1. Imagine that you are creating a furniture shop simulator that represent a family of related products, say: Chair + Sofa + Coffee Table. Several variants of this family. For example, products Chair + Sofa + Coffee Table are available in these variants: IKEA, Victorian Style , Art Deco. You need a way to create individual furniture objects so that they match other objects of the same family. Customers get quite mad when they receive non-matching furniture because an IKEA sofa does not match a Victorian-style chairs. Also, you want to add new products or families of products to the program. Furniture vendors update their catalogs very often, without changing the core code each time it happens
2. Suppose you are decorating your room with two different types of tables; one is made of wood and one of steel. For the wooden type, you need to visit to a carpenter, and for the other type, you may need to go to a metal shop. All of these are table factories. So, based on demand, you decide what kind of factory you need
3. Creating a GUI framework that can create different UI components (e.g., WindowsButton, WindowsCheckbox, MacButton, MacCheckbox) based on the client's platform.
4. Car manufacturing plant, where different factories produce cars of different models (sedan, SUV, sports car) with their own specific parts.
5. To complete an order for a computer, different hardware parts are assembled based on customer preferences. For example, a customer can opt for a 500GB hard disk with an Intel processor, and another customer can choose a 250GB hard disk with an AMD processor.
6. you want to convert one text format to another text format, such as converting from RTF to ASCII.
7. Building a complex object like a Car by specifying the different parts (e.g., engine, wheels, color) step by step.
8. Construction of a house, where a builder class handles the step-by-step construction process, allowing customization of rooms, materials, and finishes.
9. Imagine that you’re creating a logistics management application. The first version of your app can only handle transportation by trucks. After a while, your app becomes pretty popular. Each day you receive dozens of requests from sea transportation companies to incorporate sea logistics into the app. Great news, right? But how about the code? At present, most of your code is coupled to the Truck class. Adding Ships into the app would require making changes to the entire codebase. Moreover, if later you decide to add another type of transportation to the app, you will probably need to make all of these changes again. As a result, you will end up with pretty nasty code, riddled with conditionals that switch the app’s behavior depending on the class of transportation objects.
10. In an application, you may have different database users. For example, one user may use Oracle, and the other may use SQL Server. Whenever you need to insert data into your database, you need to create either a SqlConnection or an OracleConnection and only then can you proceed. If you put the code into if-else (or switch) statements, you need to repeat a lot of code, which isn’t easily maintainable. This is because whenever you need to support a new type of connection, you need to reopen your code and make those modifications.
11. A framework for creating different types of buttons (e.g., WindowsButton, MacButton) based on the client's platform.
12. Pizza restaurant chain, where each branch has its own factory to create pizzas based on the regional preferences and ingredients.
13. Suppose you have a master copy of a valuable document. You need to incorporate some change into it to analyze the effect of the change. In this case, you can make a photocopy of the original document and edit the changes in the photocopied document.
14. Let’s assume that you already have an application that is stable. In the future, you may want to modify the application with some small changes. You must start with a copy of your original application, make the changes, and then analyze further. Surely you do not want to start from scratch to merely make a change; this would cost you time and money
15. Creating new instances of a Document object by cloning an existing Document object rather than creating it from scratch.
16. Creating multiple instances of a document template in a word processing application by duplicating an existing document.
17. A country can have only one official government. Regardless of the personal identities of the individuals who form governments, the title, The Government of X, is a global point of access that identifies the group of people in charge
18. In some software systems, you may decide to maintain only one file system so that you can use it for the centralized management of resources
19. Logger class in a system where only one instance of the logger should exist to maintain a single log file.
20. Database connection pool in a multi-threaded application, where only one pool instance is needed to handle all database connections.
