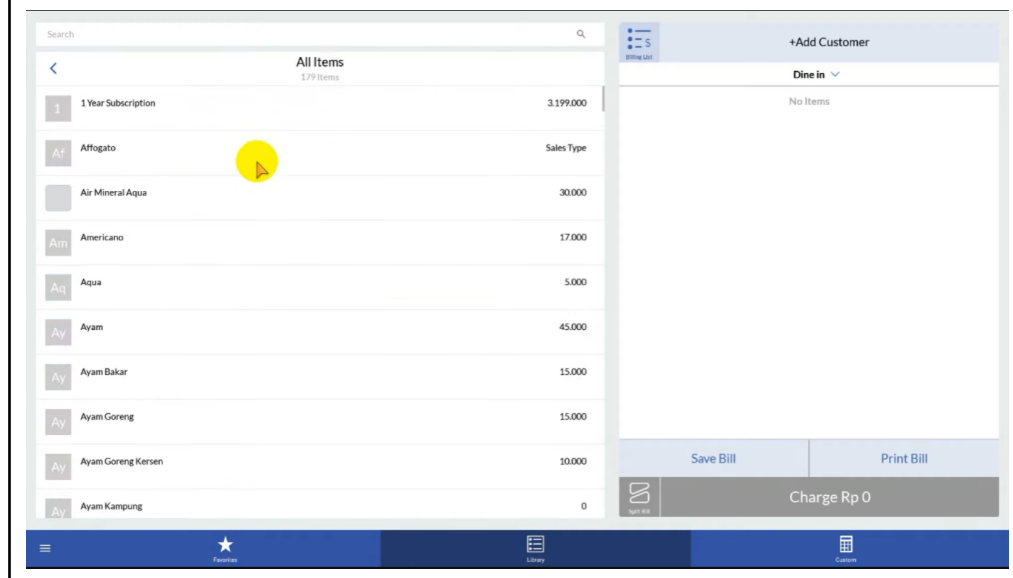
section/halaman/tempat dimana pengguna dapat “menjual” barangnya (kasir). Field yang wajib adalah stok, nama barang, harga barang, harga beli barang, kategori, dan gambar.

Catatan: hati-hati dengan barang yang dihapus atau diupdate! Barang tersebut meskipun dihapus/diubah, pada *bill* yang sudah terbuat tetap harus ada datanya seperti normal saat dibeli. Dihapus berarti tidak bisa dijual/di restock.

Setiap ada pencarian barang, dapat dicari berdasarkan nama, harga dan kategori.



Ilustrasi Kasir. Sumber Gambar: [Moka App](#)





Cancel Affogato - Rp 25.000 Save

QUANTITY

1 - +

DISCOUNTS

member (10%) beli 1 free 1 (100%)

Owner (20%) opening (30%)

Compliment (100%) Dinein (15%)

SALE TYPE | CHOOSE ONE

Dine in Gofood

GrabFood Take Away

You can't select certain sales types because it doesn't have a price. Please refer to Backoffice for more details.

NOTES

Description

Ilustrasi Manajemen Barang. Sumber Gambar: [Moka App](#)

Catatan: Gambar hanya Ilustrasi, sesuaikan lagi dengan Spesifikasi dan Kreativitas kalian.

## 2. Bill

Merupakan entitas yang membantu menyimpan **data sementara** barang apa saja beserta jumlahnya untuk pelanggan yang sedang dilayani. Ibaratnya, entitas ini lah yang merepresentasikan data ketika pengguna berada di halaman kasir. Tetap disimpan pada *data store*, karena kata **sementara** disini diartikan *bill customer* yang masih memungkinkan menambah barang. Namun, apabila aplikasi di restart, data *bill* tetap sama.

## 3. Fixed Bill

Merupakan entitas yang membantu menyimpan **data yang sudah tetap** mengenai barang apa saja beserta jumlahnya untuk pelanggan sedang dilayani. Ibaratnya, ketika pengguna melakukan "checkout" maka dari yang sebelumnya menggunakan entitas Bill, akan berubah menjadi menggunakan entitas Fixed Bill ini. Karena apabila Bill sudah dibayar, maka tidak mungkin menambah barang sehingga *customer* harus membuat Bill baru.

