Design patterns are time-tested templates for tackling common programming challenges.  
  
Learning them can make you a more efficient, all-rounded developer.  
  
In this post, I'll summarize 15 fundamental patterns that every developer should know :  
  
  
𝗖𝗼𝗺𝗺𝗮𝗻𝗱: Break down complex tasks into bite-sized, reusable steps for cleaner code and easier collaboration. Imagine delegating chores!  
  
𝗖𝗼𝗺𝗽𝗼𝘀𝗶𝘁𝗲: Build flexible object structures like LEGO blocks, allowing you to easily add or remove functionalities. Think nesting folders within folders!  
  
𝗜𝘁𝗲𝗿𝗮𝘁𝗼𝗿: Seamlessly loop through any collection like a well-oiled machine, without touching its internal machinery. Picture traversing a maze without breaking the walls!  
  
𝗗𝗲𝗰𝗼𝗿𝗮𝘁𝗼𝗿: Enhance existing objects with new features on the fly, like adding toppings to a pizza!  
  
𝗣𝗿𝗼𝘅𝘆: Control access to an object like a bouncer at a VIP club, ensuring only authorized users enter.  
  
𝗙𝗮𝗰𝗮𝗱𝗲: Simplified access to complex systems? Think one-stop shop for all your software needs! ️  
  
𝗩𝗶𝘀𝗶𝘁𝗼𝗿: Perform operations on various object types without modifying their code, like a doctor treating different patients. ‍⚕️  
  
𝗔𝗱𝗮𝗽𝘁𝗲𝗿: Bridge the gap between incompatible interfaces, making them play nice like friends at a party!  
  
𝗕𝗿𝗶𝗱𝗴𝗲: Separate an object's abstraction from its implementation, giving you ultimate flexibility to change one without affecting the other. Think changing the engine of a car without affecting the steering wheel!  
  
𝗙𝗹𝘆𝘄𝗲𝗶𝗴𝗵𝘁: Conserve memory by sharing common parts between objects, like using the same tires for multiple cars!  
  
𝗖𝗵𝗮𝗶𝗻 𝗼𝗳 𝗥𝗲𝘀𝗽𝗼𝗻𝘀𝗶𝗯𝗶𝗹𝗶𝘁𝘆: Pass requests along a chain of handlers until one takes action, like relaying a message through a team.  
  
𝗢𝗯𝘀𝗲𝗿𝘃𝗲𝗿: Keep multiple objects informed when one changes, like a news network broadcasting updates simultaneously.  
  
𝗕𝘂𝗶𝗹𝗱𝗲𝗿: Construct complex objects step-by-step with controlled precision, envisioning a building being assembled brick by brick.  
  
𝗦𝘁𝗮𝘁𝗲: Modify an object's behavior based on its internal state, like a light switch changing functions depending on whether it's on or off.  
  
𝗙𝗮𝗰𝘁𝗼𝗿𝘆: Create objects without specifying their exact type, letting the factory decide based on your needs. Imagine a bakery producing different pastries on demand!  
  
𝗥𝗲𝗮𝗱𝘆 𝘁𝗼 𝗰𝗼𝗻𝗾𝘂𝗲𝗿 𝘆𝗼𝘂𝗿 𝗰𝗼𝗱𝗶𝗻𝗴 𝗰𝗵𝗮𝗹𝗹𝗲𝗻𝗴𝗲𝘀 𝘄𝗶𝘁𝗵 𝗰𝗼𝗻𝗳𝗶𝗱𝗲𝗻𝗰𝗲?  
  
Save this post for future reference.  
Share your favorite design patterns in the comments.  
Challenge yourself to learn and implement a new pattern this week.  
  
𝗣.𝗦. Remember, this is just a starting point! Dive deeper into each pattern to unlock their full potential.  
  
Happy coding!  
  
Follow me here - [Brij kishore Pandey](https://www.linkedin.com/in/ACoAAAKDuMsBugjGZwz0pJy43LJ-6bVwc0gm9xQ)

Activate to view larger image,