21. When you travel from the US to Europe for the first time, you may get a surprise when trying to charge your laptop. The power plug and sockets standards are different in different countries. That’s why your US plug won’t fit a German socket. The problem can be solved by using a power plug adapter that has the American-style socket and the European-style plug
22. you use an electrical outlet adapter/AC power adapter in international travels. These adapters can act as middlemen so that an electronic device, say a laptop that accepts a U.S. power supply, can be plugged into a European power outlet. Consider another example. Suppose you need to charge your mobile phone. But you see that the electrical outlet is not compatible with your charger. In this case, you may need to use an adapter. Now imagine a situation where you need to plug an application into an adapter (which is X-shaped in this example) to use the intended interface. Without using this adapter, you cannot join the application and the interface.
23. Suppose you have an application that can be broadly classified into two parts: the user interface (UI or front end) and the database (back end). Through the user interface, clients can pass some specific type of data or objects. Your database is compatible with those objects and can store them smoothly. Over a period of time, you may feel that you need to upgrade your software to make your clients happy. So, you may want to allow some other type of object also to pass through the UI. But in this case, the first resistance will come from your database because it cannot store these new types of objects. In such a situation, you can use an adapter that will take care of the conversion of these new objects to a compatible form that your old database can accept.
24. Adapting an existing payment gateway interface to a new payment service API.
25. Using a power adapter to convert the electrical input from one country's standard to another country's standard.
26. In a software product development company, the development team and the marketing team both play crucial roles. The marketing team does a market survey and gathers the customer requirements. The development team implements those requirements in the product to fulfill the customer needs. Any change (say, in the operational strategy) in one team should not have a direct impact on the other team. In this case, you can think of the marketing team as playing the role of the bridge between the clients of the product and the development team of the software organization
27. GUI frameworks separates abstractions from the platform-specific implementation. For example you can separate a window abstraction from a window implementation for Linux or macOS.
28. Separating the shape abstraction from its rendering implementation, allowing different shapes to be rendered differently.
29. Operating system drivers, where the operating system interface (abstraction) is separate from the specific device drivers (implementation).
30. Armies of most countries are structured as hierarchies. An army consists of several divisions; a division is a set of brigades, and a brigade consists of platoons, which can be broken down into squads. Finally, a squad is a small group of real soldiers. Orders are given at the top of the hierarchy and passed down onto each level until every soldier knows what needs to be done
31. Think of an organization that consists of many departments. In general, each of these departments consists of multiple employees (in other words, all these participants are basically employees in the organization). Some employees are grouped together to form a department, and those departments can be further grouped together to build the whole organization
32. Modeling a company hierarchy where both individual employees and departments can be treated as a single unit.
33. File system hierarchy, where files and folders can be treated as a single unit for operations like copying, moving, or deleting.
34. When you’re cold, you wrap yourself in a sweater. If you’re still cold with a sweater, you can wear a jacket on top. If it’s raining, you can put on a raincoat. All of these garments extend your basic behavior but aren’t part of you, and you can easily take off any piece of clothing whenever you don’t need it
35. Suppose in a GUI-based toolkit you want to add some border properties. You could do this with inheritance, but that cannot be treated as an ultimate solution because you do not have absolute control over everything from the beginning. So, this technique is static in nature. Decorators offer a flexible approach. They promote the concept of dynamic choices. For example, you can surround the component in another object. The enclosing object is termed a decorator, and it must conform to the interface of the component that it decorates. It will forward the requests to the component, and it can perform additional operations before or after those requests. In fact, you can add an unlimited number of responsibilities with this concept
36. When need to add to some shape component a border or a shadow functionality
37. Adding additional functionalities (e.g., encryption, compression) to a file stream without modifying the original file stream class.
38. Suppose you are going to host a birthday party with 300 guests. Nowadays you can hire a party organizer and let them know the key information such as the party type, date and time of the party, number of attendees, and so on. The organizer will do the rest for you. You do not need to think about how they will decorate the party room, whether the food will be buffet style, and so on
39. When you call a shop to place a phone order, an operator is your facade to all services and departments of the shop. The operator provides you with a simple voice interface to the ordering system, payment gateways, and various delivery services.
40. Providing a simplified interface for a complex subsystem like a computer system, hiding the complexity of interacting with individual components.
