Design Pattern

Aliaa Ali

سؤال انترفيو مهم هيقولك قول ديزاين باترن انت عارفه وبيحل مشكله ايه؟؟

طيب ايه هو الديزاين باترن؟

دا **عبارة عن طريقه لكتابه الكود مشتركه بين المبرمجين لحل مشاكل متكرره**

طب هما عملوه ليه يعنى وفايدته ايه ؟🤔

عشان المشاكل اللى بتتكرر كتير فا هو يعتبر احسن طريقة لحل مشكلة معينة بتواجه المبرمج فا بقوا يخترعولها ديزاين باترن

Reduce Cost :

اقل تكلفة و جهد ف الكود

Reusable :

بقدر استخدم الكود فى اى مشروع تانى محتاج الكود دا و اعدل عليه براحتي

Scalability:

اقدر ازود Features براحتى انقص براحتى

Predictability:

تحكم ف الكود اسهل و Debug اسرع

عندنا 3 انواع من الديزاين باترن

1-Creational type

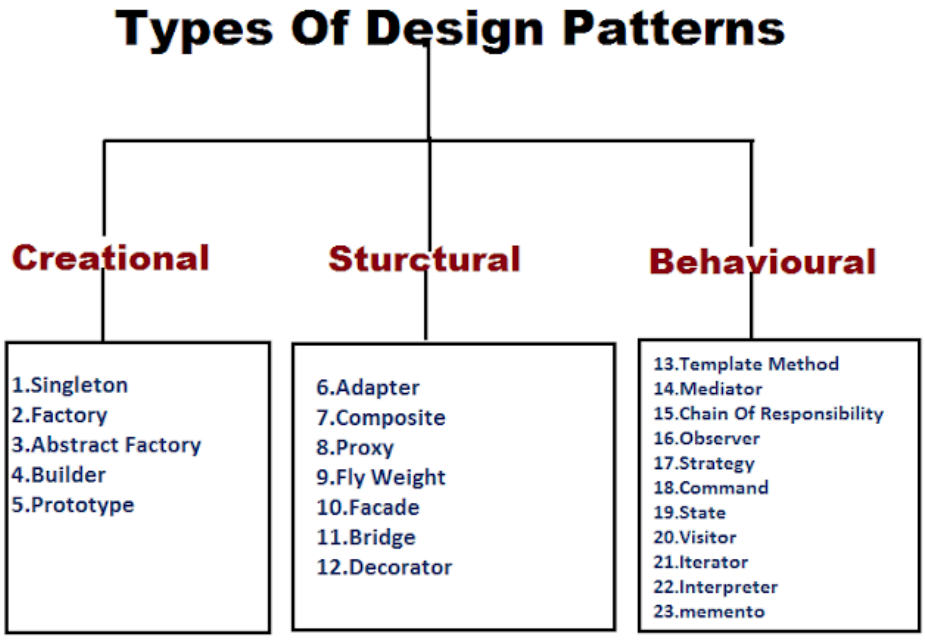
وتأتي فكرتها من إنشاء ال objects

2-Structural type

تتعلق بشكل الclass وكيفيه عمله

3-Behavioral type

وتهتم بالتواصل بين ال objects وبعضها

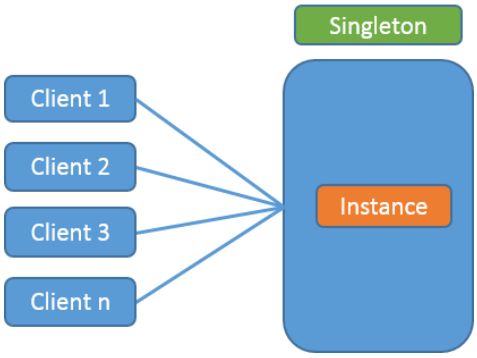


هنتكلم عن واحده من كل نوع فيهم

1- Singleton Pattern

هو نوع من أنواع الـ **Creational** Design Patterns

هو Pattern بيخليك تعمل Instance واحد من الـClass أو الـObject عشان تستخدمه كذا مرة في كذا مكان



طب وبنستخدمه امتى؟! وليه؟!

بتستخدمه لما يكون عندك Class كبير من ناحية الـPerformance والـMemory + بتستخدمه فى كذا مكان فى الـApp بتاعك

ميزته إنه بيـImprove الـPerformance بتاعك + بيقلل استهلاك الـMemory عندك فبالتالي بتطلع بـApp نضيف

في طريقتين للـCreate:

1- الـInitialization Eager: ولكن ليه Cons&Pros: الـPros إنه ThreadSafe لما يكون الـApp بتاعك صغير ومبيحتاجوش Services كتير + جميل بإنك بتكتب سطر Code واحد وسريع والدنيا تمام

الـConsequences : إن لو عندك عدد كبير من الـService بيحتاجوه 👈 فلازم يتعملهم Create أول مالـSystem يقوم وبالتالي بيسبب بطئ فى الـPerformance + بيتعملهم Create بـHashCode مختلف فبياكل مساحة Memory كبيرة

وهنا بنلجأ للحل الأكثر مرونة ألا وهو:

1- الـLazy Initialization: هنا بنخلي الـInstance نايمة + لما حد يـCallـها يتعملها Activation

2- Adapter Pattern

هو نوع من أنواع الـ **Structural** Design Patterns

بيساعدني في تحديد طريقة الـ Structure للـ Object بتاعي حسب الـ Case اللي بتواجهني

عشان الـ Components Testing يكون سهل وبسيط +أقدر أ Extend الـ Code بتاعي من غير مشاكل

ولما أحتاج أزود Features جديدة متسمعش عندي في باقي الـ Modules التانية فمحتاجش إني أعدل على الـBase Code

بستخدمه امتى؟؟

*بنستخدم الـ Adapter لما يكون عندنا Class من Interface معين وعاوز تحوله لـ Interface من نوع تاني , ساعتها بنعمل Adapter Class يحول الـ Interface الأول عشان يناسب الـ Interface التاني.

