**1. What is the different between stack and heap memory?**

## Ans. Difference Between Stack and Heap Memory

The following table summarizes all the major differences between stack memory and heap space.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Stack Memory** | **Heap Space** |
| **Application** | It stores items that have a very short life such as **methods, variables,** and **reference variables** of the objects. | It stores **objects** and Java Runtime Environment (**JRE**) classes. |
| **Ordering** | It follows the **LIFO** order. | It does not follow any order because it is a dynamic memory allocation and does not have any fixed pattern for allocation and deallocation of memory blocks. |
| **Flexibility** | It is **not flexible** because we cannot alter the allocated memory. | It is **flexible** because we can alter the allocated memory. |
| **Efficiency** | It has **faster** access, allocation, and deallocation. | It has **slower** access, allocation, and deallocation. |
| **Memory Size** | It is **smaller** in size. | It is **larger** in size. |
| **Java Options Used** | We can increase the stack size by using the JVM option -Xss. | We can increase or decrease the heap memory size by using the -[Xmx](https://www.javatpoint.com/java-xmx) and -Xms JVM options. |
| **Visibility or Scope** | The variables are visible only to the owner thread. | It is visible to all threads. |
| **Generation of Space** | When a thread is created, the operating system automatically allocates the stack. | To create the heap space for the application, the language first calls the operating system at run time. |
| **Distribution** | Separate stack is created for each object. | It is shared among all the threads. |

|  |  |  |
| --- | --- | --- |
| **Parameter** | Stack Memory | Heap Space |
| **Exception Throws** | JVM throws the **java.lang.StackOverFlowError** if the stack size is greater than the limit. To avoid this error, increase the stack size. | JVM throws the **java.lang.OutOfMemoryError** if the JVM is unable to create a new native method. |
| **Allocation/ Deallocation** | It is done automatically by the **compiler**. | It is done manually by the **programmer**. |
| **Cost** | Its cost is **less**. | Its cost is **more** in comparison to stack. |
| **Implementation** | Its implementation is **hard**. | Its implementation is **easy**. |
| **Order of allocation** | Memory allocation is **continuous**. | Memory allocated in **random** order. |
| **Thread-Safety** | It is thread-safe because each thread has its own stack. | It is not thread-safe, so properly synchronization of code is required. |

**2. How many types of memory allocated by JVM?**

Let’s see about them in brief:

#### 1. Class (Method) Area

The class method area is the memory block that stores the class code, variable code(static variable, runtime constant), method code, and the constructor of a Java program. (Here method means the function which is written inside the class). It stores class-level data of every class such as the runtime constant pool, field and method data, the code for methods.

#### 2. Heap

The Heap area is the memory block where objects are created or objects are stored. Heap memory allocates memory for class interfaces and arrays (an array is an object). It is used to allocate memory to objects at run time

#### 3. Stack

Each thread has a private JVM stack, created at the same time as the thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.

Java Stack stores frames and a new frame is created each time at every invocation of the method. A frame is destroyed when its method invocation completes

#### 4. Program Counter Register:

Each JVM thread that carries out the task of a specific method has a program counter register associated with it. The non-native method has a PC that stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.

#### 5. Native method Stacks:

Also called C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when it’s created And it can be of a fixed or dynamic nature.

**3. How to create a Clone of an Object?**

Ans. The **object cloning** is a way to create exact copy of an object. The clone() method of Object class is used to clone an object.

The **java.lang.Cloneable interface** must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates **CloneNotSupportedException**.

The **clone() method** is defined in the Object class. Syntax of the clone() method is as follows:

1. **class** Student18 **implements** Cloneable{
2. **int** rollno;
3. String name;
4. Student18(**int** rollno,String name){
5. **this**.rollno=rollno;
6. **this**.name=name;
7. }
8. **public** Object clone()**throws** CloneNotSupportedException{
9. **return** **super**.clone();
10. }
11. **public** **static** **void** main(String args[]){
12. **try**{
13. Student18 s1=**new** Student18(101,"amit");
14. Student18 s2=(Student18)s1.clone();
15. System.out.println(s1.rollno+" "+s1.name);
16. System.out.println(s2.rollno+" "+s2.name);
17. }**catch**(CloneNotSupportedException c){}
18. }
19. }

4. Clone method is used of either shallow or deep copy?

Shallow copy

**5. What is Shallow and Deep copy in java?**

## Shallow Copy

When we do a copy of some entity to create two or more than two entities such that changes in one entity are reflected in the other entities as well, then we can say we have done a shallow copy. In shallow copy, new memory allocation never happens for the other entities, and the only reference is copied to the other entities. The following example demonstrates the same.