41. Online shopping checkout process, where a simplified interface hides the complexity of handling payment, shipping, and inventory management.
42. Suppose you have a pen. You can use different ink refills to write with different colors. So, the pen without the refill can be considered the flyweight with intrinsic data, and the refills can be considered the extrinsic data in this example
43. Suppose in a computer game you have a large number of participants whose core structures are the same, but their appearances vary (for example, they may have different states, colors, weapons, and so on). Therefore, if you want to store all of these objects with all these variations/states, the memory requirement will be huge. So, instead of storing all these objects, you can design the application in such way that you will create one of these instances, and your client object will maintain all of these variations/states. Consider another example. Suppose a company needs to print business cards for its employees. In this case, what will be the starting point? The business can create a common template where the company logo, address, and so on, is already printed (intrinsic), and later the company will put the particular employee details (extrinsic) on the cards.
44. Sharing common data (e.g., font styles, colors) between multiple text objects in a word processing application to reduce memory usage.
45. Text editor, where a limited number of font styles and colors are reused across multiple instances of text objects to conserve memory.
46. A credit card is a proxy for a bank account, which is a proxy for a bundle of cash. Both implement the same interface: they can be used for making a payment. A consumer feels great because there’s no need to carry loads of cash around. A shop owner is also happy since the income from a transaction gets added electronically to the shop’s bank account without the risk of losing the deposit or getting robbed on the way to the bank
47. An ATM implementation will hold proxy objects for bank information that exists on a remote server. In the real programming world, creating multiple instances of a complex object (a heavy object) is costly in general. So, whenever you can, you should create multiple proxy objects that can point to the original object. This mechanism can also help you to save the computer/system memory
48. Implementing a proxy class that controls access to a sensitive resource (e.g., database connection) by performing authentication and authorization checks.
49. Virtual private network (VPN) service, where a proxy server acts as an intermediary between the user's device and the internet to provide security and privacy.
50. Most software organizations have some customer care representatives who take feedback from customers and forward any issues to the appropriate departments in the organization. However, not all of these departments will start fixing the issue simultaneously. The department that seems to be responsible will take a look at the issue first, and if those employees believe that the issue should be forwarded to another department, they will forward it. You may see a similar scenario when a patient visits a hospital. Doctors from one department can refer the patient to a different department (for further diagnosis) if they think it’s needed
51. Consider a software application (say, a printer) that can send e-mails and faxes. Obviously, any customer can report either fax issues or e-mail issues, so you need to introduce two different types of error handlers: EmailErrorHandler and FaxErrorHandler. EmailErrorHandler will handle e-mail errors only, and it will not be responsible for fax errors. In the same manner, FaxErrorHandler will handle fax errors and will not care about e-mail errors. You can form a chain like this: whenever your application finds an error, it will just raise a ticket and forward the error with a hope that one of those handlers will handle it. Let’s assume that the request first comes to FaxErrorhandler. If this handler agrees that it is a fax issue, it will handle it; otherwise, it will forward the issue to EmailErrorHandler. Note that here the chain is ending with EmailErrorHandler. But if you need to handle another type of issue, say, an authentication issue because of security vulnerabilities, you can make an AuthenticationErrorHandler and put it after EmailErrorHandler. Now if an EmailErrorHandler also cannot fix the issue completely, it will forward the issue to AuthenticationErrorHandler, and the chain will end there. (This is just an example; you are free to place these handlers in any order you’d like.) The bottom line is that the processing chain may end in either of these two scenarios: A handler can process the request completely. You have reached the end of the chain.
52. Implementing a series of email filters where each filter checks for specific conditions and passes the email to the next filter if it doesn't match.
53. Approval workflow in an organization, where multiple levels of management review and approve requests or documents sequentially.