## ● Sistem Laporan

Program mampu melakukan print atau menyimpan data yang dimiliki dalam file pdf yang lebih mudah dibaca dan bisa dipindahkan oleh pengguna. Data yang dapat disimpan menjadi pdf adalah sebagai berikut:

1. Laporan penjualan

Laporan yang menampilkan semua penjualan yang telah diproses oleh program.

2. Fixed Bill

Data final mengenai sebuah transaksi yang dilakukan oleh seorang pelanggan.

Agar pengguna dapat melakukan hal lain selama program melakukan print ke pdf, manfaatkanlah *threading* untuk implementasi fitur ini. Pada *thread* yang melakukan print ke pdf, panggil method `Thread.sleep(10000)` untuk mensimulasikan proses print ke pdf yang membutuhkan waktu lama.

Tidak ada format spesifik untuk file pdf hasil penyimpanan/format dibebaskan.

## ● Halaman Setting

Terdapat halaman setting yang menyediakan opsi untuk mengubah tempat penyimpanan file data serta format. Layout halaman dibebaskan. Jika data file disimpan lebih dari 1 file maka basis penyimpanan adalah memilih folder.

Untuk menghapus/mencopot plugin (akan dijelaskan di spesifikasi bagian plugin), bisa dilakukan pada halaman Settings.

Jika ada hal-hal lain yang bisa dikonfigurasi atau disetting, dapat diletakkan pada halaman ini.

## ● Data Store

Data store merupakan tempat dimana kalian akan menyimpan data-data customer, produk, dan data-data lain yang diperlukan dalam aplikasi ini. Penyimpanan data dapat memanfaatkan *extension file* JSON atau XML atau OBJ. Penyimpanan data store harap dipisah untuk setiap konteks, seperti customer dan produk dijadikan dalam file data store yang berbeda, dst.

Pengerjaan data store ini dilakukan dengan memanfaatkan [adapter pattern](#). Penggunaan library untuk parsing JSON dan XML diizinkan, namun kalian tidak boleh digunakan secara mentah-mentah di program utama, melainkan kalian harus menggunakan kelas sendiri yang memanfaatkan library tersebut sebagai komponen pendukung (parser) yang mengikuti *design pattern* adapter kalian.

Format file JSON atau XML tidak boleh berisikan array bytes (serialize object jadi bytes menggunakan Java API lalu diletakkan pada JSON/XML), melainkan harus diserialisasi dengan nilai asli (String, integer, objek (nested json/xml)) untuk alasan kompatibilitas, editable, dan mudah dibaca dengan mata telanjang.

Untuk format OBJ, gunakan Java API Serializable. Ada beberapa referensi penting yang bisa dimanfaatkan:

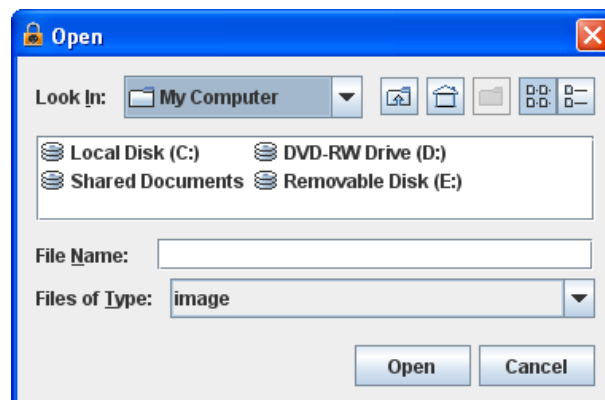
- [Java Serialization](#)
- [Transient Keyword, jika diperlukan.](#)

Tugas kalian: membuat data store yang bisa diganti-ganti antara JSON, XML maupun OBJ. Penggantian format bisa dilakukan melalui Settings.

