
# Corak Reka Bentuk dalam Java + SOLID: Membina Kod yang Teguh dan Boleh Diselenggara

**Bruno Carvalho de Aquino**
Diterbitkan 25 Okt 2024





Dalam pembangunan Java, kita sering menghadapi masalah kompleks yang memerlukan reka bentuk yang bijaksana. **Corak Reka Bentuk** Dan

**Prinsip SOLID** ialah strategi utama untuk menulis kod iaitu **Boleh berskala, boleh diselenggara dan elegan**. Mari selami cara menggabungkan kedua-dua konsep ini boleh mencipta perisian teguh yang tahan perubahan! 🚀

### Mengapa Menggunakan Corak Reka Bentuk?

Corak Reka Bentuk menyediakan **penyelesaian boleh guna semula** kepada masalah biasa. Ia bertindak sebagai pelan tindakan, membawa struktur kepada kod dan memastikan ia konsisten, modular dan mudah diselenggara. Corak seperti **Kilang, Singleton, dan Pemerhati** ialah alat yang tidak ternilai untuk mana-mana pembangun Java.

### Bagaimana Prinsip SOLID Melengkapkan Corak Reka Bentuk

Yang **Prinsip SOLID** (Tanggungjawab Tunggal, Terbuka/Tertutup, Penggantian Liskov, Pengasingan Antara Muka, Penyongsangan Kebergantungan) menyediakan garis panduan untuk reka bentuk berorientasikan objek. Mereka membantu menggunakan Corak Reka Bentuk dengan berkesan, memastikan kod boleh disesuaikan untuk keperluan masa hadapan.

### 1. Singleton + SRP (Prinsip Tanggungjawab Tunggal)

Yang **Singleton** Corak memastikan hanya satu contoh kelas wujud, berguna untuk mengurus konfigurasi. Tetapi untuk mengikuti **SRP**, pastikan Singleton mempunyai **tanggungjawab tunggal dan jelas**.

> ***Pro Tip**: Avoid the "God Singleton." Keep it focused on one purpose, like configuration management, and delegate other responsibilities.*

### 2. Kaedah Kilang + OCP (Prinsip Terbuka/Tertutup)

Yang **Kaedah Kilang** memudahkan penciptaan objek tanpa mendedahkan butiran instantiation. Ini sejajar dengan **OCP**, membuat kod anda **Terbuka untuk sambungan, ditutup untuk pengubahsuaian**.

**Sebagai contoh**: Perkhidmatan Pembayaran boleh menyokong CreditCardPayment dan PayPalPayment. Menggunakan Kilang, anda boleh menambah CryptoPayment tanpa menukar kod sedia ada.

**3. Pemerhati + LSP (Prinsip Penggantian Liskov)**

Yang **Pemerhati** Corak berfungsi dengan baik untuk pengendalian acara. Menurut **LSP**, subkelas harus menggantikan kelas induk mereka tanpa mengubah tingkah laku.

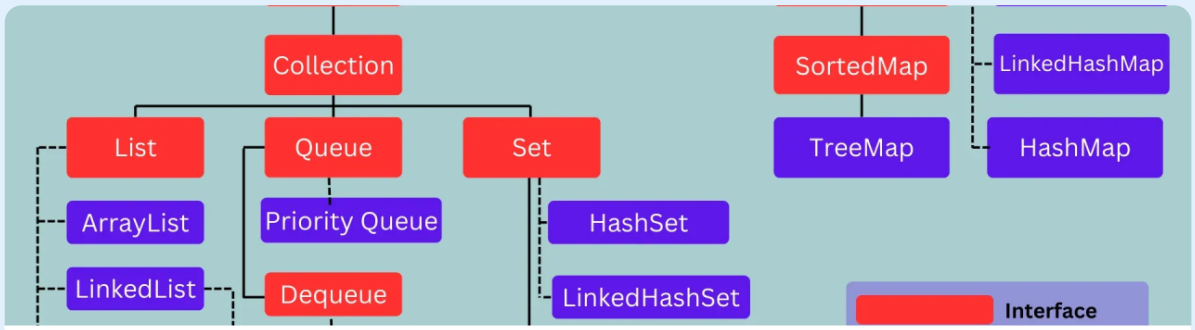
**Sebagai contoh**: Kelas WeatherData boleh memberitahu pemerhati seperti CurrentConditionsDisplay. Pemerhati baharu boleh ditambah tanpa mengubah WeatherData.


