# 1)What is the heap memory in Java?

# The heap memory in Java is the area of memory where objects are stored. All objects created in Java are stored in the heap memory.

# 2)What is the stack memory in Java?

# The stack memory in Java is the area of memory where primitive data types and references to objects are stored. The stack memory is used to store local variables and method calls.

# 3)What is the difference between heap and stack memory in Java?

# The heap memory is used to store objects, while the stack memory is used to store primitive data types and references to objects. Objects are allocated on the heap and references to those objects are stored on the stack.

# 4)What are the various access specifiers in Java?

# Public - The classes, methods, or variables which are defined as public, can be accessed by any class or method

# Protected - Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.

# Default - Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.

# Private- The private class, methods, or variables defined as private can be accessed within the class only.

# 5)Java String Pool

# A collection of strings in Java's Heap memory is referred to as Java String Pool. In case you try to create a new string object, JVM first checks for the presence of the object in the pool. If available, the same object reference is shared with the variable, else a new object is created.

# 6) Significance of Abstract keyword

# The abstract keyword is a non-access modifier, used for classes and methods. It is the process of hiding certain details and showing only essential information to the user.

# 7) What is abstract class ?

# It is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class). An abstract class can have both abstract and regular methods.

# 8) What is abstract method ?

# It can only be used in an abstract class, and it does not have a body. The body is provided by the class which inherits.

# 9) Can a method or a class be final and abstract at the same time?

# No, it is not possible. A class or a method cannot be final or abstract at the same time because the final method or final class cannot be further modified whereas an abstract class or an abstract method must be modified further.

# 10) Abstract methods of abstract classes are mandatory to be overridden by sub classes. Is that true? YES

# 11) Different packages can have classes with the same name – true

# 12)What is boxing and unboxing.?

# Wrapping of primitive content into corresponding wrapper class object is called boxing. Unwrapping the wrapper class object into corresponding primitive content is called unboxing.

# 13)What is the difference between auto-widening, auto-upcasting and auto-boxing.?

# Auto-widening occurs when small sized primitive type is casted to big sized primitive type. Auto-upcasting occurs when sub class type is casted to super class type. Auto-boxing occurs when primitive type is casted to corresponding wrapper class.

# 14)What is Thread in Java?

# Threads are basically the lightweight and smallest unit of processing that can be managed independently by a scheduler.

# 15)A thread can have one of the following states during its lifetime:

# New: In this state, a Thread class object is created using a new operator, but the thread is not alive. Thread doesn't start until we call the start() method.

# Runnable: In this state, the thread is ready to run after calling the start() method. However, the thread is not yet selected by the thread scheduler.

# Running: In this state, the thread scheduler picks the thread from the ready state, and the thread is running.

# Waiting/Blocked: In this state, a thread is not running but still alive, or it is waiting for the other thread to finish.

# Dead/Terminated: A thread is in terminated or dead state when the run() method exits.

# 16)What are the two ways of implementing thread in Java?

# Extending the Thread class AND Implementing Runnable interface in Java

# 17)What are the wait() and sleep() methods?

# wait(): As the name suggests, it is a non-static method that causes the current thread to wait and go to sleep until some other threads call the notify () or notifyAll() method for the object’s monitor (lock). It simply releases the lock and is mostly used for inter-thread communication. It is defined in the object class, and should only be called from a synchronized context.

# sleep(): As the name suggests, it is a static method that pauses or stops the execution of the current thread for some specified period. It doesn’t release the lock while waiting and is mostly used to introduce pause on execution. It is defined in thread class, and no need to call from a synchronized context.

# 18)What’s the difference between notify() and notifyAll()?

# notify(): It sends a notification and wakes up only a single thread instead of multiple threads that are waiting on the object’s monitor.

# notifyAll(): It sends notifications and wakes up all threads and allows them to compete for the object's monitor instead of a single thread.

# 19) What is start() method?

# start(): In simple words, the start() method is used to start or begin the execution of a newly created thread. When the start() method is called, a new thread is created and this newly created thread executes the task that is kept in the run() method. One can call the start() method only once.

# 20) what is run() method >

# run(): In simple words, the run() method is used to start or begin the execution of the same thread. When the run() method is called, no new thread is created as in the case of the start() method. This method is executed by the current thread. One can call the run() method multiple times.

# 21)Explain the meaning of the deadlock and when it can occur?

# Deadlock, as the name suggests, is a situation where multiple threads are blocked forever. It generally occurs when multiple threads hold locks on different resources and are waiting for other resources to complete their task.