## ● Plugin

Plugin adalah perangkat lunak yang menambahkan fitur tertentu pada suatu program tanpa harus mengubah program itu sendiri. Biasanya, plugin digunakan agar pengguna dapat memilih komposisi fitur pada program yang tersedia. Sebagai contoh, *wordpress* adalah salah satu perangkat lunak berbasis website yang menggunakan sistem plugin, sehingga pengguna dapat memilih plugin yang dibutuhkan dan memungkinkan *developer* untuk membuat plugin baru tanpa mengubah *wordpress* itu sendiri.

Tugas anda adalah membuat sistem plugin pada tugas besar ini. Secara umum, terdapat sebuah menu plugin yang dapat membuka sebuah panel baru untuk membaca file. Tampilan memilih file adalah sebagai berikut.



Sumber Gambar: [Web MIT](#)

File yang bisa dibaca hanyalah *file .jar* yang spesifikasinya dijelaskan pada bagian berikutnya. Untuk menjalankan JAR yang dibaca secara dinamis, dapat memanfaatkan Java Reflection dan Java Class Loader. Berikut adalah bahan bacaan yang mungkin dapat membantu anda untuk mengerjakan bagian ini:

1. [Java Reflection: Dynamic Class Loading](#)
2. [Load JAR File Dynamically](#)
3. [Class Loader: How it works](#)
4. [Java Class Loader](#)
5. [Building Simple Class Loader \(Part 1\)](#)
6. [Building Simple Class Loader \(Part 2\)](#)

Apabila program di restart, program harus bisa membaca plugin yang sebelumnya di load. Untuk menghapus/mencopot plugin, bisa dilakukan pada halaman Settings. Boleh diasumsikan, JAR plugin tidak pernah dipindahkan atau dihapus setelah di load.

## ● Komponen Plugin

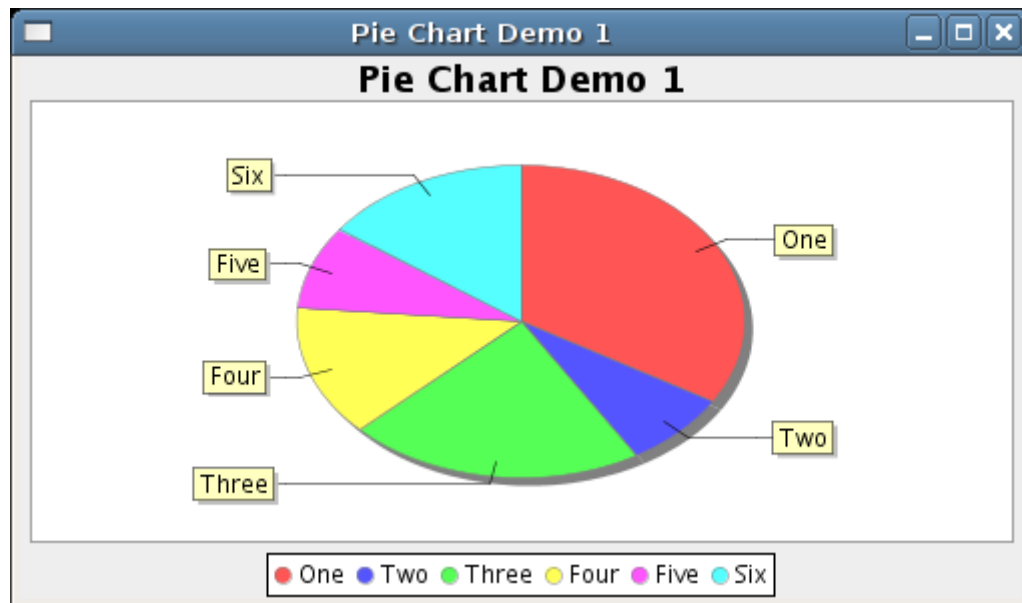
BNMO ingin melihat data-data tokonya dan menarik beberapa kesimpulan. Tetapi sulit sekali untuk melihat data tersebut dalam bentuk tabel atau *object*. Selain itu, BNMO juga di beberapa cabang toko lain, memiliki kesulitan karena *policy* yang sedikit berbeda dengan toko utama.