## Deep Copy

When we do a copy of some entity to create two or more than two entities such that changes in one entity are not reflected in the other entities, then we can say we have done a deep copy. In the deep copy, a new memory allocation happens for the other entities, and reference is not copied to the other entities. Each entity has its own independent reference. The following example demonstrates the same.

|  |  |
| --- | --- |
| **Shallow Copy** | **Deep Copy** |
| It is fast as no new memory is allocated. | It is slow as new memory is allocated. |
| Changes in one entity is reflected in other entity. | Changes in one entity are not reflected in changes in another identity. |
| The default version of the clone() method supports shallow copy. | In order to make the clone() method support the deep copy, one has to override the clone() method. |
| A shallow copy is less expensive. | Deep copy is highly expensive. |
| Cloned object and the original object are not disjoint. | Cloned object and the original object are disjoint. |

**6. Can override method throw the exception?**

Yes, it is possible to throw an exception in an overriding method in Java, but there are some rules that must be followed. These rules depend on whether the exception is checked or unchecked, and whether the overridden method (the method of the parent class) declares an exception or not. Here are the rules:

* If the overridden method does not declare an exception, the overriding method can throw any unchecked exception, but not any checked exception. An unchecked exception is a runtime exception that does not need to be declared or handled explicitly, such as ArithmeticException or NullPointerException. A checked exception is a compile-time exception that must be declared in the method signature or handled with a try-catch block, such as IOException or SQLException.
* If the overridden method declares an exception, the overriding method can throw the same exception, a subclass of that exception, or no exception at all, but not a superclass of that exception. For example, if the overridden method throws an IOException, the overriding method can throw an IOException, a FileNotFoundException (which is a subclass of IOException), or no exception, but not an Exception (which is a superclass of IOException).

These rules are based on the principle of **Liskov substitution**, which states that a subclass object should be able to replace a superclass object without breaking the functionality of the program. If the overriding method throws a new or broader checked exception than the overridden method, it would violate this principle and cause a compile-time error.

class Task {

public void m1(){

System.out.println("Super class C");

}

}

class First extends Task{

public void m1() throws ArithmeticException{

System.out.println("overriding method ");

}

}

class Main{

public static void main(String args[]) {

Task obj = new First();

obj.m1();

}

}

**7. Why java does not support the pointer?**

1. **Platform Independence:** Java was designed to be platform-independent, meaning that a Java program can run on any platform with a compatible Java Virtual Machine (JVM). The use of explicit pointers could introduce platform-specific behavior and make it more challenging to ensure consistency across different platforms.
2. **Garbage Collection:** Java uses automatic memory management through garbage collection. This eliminates the need for developers to manually allocate and deallocate memory using pointers, reducing the risk of memory leaks and simplifying memory management.
3. **Simplicity and Ease of Use:** Java aimed to be a relatively simple and easy-to-learn language. The introduction of explicit pointers could add complexity and make the language less approachable for beginners.
4. **Security:** Explicit pointers can be exploited by malicious code to gain unauthorized access to memory, potentially leading to security vulnerabilities. By restricting direct memory access through pointers, Java enhances its security model.

**8. What are the responsibilities of Garbage Collector?**

1. **Memory Management:** The GC keeps track of all allocated memory and identifies objects that are no longer reachable or referenced by the program. These unreferenced objects are candidates for removal to free up memory.
2. **Identifying Garbage:** The GC identifies objects that are no longer accessible by traversing the object graph starting from the program's roots (e.g., local variables, static variables, and method call stacks). Objects that cannot be reached from these roots are considered garbage.
3. **Reclaiming Memory:** Once the GC has identified garbage objects, it reclaims the memory occupied by these objects, making it available for future allocations. This process is known as "garbage collection."
4. **Minimizing Memory Leaks:** By automatically managing memory, the GC helps prevent memory leaks, which can occur when objects are no longer needed but not explicitly deallocated by the program.
5. **Ensuring Memory Safety:** The GC helps prevent memory-related errors like null pointer exceptions and buffer overflows by ensuring that objects are only accessed while they are still valid and reachable.

**9. What is Singleton class in java?**

A Singleton class is a class that allows only one instance of itself to be created and provides a global point of access to that instance. This is achieved by making the constructor private, so that no other instances of the class can be created, and providing a static method that returns the single instance of the class.

