# Assignment 03 Application Design: Patterns and Frameworks 44642

Answer **all** the questions below. In your answer for each question explain with sample code or image whichever is preferable.

1. **What are generics?**

*Generics are a programming language feature that allows for the creation of functions, classes, and data structures that can operate on a variety of data types. They enable developers to write code that is more reusable and type-safe, by allowing for the specification of a type of parameter that can represent any type, without the need for separate implementations for each type.*

**JAVA CODE: Question01 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we change the scope of the overridden method in the subclass for private, public, default and protected? Explain how can it be changed for each scope?**

*We can change the scope of an overridden method in the subclass to a more permissive access level (i.e., from private to protected, from protected to public, etc.), but we cannot make it less permissive (i.e., from public to private, from protected to private, etc.). This is because the subclass is supposed to be a more specific type of superclass and thus cannot have less functionality than the superclass.*

**JAVA CODE: Question02 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What is the covariant return type?**

*Covariant return type is a feature in Java that allows a method in a subclass to return a type that is a subclass of the type returned by the method in the superclass. In other words, it allows a method to return a more specific type than the method it overrides.*

**JAVA CODE: Question03 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we override the static and private methods? Why?**

*We cannot override static methods in Java. When a subclass defines a method with the same name, return type and parameters as a static method in the superclass, it does not override the static method, but instead, it hides the static method of the superclass.*

*Private methods are also not overridden in Java, because they are not visible to the subclass. If a subclass defines a private method with the same name, return type and parameters as a private method in the superclass, it is a new method and not related to the private method in the superclass.*

**JAVA CODE: Question04 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Difference between String Buffer and StringBuilder?**

***StringBuffer*** *is a thread-safe class, which means that its methods are synchronized, and multiple threads can safely access it at the same time. However, this can also make it slower, especially in single-threaded applications.*

***StringBuilder*** *is not thread-safe, but it is faster than StringBuffer in single-threaded applications, because its methods are not synchronized.*

**JAVA CODE: Question05 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Difference between String class and String Buffer?**

*The String class and StringBuffer class are both used for manipulating strings in Java, but they have some key differences.*

***String objects*** *are immutable, which means that once a String object is created, its contents cannot be changed. This can make it less efficient to manipulate strings using the String class, because every time you modify a String object, a new String object is created in memory.*

*On the other hand,* ***StringBuffer*** *objects are mutable, which means that you can modify their contents without creating new objects in memory. This can make StringBuffer more efficient for manipulating strings, especially when you need to perform a lot of modifications.*

**JAVA CODE: Question06 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we declare constructor as final?**

*No, we cannot declare constructors as final in Java. It is not possible to declare constructors as final because constructors are responsible for creating objects, and final keyword can only be used to prevent methods or classes from being overridden or extended, respectively.*

*Trying to declare a constructor as final will result in a compilation error.*

**JAVA CODE: Question07 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we have try without catch block in java?**

*Yes, we can have a try block without a corresponding catch block in Java, as long as there is a finally block. The finally block is executed regardless of whether an exception is thrown or not.*

**JAVA CODE: Question08 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What is try with the resource?**

*Try-with-resources is a feature introduced in Java 7 that simplifies the process of closing resources. A resource is an object that must be closed after the program is finished with it, such as a file or database connection. Try-with-resources automatically closes these resources at the end of the block.*

**JAVA CODE: Question09 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we modify the throws clause of the superclass method while overriding it in the subclass?**

*No, we cannot modify the throws clause of the superclass method while overriding it in the subclass. The throws clause of the overridden method in the subclass must be compatible with the throws clause of the superclass method.*

**JAVA CODE: Question10 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What is an association, aggregation, and composition in UML?**

***Association*** *relationship is a structural relationship in which different objects are linked within the system. It exhibits a binary relationship between the objects representing an activity.*

***Aggregation*** *is a subset of association, is a collection of different things. It is more specific than an association. It describes a part-whole or part-of relationship.*

***Composition*** *is a type of association relationship where a composite object has a life cycle that is dependent on the life cycle of its components. It represents a "part-whole" or "has-a" relationship, where the composite object owns its components, and the components cannot exist without the composite object.*

**JAVA CODE: Question11 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Difference between final, finally and finalize()?**

***Final*** *is the keyword and access modifier which is used to apply restrictions on a class, method, or variable. Final keyword is used with the classes, methods, and variables.*

***Finally*** *is the block in Java Exception Handling to execute the important code whether the exception occurs or not. Finally block is always related to the try and catch block in exception handling.*

***Finalize*** *is the method in Java which is used to perform clean up processing just before object is garbage collected. Finalize () method is used with the objects.*

**JAVA CODE: Question12 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Difference between Vector and ArrayList?**

*The* ***ArrayList*** *class is a standard collection class that extends AbstractList and implements the List interface from Collection Interfaces. It is specified inside java.util package. A typical array in Java is always a fixed length. This means that once something is formed, its size does not change on demand.*

***Vector*** *is a legacy class that has been redesigned to support the collection class in the hierarchy of the Collection Framework. The AbstractList class extends the vector class in java.util package and the List interface implements the interface.*