Buatlah plugin untuk menunjukkan sebuah *insight* mengenai data melalui visualisasi. *Insight* yang ditampilkan dibebaskan namun perlu ada visualisasi data melalui *chart* seperti *pie chart*, *bar chart*, dan grafik lain yang sejenis. Akan lebih lanjut dibahas pada paragraf selanjutnya.

Saat plugin berhasil terbaca oleh program utama, maka terdapat menu baru dan apabila dibuka akan ada halaman khusus untuk Sistem Analitik ini.

Visualisasi data yang dibuat perlu diperbaharui untuk mendapatkan terbaru. Silakan pakai *threading* untuk memperbaharui data secara *real time*, setiap X detik. Untuk menghasilkan *chart*, dapat menggunakan JFreeChart yang dapat dibaca di tautan berikut. Diperbolehkan juga menggunakan *library* lain untuk membuat *chart*.

1. <https://www.jfree.org/jfreechart/samples.html>.
2. [Tutorial lain menggunakan maven](#).
3. [Tutorial untuk menginstall pada maven](#).



Sumber Gambar: [JFreeChart](#)

Silakan buat sebuah *base* plugin yang kalian akan extend menjadi dua buah plugin.

#### 1. Base plugin

Base plugin ini berisikan utilitas yang dapat dipakai oleh plugin 1 dan 2 beserta sebuah class untuk membuat halaman kosong pada sistem utama. Secara umum, pengguna tidak dapat meload *base plugin* ini karena hanya berisi API/Utilitas untuk plugin 1 dan plugin 2.

#### 2. Plugin Chart 1

Plugin 1 berisikan beberapa *insight* data menggunakan Line chart **dan** Bar chart pada 2 kotak yang berbeda atau tidak digabung. Jika plugin ini *diload* dan pengguna belum pernah me-*load* plugin chart lain, maka program perlu menginisiasi halaman kosong yang dibuat pada *base plugin*. Lalu memasukkan Line chart **dan** Bar chart pada halaman kosong tersebut. Diharapkan untuk memanfaatkan utilitas dari *base plugin*.

#### 3. Plugin Chart 2

Sama seperti Plugin 1, namun berisi Pie chart. Skema *loading* seperti inisiasi halaman kosong jika pengguna belum pernah melakukan *loading* plugin chart 1 juga ada. Diharapkan untuk memanfaatkan utilitas dari *base plugin*.

#### 4. Plugin Sistem

Tidak menginisiasi halaman kosong untuk memasukkan chart karena plugin ini tidak berisi *chart*. Plugin ini berkaitan dengan sistem, kelompok anda boleh harus menyediakan dua buah plugin:

1. Mengubah mata uang melalui *settings* (*default* memakai IDR).

Untuk penyesuaian harga bisa langsung mengupdate harga/receipt yang sudah ada dari mata uang sekarang ke mata uang yang diganti menggunakan rumus perbandingan. Untuk kurs, silakan tentukan sendiri, tidak perlu akurat. Minimal terdapat 3 mata uang. Mata uang dan kurs-nya disimpan pada data store, **tidak boleh di hardcode**.

Rekomendasi referensi untuk *design pattern* pada program utama: [Decorator Pattern](#) dan [Composite Pattern](#). Referensi ini masih abstrak, karena diperlukan untuk mengerti kedua *pattern* tersebut.

2. Menambah sistem *discount*, *tax*, dan *service charge* pada menu pembayaran.

Harus disimpan pada data store. Besaran tax dan service charge *flat percent*/tetap untuk setiap bill dan dapat diubah melalui settings. Sedangkan discount tidak *flat percent* dan dapat disesuaikan setiap bill.

Perlu diperhatikan, program utama **tidak boleh** menyediakan kolom/field data sebelum plugin ini di/load. Harus memiliki paradigma:

1. Pembuat program utama hanya menyediakan API/alat untuk *developer* lain meng-extend dan membuat plugin di masa yang akan datang. Prinsip *open closed* pada SOLID.
2. Pembuat program utama diasumsikan tidak tahu *developer* lain ingin membuat 4 buah plugin ini sehingga *developer* program utama tidak menyediakan implementasi konkrit untuk plugin chart 1, chart 2 maupun plugin sistem 1 dan plugin sistem 2. Prinsip *open closed* pada SOLID.
3. Pembuat program utama hanya memberikan ruang untuk menambah dan meng-extend program utama. Prinsip *open closed* pada SOLID.
4. Pembuat *plugin* pure memakai API Utilitas atau *interface* yang disediakan oleh *developer program utama*.