54. After a long walk through the city, you get to a nice restaurant and sit at the table by the window. A friendly waiter approaches you and quickly takes your order, writing it down on a piece of paper. The waiter goes to the kitchen and sticks the order on the wall. After a while, the order gets to the chef, who reads it and cooks the meal accordingly. The cook places the meal on a tray along with the order. The waiter discovers the tray, checks the order to make sure everything is as you wanted it, and brings everything to your table. The paper order serves as a command. It remains in a queue until the chef is ready to serve it. The order contains all the relevant information required to cook the meal. It allows the chef to start cooking right away instead of running around clarifying the order details from you directly
55. in the menu system of an editor or integrated development environment (IDE). For example, to support undos, multiple undos, or similar operations in a software application
56. Implementing an undo/redo functionality where each user action is encapsulated as a command object that can be executed and undone.
57. Remote control for a television, where each button press triggers a specific command to perform actions like changing channels or adjusting volume.
58. You can think of a translator who translates a foreign language. You can also consider music notes as a grammar, where musicians play the role of interpreters.
59. The Java compiler interprets the Java source code into bytecode that is understandable by the Java Virtual Machine. In C#, the source code is converted to MSIL intermediate code, which is interpreted by the CLR. Upon execution, this MSIL is converted to native code (binary executable code) by the JIT compiler
60. Implementing a query language parser that interprets user input and translates it into database queries.
61. Language translation service, where an interpreter converts a sentence or phrase from one language to another.
62. Suppose there are two companies, Company A and Company B. Company A stores its employee records (say each employee’s name, address, salary details, and so on) in a linked list data structure, and Company B stores its employee data in an array. One day the two companies decide to merge. you do need not to write any code from scratch. In a situation like this, you can have a common interface through which you can access the data for both companies. So, you can simply call those methods without rewriting the code.
63. Similarly suppose, in a college, the arts department is using an array data structure to maintain its student records, and the science department is using a linked list data structure to keep its student records. The administrative department does not care about the different data structures. It is simply interested in getting the data from each department and wants to access the data in a uniform way.
64. Implementing a custom iterator for traversing elements of a custom collection class.
65. Iterating over the elements of a collection like an array, list, or database query result set.
66. When an airplane needs to take off, a series of verifications takes place. These kinds of verifications confirm that all components and parts (which are dependent on each other) are in perfect condition. Another example is when the pilots of different airplanes (who are approaching or departing the terminal area) communicate with the airport towers. They do not explicitly communicate with other pilots in different airlines. They simply send their status to the tower only. These towers send signals to confirm who can take off (or land). You must note that these towers do not control the whole flight. They enforce constraints only in the terminal areas.
67. When a client processes a business application, you may need to implement some constraints. For example, suppose you have a form where clients need to supply their user IDs and passwords to access the accounts. In the same form, you may need to supply other mandatory fields such as e-mail ID, communication address, age, and so on. Let’s assume you are applying the constraints as follows. Initially you will check whether the user ID supplied by a user is a valid one. If it is a valid ID, then only the password field will be enabled. After supplying these two fields, you may need to check whether the e-mail ID is provided by the user. Let’s assume further that after providing all this information (a valid user ID, a password, a correctly formatted e-mail ID, and so on), your submit button will be enabled. In other words, the submit button will be enabled only if the user supplies a valid user ID, password, a valid email ID, and other mandatory details. You can also ensure that the user ID is an integer, so if a user by mistake provides any character in that field, the submit button will stay in disabled mode. When a program consists of many classes and the logic is distributed among them, the code becomes harder to read and maintain. In those scenarios, you want to bring new changes into the system’s behavior.
68. Implementing a chat application where users can send messages to each other through a central chat room object.
69. Air traffic control system, where a central mediator coordinates communication between different aircraft and ground control.
70. A turnstile has some rotating arms, which initially are locked. When you are allowed to go through it (say by putting some coins in), the locks will open, and the arms will be allowed to rotate. Once you pass through, the arms will return to a locked state.
71. In a drawing application, you may need to revert to an older state
72. Implementing a text editor that allows users to undo and redo changes by saving and restoring the state of the text document.
73. Text editor with undo/redo functionality, where the state of the document is saved as a memento object before each change to allow for undoing or redoing.