**Corak Reka Bentuk Pemerhati**

Senuka Bandara · 10 bulan yang lalu


**Memahami Refleksi dalam Java**

Saad Aslam · 1 tahun yang lalu




**Amalan Terbaik untuk Koleksi Java dan Generik:...**

Omar Ismail · 10 bulan yang lalu

### 4. Pembina + ISP (Prinsip Pengasingan Antara Muka)

Yang **Pembina** Corak membantu membina objek kompleks langkah demi langkah. Berikut **ISP**, gunakan antara muka yang lebih kecil untuk memastikan kod anda modular.

**Sebagai contoh**: VehicleBuilder boleh melaksanakan antara muka EngineBuilder dan WheelsBuilder secara berasingan, memastikan setiap antara muka tertumpu pada tugas tertentu.

**5. Strategi + DIP (Prinsip Penyongsangan Kebergantungan)**

Yang **Strategi** corak membolehkan penconisan algoritma secara dinamik, melengkapkan **BERENANG**. Bergantung kepada abstraksi, bukan pelaksanaan konkrit.

**Sebagai contoh**: DiscountCalculator harus bergantung pada antara muka DiscountStrategy, dengan pelaksanaan seperti HolidayDiscount dan MemberDiscount. Strategi baharu boleh ditambah tanpa mengubah suai logik teras.

**Mengapa Gabungan Ini Berfungsi**

Menggabungkan Corak Reka Bentuk dengan hasil SOLID dalam kod iaitu:

- **Modular**: Komponen yang ditakrifkan dengan jelas menjadikan kod lebih mudah diuruskan.
- **Boleh diperluaskan**: Ciri baharu boleh ditambah dengan kesan yang minimum.
- **Boleh diselenggara**: Pemisahan kebimbangan yang betul memudahkan penyahpepijatan.
- **Boleh diuji**: Komponen yang dipisahkan menjadikan ujian mudah.

**Fikiran Akhir: Melabur dalam Kualiti Kod**

Menggunakan Corak Reka Bentuk dan SOLID bukan sahaja tentang mengikuti amalan terbaik—ia tentang melabur dalam kualiti kod untuk jangka panjang.

**Bagaimana dengan awak?**

➡️ **Corak Reka Bentuk dan prinsip PEPEJAL manakah yang paling kerap anda gunakan?** Kongsi pengalaman anda dalam komen! Mari kita bincangkan apa yang berkesan dan apa yang tidak.

🔥 **Menikmati artikel ini? Ikuti saya untuk mendapatkan lebih banyak pandangan tentang Java, Corak Reka Bentuk dan seni bina perisian!** 🔥

#Java #Corak Reka Bentuk #PEPEJAL #Kejuruteraan Perisian #Kod Bersih #Pengaturcaraan #Senibina Perisian #JavaDevelopment #Amalan Terbaik #Komuniti Dev

**Hugo Ferreira da Silva** 1thn

Achieving good code quality that is easy to maintain even after several months (or years) is essential! Great article!

Suka · Balas | 1 Reaksi

**Ezequiel Cardoso** 1thn

Nice post! Thanks for sharing

Suka · Balas | 1 Reaksi

**Jader Lima** 1thn

Thanks for sharing!

Suka · Balas | 1 Reaksi

**Alexandre Germano Souza de Andrade** 1thn

Useful tips, Thanks for sharing!

Suka · Balas | 1 Reaksi

Lihat lagi komen

Untuk melihat atau menambahkan komen, **daftar masuk**