Catatan lain: deliverables dari spesifikasi ini adalah, 4 buah jar yang berbeda-beda. Plugin Chart 1, Plugin Chart 2, Plugin Sistem 1 dan Plugin Sistem 2.



# Bonus & Eksplorasi

*Kenapa dinamakan bonus? Untuk menambahkan pengetahuan dan eksplorasi.*

## 1. Unit Testing

Unit Testing mencakup tes yang terisolasi untuk salah satu modul/kelas tertentu untuk menjamin fungsionalitas dasar sudah berjalan benar. Manfaat terbesar adalah dengan adanya unit testing dengan coverage yang cukup, Anda dapat melakukan modifikasi kode/refactoring secara lebih praktis (tidak perlu uji coba fungsionalitas manual) dan aman (sudah terjamin benar).

Deliverables Bonus Unit Testing:

1. Implementasikan unit testing untuk **semua kelas** yang Anda buat.
2. Tunjukkan coverage unit testing untuk setiap kelas yang ada unit test diatas 60%, letakkan screenshot di laporan. Gunakan [JaCoCo](#) atau *coverage tools* lainnya.

*60% coverage sebenarnya masih rendah, tetapi, yang penting you got the point.*

## 2. Lombok dan Null Safety dengan Annotation

[Lombok](#) adalah sebuah *library open-source* yang ditulis dalam bahasa Java, yang berfungsi untuk mengurangi boilerplate code (kode yang berulang-ulang) dalam pembuatan program. Lombok menyediakan sebuah set annotation yang dapat digunakan untuk secara otomatis menghasilkan getter, setter, constructor, equals, hashCode, dan lain-lain, sehingga membuat kode program menjadi lebih bersih dan mudah dibaca. Lombok juga mendukung beberapa fitur lainnya seperti logging, builder pattern, dan lain-lain, yang dapat membantu meningkatkan produktivitas dan kualitas kode dalam pengembangan aplikasi Java. [Referensi untuk bacaan lombok.](#)

Selain masalah kode *boilerplate* yang diselesaikan Lombok, masalah yang umum pada bahasa Java adalah *NullPointerException* hal ini biasa disebabkan oleh sebuah fungsi yang mengembalikan object *null*, memasukkan object *null* dan sebagainya. Untuk memberikan informasi lebih lanjut kepada programmer yang memakai fungsi tersebut dan IDE yang dipakai, gunakan notasi **@NotNull** dan **@Nullable** untuk menjaga keamanan dari object tersebut. Terdapat dua package yang umum digunakan, diantaranya:

1. *JavaX Annotation* (*javax.annotation*)
2. [Jetbrains Annotations](#). library anotasi yang disediakan oleh JetBrains, pengembang dari IntelliJ IDEA dan Kotlin.

Tugas Anda pada bonus ini:

1. Integrasikan lombok dengan semua kelas yang dibuat untuk mengurangi *boilerplate*.
2. Letakkan *null safety* di setiap kode program yang kalian buat dengan JetBrains Annotations/JavaX Annotation.

*Dengan mengerjakan bonus ini, pada beberapa konteks, kalian secara tidak sadar memakai design pattern decorator.*

*Rekomendasi dari viel karena make lombok bikin cepet selesai, hehe, gak capek bikin constructor, getter, setter, equals, to string and etc. Dan null safety menghindari banyak error NPE pas lagi development, kalau IDE kalian benar settingnya, bakal langsung merah/dikasih tau possible NPE, jadi ga bete error pas jalanin program kena NPE dimana-mana.*

#### Catatan:


Semua poin bonus yang sekarang harapannya menambah ruang eksplorasi kalian, terutama tertuju kepada mahasiswa yang kemarin memberi *feedback* tugas besar 1 membuat game kurang berguna dan tidak relate dengan pembuatan aplikasi/game pada umumnya.