**JAVA CODE: Question13 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What are the different ways to make ArrayList methods synchronized?**

To **synchronize an**ArrayList*, we can use two JDK-provided methods.*

***1. Collections.synchronizedList() method.***

*The synchronizedList() returns a synchronized and thread-safe list backed by the specified list.*

*It is recommended that we should manually synchronize the returned list when traversing it via Iterator, Spliterator or Stream else it may result in non-deterministic behavior. No explicit synchronization is needed to add, or remove elements from this synchronized arraylist.****2. Using CopyOnWriteArrayList.***

*The CopyOnWriteArrayList is a thread-safe variant of ArrayList in which all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array. This class is very useful when we cannot or do not want to synchronize the traversals of the arraylist. It is part of thread-safe Java collections.*

**JAVA CODE: Question14 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Difference between Hash table and Hash Map?**

***HashMap*** *is used for the implementation of the Map interface in Java. Since it is a type of data structure, it is used to save the data. The data is stored in the form of keys and values. The key and value can be accessed with the index of another type. The HashMap is unsynchronized.*

*The single null key and multiple null values can be stored with the HashMap.*

***HashTable*** *is also a type of data structure and hence is used to store or arrange data. This data structure also stores data in the form of key and value pairs. The hashtable comprises the following parts like Hash Function and Array. In Hashtable, non-null objects can be stored as a value and key pairs but null values are not stored in this, unlike HashMap. This is a synchronized way of storing data.*

**JAVA CODE: Question15 package -> NagaAkhil\_ChaparalaAssignment03**

1. **In Java 8, explain how Hasp Map internally works?**

***HashMap*** *works on the principle of hashing, we have put(key, value) and get(key) method for storing and retrieving Objects from HashMap. When we pass Key and Value object to put() method on Java HashMap, HashMap implementation calls hashCode method on Key object and applies returned hashcode into its own hashing function to find a bucket location for storing Entry object, important point to mention is that HashMap in Java stores both key and value object as Map. Entry in a bucket which is essential to understand the retrieving logic.*

**JAVA CODE: Question16 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Difference between fail fast and fail-safe iterator?**

***Fail-Fast Iterator:*** *A fail-fast iterator immediately throws a ConcurrentModificationException if the collection is structurally modified during iteration. This is achieved by keeping a mod count of the collection and checking it before each operation. Fail-fast iterators are used by most of the collection classes such as ArrayList, HashMap, HashSet, etc.*

***Fail-Safe Iterator:*** *A fail-safe iterator operates on a copy of the collection and hence does not throw a ConcurrentModificationException even if the collection is modified while iterating over it. Fail-safe iterators are used by some of the concurrent collection classes such as CopyOnWriteArrayList, ConcurrentHashMap, etc.*

**JAVA CODE: Question17 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we start the thread twice?**

*No. After starting a thread, it can never be started again. If you does so, an IllegalThreadStateException is thrown. In such case, thread will run once but for second time, it will throw exception.*

**JAVA CODE: Question18 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What are the different ways to create a thread in java? Which one is preferred?**

*There is exactly one way to create a new thread in Java and that is to instantiate java.lang.Thread (to actually run that thread you also need to call start()). Everything else that creates threads in Java code falls back to this one way behind the cover.*

*There are two different ways to specify which code to run in that Thread:*

*Implement the interface java.lang.Runnable and pass an instance of the class implementing it to the Thread constructor. Extend Thread itself and override its run() method. The first approach (implementing Runnable) is usually considered the more correct approach because you don't usually create a new "kind" of Thread, but simply want to run some code (i.e. a Runnable) in a dedicated thread.*

**JAVA CODE: Question19 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What are the different states a thread will go through?**

*A thread can be in one of the following states:*

*NEW: A thread that has not yet started is in this state.*

*RUNNABLE: A thread executing in the Java virtual machine is in this state.*

*BLOCKED: A thread that is blocked waiting for a monitor lock is in this state.*

*WAITING: A thread that is waiting indefinitely for another thread to perform a particular action is in this state.*

*TIMED\_WAITING: A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.*

*TERMINATED: A thread that has exited is in this state.*

*A thread can be in only one state at a given point in time. These states are virtual machine states which do not reflect any operating system thread states.*

**JAVA CODE: Question20 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What is Serialization? How do we achieve it?**

***Serialization*** *is the process of converting a data object—a combination of code and data represented within a region of data storage—into a series of bytes that saves the state of the object in an easily transmittable form.*

*For serializing the object, we call the writeObject() method of ObjectOutputStream class, and for deserialization we call the readObject() method of ObjectInputStream class. We must have to implement the Serializable interface for serializing the object.*

**JAVA CODE: Question21 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What is immutable class? Is String class immutable?**

*Immutable classes define objects which, once created, never change their value. A variable of an immutable type may only be changed by re-assigning to that variable. When we wish to only modify some portion of an immutable class, we are compelled to reassign the whole object.*

*The String is immutable in Java because of the security, synchronization and concurrency, caching, and class loading. The reason of making string final is to destroy the immutability and to not allow others to extend it. The String objects are cached in the String pool, and it makes the String immutable.*

