# 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?

**Answer:** Generic programming is what we refer to in Object Oriented Programming (OOP) as the practice of using strategies to build code that may be reused, where, for instance, the attributes or argument types are maintained broad and thus not particularly defined. Subject to the conditions you specify, generic code enables you to construct adaptable, reusable functions and types that can work with any type. Generic programming, n.d., explains how to build code that avoids duplication and states its meaning clearly and abstractly. This may be somewhat less efficient, but it can be quite beneficial for the flexibility and reuse of the code.

**Sample Code:** A screenshot of a computer

Description automatically generated

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

**Answer**: Yes, we can modify the overridden method's scope in the subclass. We must be aware, nevertheless, that we cannot make the method less accessible. When modifying the method's accessibility, the following consideration must be made. You can alter the private to protected, public, or default. You can set the protected to public or default. Public can be chosen as the default. The public always will be the public.

**Sample Code:**

A screenshot of a computer

Description automatically generated

1. What is the covariant return type?

**Answer**: According to the covariant return type, the return type may vary in the same way as the subclass. Changing the return type was not a supported method overriding strategy prior to Java 5. However, with Java 5, it is now possible to override a method by altering the return type. This is true if a subclass overrides a method whose return type is non-Primitive but changes it to a subclass type.

**Sample Code:**

A screenshot of a computer

Description automatically generated

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

**Answer**: No. In another class, a private method won't even be available or visible. Therefore, if you attempt to override it, you will be creating a new private method with the same name. The superclass method will be hidden if we write an identical static method with the same name and parameter, which is known as method hiding.

**Sample Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Difference between String Buffer and StringBuilder?

**Answer**: String Buffer is thread safe and synchronized. It signifies that two threads cannot call the String Buffer’s functions at the same time. StringBuilder is not thread safe because it is not synchronized. It implies that two threads can simultaneously be called StringBuilder methods. StringBuilder is more efficient than String Buffer.

**Sample Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Difference between String class and String Buffer?

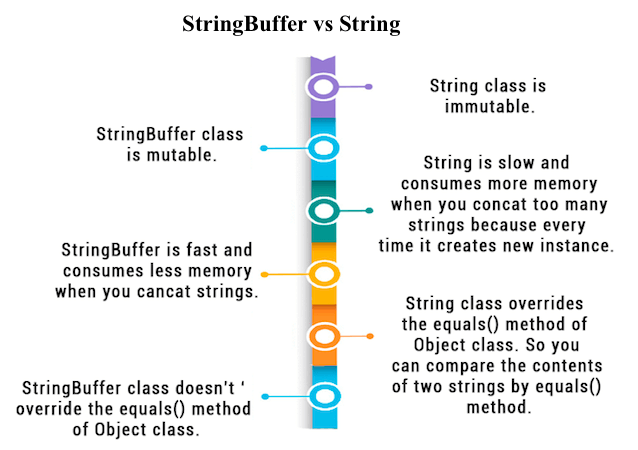
**Answer**: String is an immutable class, meaning that once generated, its object cannot be changed, albeit it can reference other objects. Immutable objects are thread safe and highly helpful in multithreading environments since no single thread may alter the state of the object. Mutable classes called string buffers can be used to perform operations on string objects like reversing strings, concatenating strings, and more. Without making a new object out of the string, we can alter strings. Also thread safe is the string buffer.

**Note**: The Sample Code for String Buffer has already been attached in previous question

**Sample Code & Image:**

A screenshot of a computer

Description automatically generated



1. Can we declare constructor as final?

**Answer:** You cannot make a constructor final. Any subclass cannot override a final method. The last modification, as was already explained, prevents a method from being changed in a subclass. Making a method final would primarily serve to prevent other parties from altering its content. However, a subclass inherits all a superclass's members, excluding constructors, through inheritance. In other words, since constructors in Java cannot be inherited, final is not required before constructors. Java forbids the use of the final keyword before a constructor as a result. A build time error is produced if you attempt, like in the example below.

**Sample Code:**

A screenshot of a computer

Description automatically generated

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

**Answer:** Yes, by utilizing the finally block, we may have a try without a catch block. Try can be used with finally. As you are aware, the finally block always executes even if the try block contains an exception or a return statement, unless the system is involved.exit ( )

**Sample Code:**

A screenshot of a computer

Description automatically generated

1. What is try with the resource?

**Answer:** In Java, a try statement that declares one or more resources is known as a try-with-resources statement. The resource is an object that needs to be closed once the program is finished. At the conclusion of the statement execution, the try-with-resources statement makes sure that each resource is closed.