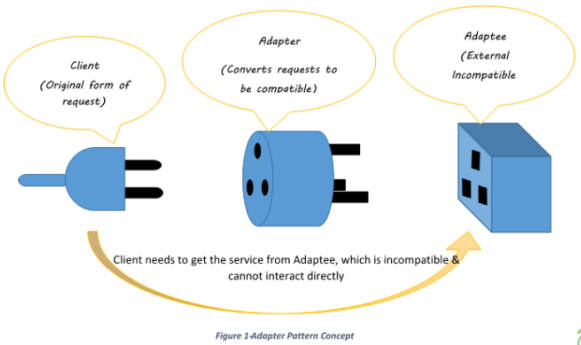


Figure 1-Adapter Pattern Concept

مثال للتوضيح

افترض انك اشتريت Mobile جديد , لكن الشاحن مش بيدخل في الفيشة اللي في الحيطة عندك ؛نظرًا إن فيشة الشاحن ثلاثية وفيشة الحيط ثنائية

فقدامك أكتر من حل:

1. إنك تغير رأس الشاحن بتاع موبايلك = وده مكلف + مش أنسب حل

2- .إنك تغير الأوضة اللي أنت قاعد فيها بمكان فيه فيشة مناسبة = وده مش مريح + حل مش لطيف

3- الحل الأنسب إنك تجيب رأس شاحن (Convertor) تركبها في راس الشاحن بتاع الـ Mobile عشان تركب في أي فيشة عندك في الشقة = وده أنسب وأريح حل.

*فالـ Adapter بيحول الـ Adaptee -الـ Class أو الـ Interface اللي مش مناسب معاك لـ Interface من النوع اللي حضرتك عاوزه

يعني تقدر تقول إنه Mid-Layer بين الـ Code والـ Legacy Class وبيكون عامل زي المترجم بينهم .

3- Strategy Design Pattern

هو نوع من أنواع الـ **Behavioral** Design Patterns

إيه هي الـCase اللي بستعمل فيها الـStrategyDP ؟!

لما بيكون عندك كذا Algorithm بيعملوا نفس الحاجة ولكن بطرق مختلفة + عاوز تغير طريقة تنفيذ الـCode أو تغير الـAlgorithm اللي عاوز تعملها Set معين خلال الـRunTime 🕐

- **ممكن مثال؟! 🤔**

لو عندك لعبة فيها Zombies + كل Level ليه Zombies معينة + كل Zombie ليه مواصفات معينة
👈 في الحالة دي بنعمل Class واحد + ننفذ الـInheritance عليه بإننا بنخلي بقية الـObjects يورثوا منه
✋ لكن فى الـStrategyDP بنعمل ده عن طريق **الـComposition** مش الـInheritance لأسباب كتير

- طب ليه؟! 🤔

- عشان الـStrategyPattern بيطبق مبدأ مهم جدًا ألا وهو الـOpenClosedPrinciple

👈 وده عن طريقه بيخلي الـClass بتاعي Open4Extensions 👈 يعني فيما بعد ممكن الـCode يتعدل من حيث الـBehaviorals و بيكون الـClass بتاعي Closed4Modifications 👈 يعني اللي بيعدل الـCode يغير الـBehaviorals براحته + بدون المساس بالـSteps اللي عاوز أنفذها بالشكل اللى عاوزه بالظبط 👌

- وبالتالي بيديلك ExpressionOfConcerns يعني بيفصل مميزات كل Object أو Class عندك عن التاني 👈 فبيديلك ModifieFlexibility في أي Class لو في أي Strategy تانية عاوز تنفذها 👈 عن طريق إنك تطَّلع الـClass اللي عاوز تضيفه بره الـCode بتاعك + تعمله Composition بره + تعملها Extends جوه 👈 عشان تغير فيه براحتك وتعملها SetBack فى الـContext اللي أنت عاوزها

🤔 طب إيه هى الـCons&Pros للـStrategyDP؟!

1- من حيث الـPros: (ايجابيات)
أ- بيخليك تقدر تغير كذا Algorithm جوه الـObject خلال الـRunTime.
ب- بيخليك تقدر تعزل الـImplementationDetails لـAlgorithm معينة عن الـCode اللى بتستخدمه.
ج- بيطبق مبدأ الـComposition بدل الـInheritance 👈 وده ليه مميزات كتير يا ريت تقرأ عنه 👍
د- بتطبق مبدأ الـOpenClosedPrinciple بإنك تحط Strategy جديدة من غير متغير الـAlgorithm كلها.

2- أما من حيث الـCons: (سلبيات)
أ- لو عندك TwoAlgorithms بيتيغيروا باستمرار (ودي حالة نادرة) متقدرش تضيف Interfaces&Classes جديدة مع نفس الـPattern.

ب- الـUsers لازم يعرفوا الاختيارات المتاحة فى الـStrategy لو في حاجة فيها Offer أو Discount عشان يقدروا يختاروا بنفسهم الـAlgorithm المناسبة ليهم.

ج- فى اللغات الجديدة زي الـPython&Haskell&JS ..إلخ 👈 بيكون فيها FunctionalTypeSupport اللي بتخليك تـImplement كذا Version من Algorithm واحد جوه AnonymousFunction 👈 بعدين تقدر تستخدم الـFunction دي زي الـStrategyObjects بالظبط ،ولكن مبتسمحش إنك تحط فيها Interfaces&Classes تانية (من الآخر استخدامها محدود فى الـStrategyPattern).