**JAVA CODE: Question22 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Do immutable classes thread safe? If yes then how?**

*Yes, immutable classes are thread-safe by default. This is because once an object of an immutable class is created, its state cannot be modified. Therefore, multiple threads can safely access and use the same instance of the immutable class without any risk of concurrent modification or race conditions. In Java, some common examples of immutable classes are String, Integer, Double, etc. These classes have private final fields and no setter methods, ensuring that their state cannot be changed once they are created.*

**JAVA CODE: Question23 package -> NagaAkhil\_ChaparalaAssignment03**

1. **Can we call the garbage collector explicitly? Will it trigger the garbage collector?**

*Yes, we can call the garbage collector explicitly in Java using the System.gc() method. However, there is no guarantee that the garbage collector will run immediately after calling this method.*

*The System.gc() method is simply a request to the JVM to run the garbage collector, but it is up to the JVM to decide when to run it. In fact, some JVMs may ignore this request altogether.*

**JAVA CODE: Question24 package -> NagaAkhil\_ChaparalaAssignment03**

1. **What are Java 8 features? Explain all of them with examples?**

***Lambda Expressions****: Lambda expressions allow you to write more concise and readable code by enabling you to pass functions as arguments to other methods.*

***Stream API:*** *The Stream API allows you to process collections of data in a functional, parallel and efficient way. It provides a set of methods for filtering, mapping, sorting, and reducing elements in a collection.*

***Date and Time API:*** *The new Date and Time API simplifies date and time handling in Java by providing a set of classes for representing dates, times, durations, and periods.*

***Default and Static Methods in Interfaces:*** *Java 8 allows you to define default and static methods in interfaces, making it easier to add new functionality to existing interfaces without breaking backward compatibility.*

***Optional Class:*** *The Optional class is a container object that may or may not contain a non-null value. It helps to avoid NullPointerExceptions in your code.*

***Method References:*** *Method references allow you to refer to methods by their names rather than by invoking them directly, making your code more readable and concise.*

***Functional Interfaces:*** *Java 8 introduces functional interfaces, which are interfaces that have only one abstract method. They are used to represent lambda expressions and method references.*

***Parallel Array Sorting:*** *The Arrays class in Java 8 provides parallelSort() method that allows you to sort arrays in parallel, making it faster and more efficient.*

***Nashorn JavaScript Engine:*** *Java 8 includes a new JavaScript engine called Nashorn, which is based on the Mozilla Rhino engine. It allows you to run JavaScript code from within Java applications.*

***Base64 Encode Decode:*** *Java 8 introduces Base64 class, which provides methods for encoding and decoding Base64 data.*

***Collectors Class:*** *The Collectors class provides a set of predefined collectors that can be used with the Stream API to collect and summarize data.*

***CompletableFuture Class:*** *The CompletableFuture class is used to perform asynchronous operations in Java. It allows you to specify a callback that will be executed when the operation completes.*

***Repeating Annotations:*** *Java 8 introduces the ability to use repeating annotations, which allows you to use the same annotation more than once on the same element.*

***Type Annotations:*** *Type annotations allow you to apply annotations to any type use, such as the type of a variable, the return type of a method, or the type of a parameter.*

***Improved Type Inference:*** *Java 8 improves type inference by allowing the compiler to infer the type of a variable based on its initial value. This makes it easier to write concise and readable code.*

**JAVA CODE: Question25 package -> NagaAkhil\_ChaparalaAssignment03**

1. **How to make a pure singleton?**

*A pure singleton is a class that ensures only one instance of the class is created and can be accessed globally. To create a pure singleton, we need to follow these steps:*

* *Make the constructor of the class private to prevent other classes from creating an instance of the class.*
* *Declare a static final variable of the class type to hold the instance of the class.*
* *Provide a public static method to return the instance of the class.*
* *Ensure that the public method returns the same instance every time it is called.*

*This approach is thread-safe and ensures that only one instance of the class is created.*

**JAVA CODE: Question26 package -> NagaAkhil\_ChaparalaAssignment03**

1. **How to make a singleton synchronized?**

*To make a singleton class synchronized, we need to synchronize the access to the instance of the class. The general approach is to make the getInstance() method synchronized. Here is a summary of the steps to create a synchronized singleton class:*

* *Create a private constructor to prevent other classes from instantiating the class directly.*
* *Create a private static variable to hold the single instance of the class.*
* *Create a public static synchronized method called getInstance() to return the single instance of the class.*
* *Inside the getInstance() method, create the instance if it doesn't already exist.*

*By synchronizing the getInstance() method, we ensure that only one thread can access the instance at a time, and it prevents multiple instances from being created. However, this approach can have performance overhead due to synchronization, so it is recommended to use it only when necessary.*

**JAVA CODE: Question27 package -> NagaAkhil\_ChaparalaAssignment03**

**GitHub Repository Link:**

[**https://github.com/chnagaakhil/Application-Design-Patterns-and-Frameworks-.git**](https://github.com/chnagaakhil/Application-Design-Patterns-and-Frameworks-.git)

**Submission:**  Change the document name as ***Lastname*Assignment03** where *Lastname* is your Last name and submit.