Dan selanjutnya, ada hadiah untuk mahasiswa yang masih merasa mudah dan selesai sangat cepat. **Bonus nomor 3, 4 dan 5 di bawah ini disarankan dikerjakan** apabila kalian punya waktu lebih untuk eksplorasi. Utamakan spesifikasi wajib dan tugas besar mata kuliah lain jika belum selesai. Tim asisten berharap tugas besar ini dapat bermanfaat dalam penggunaan sehari-hari terutama **konsep-konsep** dalam membuat aplikasi java pada umumnya.

*Kalau ada waktu boleh coba, kalau tidak ada lewati saja dan jangan dipaksa. Overall masih make sense untuk dikerjakan, tetapi kalian perlu menyediakan waktu lebih untuk mencoba-coba dan membaca. Menambah ilmu di dunia **software engineering**.*

**Bonus Nomor 3, 4, dan 5:**  Spesifikasi EXTENDED Tugas Besar 2

## Panduan Pengerjaan

Tugas Besar ini dikerjakan secara berkelompok, kelompoknya dapat dilihat di sheet  IF2210 OOP - Kelompok . Kelompoknya sama dengan kelompok yang kalian buat saat tugas besar pertama. Yang harus kalian lakukan adalah **memilih asisten, memasukkan nama dan kode kelompok** maksimal **Selasa, 11 April 2023 18:00 WIB**, jika tidak memilih maka kelompok tersebut tidak akan dinilai.

Adapun ketentuan seperti berikut.


1. JAR File Wajib bisa dijalankan di sistem operasi **LINUX**.
  - a. Akan ada **pengurangan nilai** jika terjadi anomali.
  - b. Berhak tidak dinilai jika program **tidak jalan total** atau gagal mensimulasikan **mayoritas kondisi** untuk *test case* yang dibuat asisten.

- c. Windows Subsystem Linux atau **WSL** diperbolehkan untuk substitusi sistem operasi **LINUX**.
- d. Dihimbau untuk setiap kelompok mengerjakan di **WSL** atau **Linux** daripada diakhir saat demo tidak jalan sama sekali/ada kasus unik.  
*Kemarin ketika tugas besar Algoritma Struktur Data banyak sekali anomali yang menyusahkan asisten saat demo, jadi lebih baik semua anggota mengerjakan di WSL/Linux dari awal.*
2. Diperbolehkan memakai versi java **8** atau **diatasnya** (11, 17, 19, dsb.)
  - a. Penggunaan syntax yang khusus di versi java tertentu diharapkan dibatasi penggunaannya karena mungkin untuk tidak *backward compatibility* dengan versi java di bawahnya yang menyebabkan hanya bisa dijalankan pada versi tertentu.
  - b. Meskipun kalian menggunakan compiler JDK diatas 8, target *compilation* java kalian **harus ditargetkan** ke Java 8 atau 1.8 agar bisa aman dijalankan pada java 8.
    - i. Gradle: [Configure Java Version](#).
    - ii. Maven: [How to tell maven to use Java 8](#).
3. Penggunaan *library* dibebaskan selama program **pure Java** tidak dicampur dengan HTML, CSS, maupun JS dan bahasa lainnya.
  - a. Selain *library* GUI, penggunaannya tidak boleh mentah-mentah dipakai. Harus kalian bungkus pada suatu kelas dan hanya menjadi komponen pendukung kelas yang kalian buat.
  - b. Menjelaskan poin a, Contoh yang boleh: JSON parser atau XML parser karena menjadi komponen pendukung untuk menulis/membaca data ke/dari format json/xml. Yang berarti, kalian masih membuat kelas untuk menghubungkan dari library JSON/XML ke program kalian.
4. Aplikasi GUI harus desktop app **non web based** tidak boleh diakali seperti menggunakan *webview* dari Java untuk menampilkan tampilan HTML & CSS.
5. Tidak boleh plagiat dari internet maupun kelompok lain.
6. Jangan asal spesifikasi selesai atau program jalan. Nilai bisa hancur meskipun selesai namun desain jelek dan tidak memaksimalkan OOP. Terlebih lagi, nilai individu bisa jelek kalau tidak paham apa-apa mengenai hal yang dikerjakan (bantuan *copilot*, *chatgpt*, dan teman diperbolehkan selama tidak berlebihan dan masih paham).

