NoSql, MongoDB - document style, JPA - Java persistence API

OO - inheritence, polymorphism, abstraction, interface... OO interview questions

Datastructure, - array is a random access data struction

Memory Managment: JVM, Garbabge collection, Heap vs Stack

Design Pattern: Gang of Four

Software development process: waterfall, Agile(Scrum)

Distributed Database System

frontend index web need to change for fitting computer

- SQL (Structured Query Language)

* use SQL Syntax retrieve the data from the database
* Database - table - filed
* strict requirements for data store in database tables. There is a schema in SQL said which data can go into a table
* if there are extra info, it can't record into SQL due to table don't have that column

- NoSQL (MangoDB)

* Humongous: can store lots and lots data
* Database - collections (not table but can translate it with table) - documents (dont need to use the same schema, different data structure)
* flexible, no/few relations (may have duplicate data) compare to SQL

Horizontal Scaling: Add more Servers and merge Data into one Database

Vertical Scaling: Improve Server Capacity/Hardware

|  |  |
| --- | --- |
| SQL | NoSQL |
| Data use Schemas | Schema-less |
| Relations | No/few relations |
| Data is distributed across multiple tables | Data is typically merged into a few collections |
| Horizontal scaling is difficult/impossible; Vertical scaling possible | Both horizontal and vertical scaling is possible |
| Limitations for lots of read & write queries per second | Great performance for mass read and write requests |

JPA (Java persistence API)

* collection of classes and methods to persistently store the vast amount of data into a database
* JPA reduced the burden of interaction with the database. It forms a bridge between object models and relational models
* Java standard for mapping Java object to a relational database

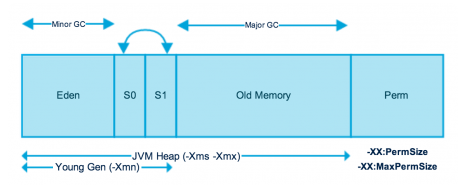
OOP (Object Oriented Programming)

* refer to language use objects in programming
* aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function
* Polymorphism: Overloading (different methods have same name but different signatures) and Overriding (subclass can provide a specific implementation of a method that already provided in their parent class)
* Inheritance: extends (A extends B, A will automatically have all methods inside B)
* Encapsulation: Get and Set method for private variable (Data Hiding, flexibility, easy testing)
* Abstractions : interface and abstract class
* interface class need to have another class implement it, all method inside interface class are interface method does not have body (Java not support multiple inheritance but can achieve multiple interface)
* abstract class need to have another class extend it, can have normal methods or abstract methods (does not have body). Main function cannot create abstract class object. More security.

Data structure:

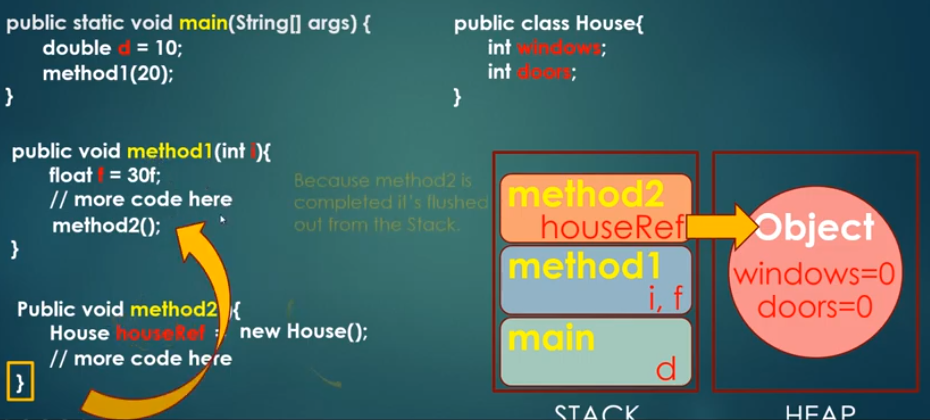
* Array is random access data structure, each element can be accessed directly and in constant time.
* Linked list is a sequential access data structure, each element can be accessed only in particular order.
* Stack and Queues are limited access data stricture
* Stack: Last in First out (push add item into stack, pop remove the last added item out form stack)
* Queues: First in First out (add() insert item into the back of the queue, remove() remove the item least recently added)

JVM Memory Model:

* JVM loads, verifies, executes the code and manage memory
* Memory allocates to the process include the Heap, Meta Space, JIT code cache, thread stacks, and shared libraries
* Heap Memory:
* 
* Young Generation Space: create object - allocated from Eden space - Minor GC is performed - all the survivor obj moves to one of the survivor spaces - Minor GC check survivor obj and move to other survivor spaces (one of the survivor space always empty) - obj that survived after many cycles of GC are moved to old generation memory space
* Old generation memory contains the object that are long lived and survived after many rounds of Minor GC.
* Garbage collection(Major GC) is performed in Old Generation memory when it's full and usually takes longer time
* Permanent generation or Meta space, contains the application metadata required by the JVM to describe the classes and methods used in the application. Perm Gen is not part of Java Heap memory

Heap memory Vs Stack memory:

* Stack: methods, local variables and reference variables
* Heap: objects and instance variables (ie: private int i;)
* Stack memory and Heap memory example:



* after method 2 finish: method 2 remove from stack - object in heap memory is eligible for garbage collection - when JVM run GC it will automatically delete the object in heap memory
* any object created in the heap space has global access and can be referenced from anywhere of the application.
* stack memory is always referenced in Last in First out order.
* We can use - Xms and -Xmx JVM option to define the startup size and maximum size of heap memory. We can use -Xss to define the stack memory size.
* When stack memory full, java runtime throws java.lang.StackOverFlowError
* When heap memory is full, java runtime throws java.lang.OutOfMemoryError: Java Heap Space error

Design Pattern:

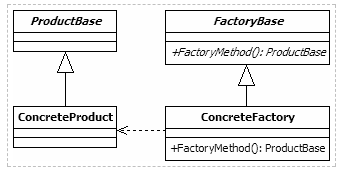
* Gang of Four:  the four authors of the book, "Design Patterns: Elements of Reusable Object-Oriented Software" and include 23 different design patterns
* Design patterns provide solutions to common software design problems
* *http://www.blackwasp.co.uk/gofpatterns.aspx*
* Creational Patterns: Abstract Factory, Builder, Factory Method, Prototype, Singleton
* Structural Patterns: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy
* Behavioral Patterns: Chain of responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor
* *factory，singleton，adapter，decorator，observer*

Software Development Methodology: Agile (Scrum) 嫁接到已有的project

* Iterative (get prototype ASAP - get review from user - updating )
* Streamlined (quick meeting, less documentation, start your job ASAP)
* Time-Boxed (plan the project in time) ie: we will have 3 weeks development and we will try to get future A,B and C. If there are extra time, we can start to development future D
* Very Collaborative (communicate each other all the time)
* Agile (Quick) Vs. Waterfall (Quality)
* Scrum: Daily Standups, Springts, Product Demos, Retrospectives

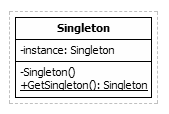
Factory Method Design Pattern

* Creational pattern to control class instantiation
* replace class constructors, abstracting the process of object generation so that the type of the object instantiated can be determined at run-time
* Factory Base: an abstract base class for the concrete factory class that will actually generate new object
* ContrereFactory: inheriting from the FactoryBase class, inherit the actual factory method
* ProductBase: This abstract class is the base class for the type of object that the factory can create. It's also the return type for the factory method
* ConcreteProduct: Multiple subclass of the Product class are defined, each containing specific functionality.



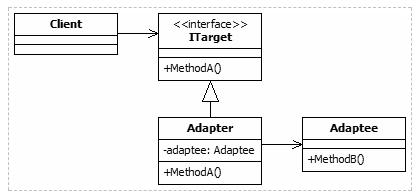
Singleton Design Pattern

* use to ensure that a class can only have one concurrent instance
* useful when a single, global point of access to a limited resource is required. Avoid global variable be copied and duplicated
* whenever additional objects of a singleton class are required, the previously created , single instance is provided



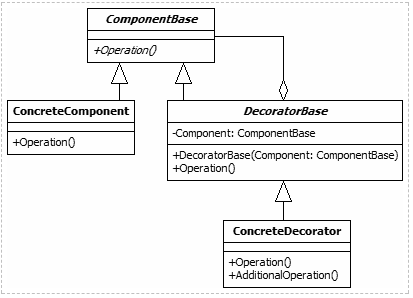
Adapter Design Pattern

* structural pattern that defined a manner for creating relationships between classes
* used to allow two incompatible types to communicate
* one class relies upon a specific interface that is not implemented by another class, the adapter act as a translator between the two types
* Client: requires the use of an incompatible type
* ITarget: expected interface for the client class
* Adaptee: contains the functionality that is required by the client
* Adapter: provides the link between the incompatible client and Adaptee classes.The adapter implements the ITarget interface and contains a private instance of the Adaptee class



Decorator Design Pattern

* extends the functionality of individual objects by wrapping then with one or more decorator classes
* this decorators can modify existing members and add new methods and properties at run-time
* provides a flexible alternative to using inheritance to modify behavior
* ComponentBase: This abstract class is the base class for both the concrete components and all decorator classes. It defines any standard members that will be implemented by these classes
* ConcreteComponent: inherits from the ComponenetBase class. May have multiple concrete component classes, each defining a type of object that may be wrapped by the decorators
* DecoratorBase: abstract base class for all decorators for components. The class inherits its public interface from ComponentBase so that decorators can be used in place of concrete object
* ConcreteDecorator: provides a decorator components. additional methods can be included in the decorator



Observer Design Pattern

* defines a link between objects so that when one object's state changes, all dependent objects are updated automatically
* allows communication between objects in a loosely coupled manner
* SubjectBase: abstract base class for concrete subject. It contains a private collection of the observers that are subscribed to a subject and methods to allow new subscriptions to be added and existing ones to be removed
* ConcreteSubject: each concrete subject maintains its own state. When a change is made to that state, the object call the base class's Notify method to indicate this to all of its observers
* ObserverBase: abstract base class for all observers. It defines a method to be called when he subject's state changes
* ConcreteObserver: concrete observer objects are the subscribers that react to changes in the subject's state. When update method is called, it examined the subject to determine which information has changed