74. If you subscribe to a newspaper or magazine, you no longer need to go to the store to check if the next issue is available. Instead, the publisher sends new issues directly to your mailbox right after publication or even in advance. The publisher maintains a list of subscribers and knows which magazines they’re interested in. Subscribers can leave the list at any time when they wish to stop the publisher sending new magazine issues to them
75. Think about a celebrity who has many followers on social media. Each of these followers wants to get all the latest updates from their favorite celebrity. So, they follow the celebrity until their interest wanes. When they lose interest, they simply do not follow that celebrity. Think of each of these fans or followers as an observer and the celebrity as the subject
76. In the world of computer science, consider a simple UI-based example. This UI is connected to some database. A user can execute some query through that UI, and after searching the database, the result is returned in the UI. you segregate the UI from the database. If a change occurs in the database, the UI should be notified so that it can update its display according to the change. To simplify this scenario, assume that you are the person responsible for maintaining a particular database in your organization. Whenever there is a change made to the database, you want to get a notification so that you can take action if necessary
77. Implementing an event system where multiple objects can subscribe to events and get notified when the events occur.
78. Stock market ticker, where investors subscribe to receive updates about the price changes of specific stocks.
79. The buttons and switches in your smartphone behave differently depending on the current state of the device: When the phone is unlocked, pressing buttons leads to executing various functions. When the phone is locked, pressing any button leads to the unlock screen. When the phone’s charge is low, pressing any button shows the charging screen
80. Consider the scenario of a network connection, say a TCP connection. An object can be in various states; for example, a connection might already be established, a connection might be closed, or the object has already started listening through the connection. When this connection receives a request from other objects, it responds as per its present state
81. The functionalities of a traffic signal or a. For example, you can change the channel if the TV is already in switched-on mode. It will not respond to the channel change requests if it is in switched-off mode
82. Implementing a vending machine that changes its behavior based on its current state (e.g., idle, accepting coins, dispensing products).
83. Ticket booking system, where the behavior of the booking process changes based on the availability of seats or the user's membership status.
84. Imagine that you have to get to the airport. You can catch a bus, order a cab, or get on your bicycle. These are your transportation strategies. You can pick one of the strategies depending on factors such as budget or time constraints
85. In a game where we can have different characters and each character can have multiple weapons to attack but at a time can use only one weapon. The method attack() will have different implementation depends on which weapon is being used.
86. Implementing different sorting algorithms (e.g., bubble sort, merge sort) that can be selected and used interchangeably based on the user's preference.
87. Sorting algorithms in programming, where different strategies (e.g., bubble sort, merge sort) can be chosen based on the size and characteristics of the data.
88. The template method approach can be used in mass housing construction. The architectural plan for building a standard house may contain several extension points that would let a potential owner adjust some details of the resulting house. Each building step, such as laying the foundation, framing, building walls, installing plumbing and wiring for water and electricity, etc., can be slightly changed to make the resulting house a little bit different from others
89. Suppose you have been hired to design an online engineering degree course. You know that, in general, the first semester of the course will be the same for all courses. For subsequent semesters, you need to add new papers or subjects to the application based on the course opted by a student.
90. Implementing a framework for creating different types of reports (e.g., PDF report, HTML report) with a common structure but varying details.
91. Baking a cake, where a base recipe provides a common structure, but specific steps (e.g., flavors, decorations) can be customized.
92. Imagine a seasoned insurance agent who’s eager to get new customers. He can visit every building in a neighborhood, trying to sell insurance to everyone he meets. Depending on the type of organization that occupies the building, he can offer specialized insurance policies: If it’s a residential building, he sells medical insurance. If it’s a bank, he sells theft insurance. If it’s a coffee shop, he sells fire and flood insurance
93. public APIs need to support plug-in operations. Clients can then perform their intended operations on a class (with the visiting class) without modifying the source
94. Implementing a tax calculation system where different tax calculation algorithms can be applied to different types of products without modifying the product classes themselves.
95. Data analytics tool, where different visitor objects can be applied to different data structures to perform specific analysis or calculations.