## Panduan Laporan

Sebagai programmer yang baik, Anda dilatih tidak hanya untuk membuat kode, tetapi juga merancang sehingga program kalian *maintainable*.

Dokumentasikan desain program Anda melalui laporan singkat dengan format **pdf** yang **wajib** berisi semua poin yang ada di **template** atau **format laporan**.

Format Laporan:  IF2210\_TB2\_Laporan\_XXX

## Asistensi & QnA

Format Dokumen Asistensi: [W IF2210\\_TB2\\_Asistensi\\_XXX.docx](#)

1. Asistensi sinkron wajib dilakukan minimal **satu kali**. Metode asistensi sinkron dibebaskan kepada kesepakatan asisten dan kelompok.
2. Dokumen Asistensi di gabung dan menjadi lampiran pada Laporan.
3. Asistensi digunakan untuk bertanya mengenai **pengerjaan dan desain program** yang sekiranya belum terbayang dan belum jelas.
4. Untuk bertanya mengenai **spesifikasi dan general**, ditanyakan melalui QnA: [QnA IF2210 2022/2023](#)
5. Silakan melakukan pengisian pemilihan asisten melalui sheet berikut [IF2210 OOP - Kelompok](#) (Sheet **Pemilihan Asisten**) maksimal 11 April 2023 18:00 WIB.
6. Deadline asistensi adalah 23:59 WIB 2 Mei 2023. **Apabila terlalu mendadak, asisten berhak menolak pengajuan asistensi.** Silakan diskusikan waktu dan metode (online/offline) bersama asisten.

## Pengumpulan

1. Softcopy laporan dan source code program dikumpulkan pada [LINK FORM](#). Submission **ditutup hari Rabu, 3 Mei 2023 22:10 WIB**. Di luar waktu pengumpulan tersebut, deliverables tidak akan diterima.
2. Untuk penyusunan Laporan Tugas Besar, akan disediakan template laporan. Laporan dikumpulkan dalam format pdf.
3. Spesifikasi pengumpulan yaitu:
  - a. **1 kelompok 1x pengumpulan saja!** Jika ada revisi, silakan submit ulang, akan dipakai submisi paling terakhir.
  - b. File source code (semua modul dan program utama) dikompres ke dalam zip.
  - c. File executable (dalam bentuk .jar) yang bisa dijalankan langsung kecuali jar plugin.
    - i. Sehingga terdapat 5 buah file JAR: Utama, Plugin Chart 1, Plugin Chart 2, Plugin Sistem 1 dan Plugin Sistem 2.
    - ii. Pastikan semua JAR dapat dijalankan.
  - d. Hapus semua file *class* dari zip pengumpulan.
  - e. File diberi nama "IF2210\_TB2\_XXX.zip" dengan penjelasan:
    - i. XXX: Kode Unik Kelompok (ditulis dalam 3 huruf) yang ada pada Sheet Kelompok
    - ii. Format untuk nama PDF Laporan sama seperti nama zip, ekstensinya saja yang diubah menjadi .pdf
  - f. Contoh : "IF2210\_TB2\_XYZ.zip" adalah file untuk kelompok XYZ.

## Penilaian

Penilaian Tugas Besar 2 akan dilakukan oleh asisten sesuai kriteria yang tertera pada dokumen Spesifikasi Tugas Besar 2.

Pastikan sebelum pengumpulan, submisi yang kalian kumpulkan sudah benar. Asisten berhak untuk tidak menilai apabila program tidak jalan meskipun kesalahan pengumpulan sesuai prosedur. **Pastikan juga program sudah memenuhi ketentuan yang sudah dituliskan diatas.**

Asisten mengharapkan kejujuran dari setiap mahasiswa, segala bentuk kecurangan dalam pengerjaan tugas besar baik kode maupun dokumen akan mendapatkan nilai E pada mata kuliah IF2210 - Pemrograman Berorientasi Objek.