Any object that complies with Java.lang can be passed.All objects that implement java.io fall under the AutoCloseable category.Closeable.

**Sample Code:**

A screenshot of a computer

Description automatically generated with medium confidence

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

**Answer:** Yes, while overriding the superclass method in the subclass, we can change the throws clause. However, there are several guidelines that must be observed when overriding when handling exceptions. The subclass overridden method can declare the unchecked exception but not the checked exception if the superclass method does not declare an exception. If the superclass method declares an exception, the overridden subclass method may do the same, declare a subclass exception, or declare none nevertheless, it may not declare a parent exception.

**Sample Code:**

A screenshot of a computer

Description automatically generated

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

**Answer:** One object is associated to another in object-oriented programming to use the functionality and service offered by the other object. In object-oriented generic program design, this connection between two objects is referred to as the association, and in the Unified Modelling Language, or UML, it is represented by an arrow. There is a slight distinction between composition and aggregation, which is also mirrored by their UML notation, even though both Composition and Aggregation are ways of associating two objects. When one class owns other classes and those other classes are unable to exist in a meaningful way once the owner is destroyed, we refer to the association between the two objects as a composition. Composition outperforms Aggregation in strength. In a nutshell, an association is a connection between two objects, and an association.

**Sample Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

**Answer:** The Java language has a keyword called final that can be used to create constants, prevent class inheritance, and prevent class overriding.

Finally - It is carried out following the catch block. Basically, when there are numerous catch blocks, we use it to place some common code.

Finalize: In Java, finalize is a method used to carry out cleanup operations immediately before an object is garbage collected.

**Sample Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Difference between Vector and ArrayList?

**Answer:**

1) The array list is not synced. The vector is coordinated.

2) If the array's capacity is reached, the Array List increases the size of the array by 50%.

If the total number of elements exceeds the array's limit, a vector increment of 100% signifies that the array size is doubled.

Three.

3) Array List is a modern class. It first appears in JDK 1.2. The class Vector is an old one.

4) Because it is not synchronized, Array List is quick. Due to its synchronization, vector is slow since it keeps other threads in a runnable or non-runnable state in a multithreading environment until the current thread releases the lock on the object.

5) Array List iterates through the elements using the Iterator interface. A Vector can navigate the elements using either the Iterator interface or the Enumeration interface.

**Sample Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

**Answer:**

The Array List implementation is not by default synchronized. It implies that it needs to be externally synchronized if a thread affects it structurally and many threads access it at the same time. A structural change either directly changes the backing array's size or implies the addition or deletion of one or more elements from the list. A structural modification is not when the value of an existing element is changed.

A Synchronized ArrayList can be created in two different ways.

1. The synchronizedList() method of collections: All access to the backing list must be made through the returning list in order to perform serial access. When iterating over the returning list, the user must always manually synchronize.

2. CopyOnWriteArrayList : An ArrayList version that is thread-safe that implements all mutative actions (such as add, set, remove, etc.) by making a separate copy of the underlying array. Instead of using a vector or other thread-safe collection, it achieves thread safety by making a distinct copy of the List.

It helps prevent conflict between running threads when you are unable or unable to synchronize the traversal.

It is expensive since every writing action (such as an add, set, or removal) necessitates a separate Array copy.

When you have a list and need to traverse its elements but don't edit it frequently, it is incredibly efficient.

**Sample Code:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Difference between Hash table and Hash Map?

**Answer:**

HashMap Key permits a single null, not thread-safe and asynchronized. Hashtable would have Key and value won't support null values & synchronized (safe for threads).

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

**Answer:**

Although a hash map makes use of a hash table, it is internally built using an array and a linked list. Internally, a hash map generates an array of buckets whenever you specify one. The nodes, or nodes in a linked list, are the buckets. A node may stand for Key Value, Hash code & Address of the Next Node

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

**Answer:**

When one thread is iterating over a collection object and another thread structurally modifies the collection by adding, removing, or changing objects on the underlying collection,

Fail - fast Iterators throws the Concurrent Modification Exception. They are known as fail-fast because they strive to throw an exception as soon as they experience failure.

In contrast, fail-safe iterators use copies of collections rather than the original collections.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Can we start the thread twice?

**Answer:**

No. A thread cannot be restarted after it has begun.

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

**Answer:**

There are two ways to tell the thread which code to execute:

1. Extension of a Thread Through the use of a Runnable Interface & Implement the Java Lang interface. Runnable and provide the Thread constructor with an instance of the class that implements it.

2. Override the run () method of the Thread itself by extending it.

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

**Answer:** The various states of a thread are ready, running, waiting, and dead.

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

**Answer:** The process of writing an object's state into a byte-stream in Java is called serialization. The technologies Hibernate, RMI, JPA, EJB, and JMS use it the most. We call the writeObject() function of the ObjectOutputStream class to serialize the object.

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

**Answer:** An Object whose values cannot be changed is called immutable class and Yes, String is class immutable

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

**Answer:** Simply put, "thread-safe" refers to the requirement that two or more threads collaborate on a common object or resource. The modifications made by any other thread shouldn't be overridden by them.

Since the String class is immutable, any attempts to alter it by a thread result in the creation of a new object. Therefore, discussing another thread would be like traveling to the Sun. The catch is that typically, we use the same old reference to point that newly formed object, which means that even the same thread cannot make any modifications to the original object.

When writing code, we just use the object's reference to evaluate any changes to the object.

In the first statement, the string "123" is shared by two threads.

To be executed in statement two: str = str+"FirstThread"

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

**Answer:** Garbage collection is a mechanism used by the CLR (Common Language Runtime) to manage the resources used by your application. Note that when you create objects in. Net, they are put in the managed heap, and the runtime will take care of clearing them up after you are finished with them. The runtime conducts a stack walk to identify the objects that are reachable and those that are not when you use the GC. Collect () method. In addition, the primary

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

**Answer:** Some of the important Java 8 features are;

* forEach() method in Iterable interface: The forEach function in java.lang was added in Java 8.Iterable interface allows us to concentrate on business logic while writing code. Java.util.function is a parameter for the forEach method.Having our business logic in a different area that we can reuse helps since we can use a consumer object as an argument.
* default and static methods in Interfaces: Java 8 improve interfaces by adding a method and implementation. To design interfaces with method implementation, we can utilize the static and default keyword. for Each method's Iterable interface implementation is as follows.
* Functional Interfaces and Lambda Expressions: An option to prevent the unintentional insertion of abstract methods in functional interfaces is the @FunctionalInterface annotation. It's recommended to utilize and can be compared to a @Override annotation. java.lang.A functional interface that only has the abstract method run() is called Runnable.
* Java Stream API for Bulk Data Operations on Collections: In order to execute operations on the collection similar to filter, map, and reduce, a new java.util.stream has been added in Java 8. Both parallel and sequential execution are supported by the Stream API. Because I work a lot with collections and we typically need to filter out items from big data, this is one of the best aspects for me.
* Java Time API: For months and days of the week, the new Time API prefers Enums over integer constants. DateTimeFormatter is one of the helpful classes that transforms Date Time objects into strings.
* Collection API improvements: The default iterator method forEachRemaining(Consumer action) executes the specified action for each subsequent element until all subsequent elements have been processed or the action fails. To delete every element from this collection that satisfies the specified predicate, use the collection default method removeIf(Predicate filter). The spliterator() method of a collection returns an instance of a Spliterator that may be used to iterate over elements in either parallel or serial fashion. methods compute(), merge(), and replaceAll() for maps. Enhancement of HashMap class performance with Key Collisions
* Concurrency API improvements: ConcurrentHashMap has methods for compute, forEach, forEachEntry, forEachKey, forEachValue, merge, reduce, and search. Explicit completion of a CompletableFuture is possible (setting its value and status).

Executors use the newWorkStealingPool() function to construct a work-stealing thread pool with the goal parallelism level set to all available processors.

* Java IO improvements: Among the IO advancements I am aware of are:

Files.list(Path dir) provides a lazily filled Stream with the directory items as its elements. All lines from a file are read as a Stream using the Files.lines(Path path) method. By looking for files in a file tree with its root at a specified starting file, Files.find() provides a Stream that is slowly filled with Path.

Lines read from this BufferedReader are returned in a Stream using the BufferedReader.lines() method.

1. How to make a pure singleton?

**Answer:** When developing a singleton class, keep the following in mind when declaring a class as a singleton class:

* 1. Create a private constructor.
  2. Create a static method with this singleton class's object as its return type. This static method was created using the idea of lazy initialization.

1. How to make a singleton synchronized?

**Answer:** Synchronization prevents a block of code to be executed by more than one thread at the same time. To improve our Singleton pattern I have just added synchronized keyword in method declaration. In the following example only one thread can enter the getInstance() method and execute code at the time.

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