1. **public** **class** Singleton {
2. **private** **static** Singleton instance;
3. **private** Singleton() {}
4. **public** **static** **synchronized** Singleton getInstance() {
5. **if** (instance == **null**) {
6. instance = **new** Singleton();
7. }
8. **return** instance;
9. }
10. }

**10. What is Mutable class in java, How we can create Mutable class?**

The immutable objects are objects whose value can not be changed after initialization. We can not change anything once the object is created. For example, **primitive objects** such as [int](https://www.javatpoint.com/int-keyword-in-java), [long](https://www.javatpoint.com/long-keyword-in-java), [float](https://www.javatpoint.com/float-keyword-in-java), [double](https://www.javatpoint.com/double-keyword-in-java), **all**[**legacy classes**](https://www.javatpoint.com/legacy-class-in-java)**,**[**Wrapper class**](https://www.javatpoint.com/wrapper-class-in-java)**,**[**String class**](https://www.javatpoint.com/methods-of-string-class), etc.

1. **public** **class** JtpExample1 {
2. **private** **final** String s;
3. JtpExample1(**final** String s) {
4. **this**.s = s;
5. }
6. **public** **final** String getName() {
7. **return** s;
8. }
9. **public** **static** **void** main(String[] args) {
10. JtpExample obj = **new** JtpExample("Core Java Training");
11. System.out.println(obj.getName());
12. }
13. }

**11. StringBuffer and StringBuilder in java.**

In Java, StringBuilder and StringBuffer are used to create and manipulate strings. [The main difference between the two is that StringBuilder is **not thread-safe** while StringBuffer is 1](https://www.geeksforgeeks.org/difference-between-stringbuffer-and-stringbuilder-in-java/)[2](https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/)[3](https://www.tutorialspoint.com/java/java_string_buffer.htm).

StringBuffer is a **synchronized** class, which means that it is **thread-safe**. This makes it slower than StringBuilder, which is **not synchronized**. [However, if thread safety is required, then StringBuffer should be used instead of StringBuilder2](https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/)[3](https://www.tutorialspoint.com/java/java_string_buffer.htm).

Both classes provide similar functionality, but StringBuilder is faster than StringBuffer. [Therefore, it is preferred for single-threaded programs 1](https://www.geeksforgeeks.org/difference-between-stringbuffer-and-stringbuilder-in-java/)[2](https://www.geeksforgeeks.org/string-vs-stringbuilder-vs-stringbuffer-in-java/)[3](https://www.tutorialspoint.com/java/java_string_buffer.htm).

Here’s an example of how to use StringBuilder:

StringBuilder sb = new StringBuilder();

sb.append("Hello");

sb.append(" World!");

System.out.println(sb.toString());

**12. What is Aggregation and Composition in java?**

Ans. Aggregation is a relationship where the child object can exist independently of the parent object. For example, a car has an engine, but the engine can also exist without the car. Aggregation is also called a “weak” has-a relationship, because the parent object does not own or control the child object. Aggregation is suitable when the parent object needs to use the functionality of the child object, but does not need to manage its creation or destruction.

Composition is a relationship where the child object cannot exist independently of the parent object. For example, a human has a heart, but the heart cannot exist without the human. Composition is also called a “strong” has-a relationship, because the parent object owns and controls the child object. Composition is appropriate when the parent object needs to manage the creation and destruction of the child object, or when the child object is an integral part of the parent object.

**13. What is wait and Sleep method in java?**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Wait()** | **Sleep()** |
| 1. | The Wait() method is related to the Object class. | The Sleep () method is related to the Thread class. |
| 2. |  | The Sleep () method does not release the lock on the object during Synchronization. |
| 3. | It is not a static method. | It is a static method. |
| 4. | At the time of the Synchronization, the Wait() method releases obj. | At the time of the Synchronization, the Sleep() method doesn't release the obj, i.e., lock. |
| 5. | We can call the Wait () method only from the Synchronized context. | We can call the Sleep () method from outside the Synchronized context. |
| 6. | The Sleep() method has two overloaded methods, which are as follows:   * sleep(long milliseconds, int nanoseconds) * sleep(long milliseconds) | The Sleep() method has three overloaded methods, which are as follows:   * Wait() * wait(long timeout, int nanoseconds) * wait(long timeout) |
| 7. | The constructor of the Wait() method is defined in the following way: public final void Wait(long timeout) | The constructor of the Sleep () method in the following way: public static void Sleep (long millis) throws Interrupted\_Execption |
|  |  |  |
|  |  |  |

**14. What is Serialization in java?**

**Serialization in Java** is a mechanism of writing the state of an object into a byte-stream. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.

The reverse operation of serialization is called deserialization where byte-stream is converted into an object. The serialization and deserialization process is platform-independent, it means you can serialize an object on one platform and deserialize it on a different platform.

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.

**15. What is the difference between String and char array?**

| **Strings** | **Character Arrays** |
| --- | --- |
| [String](https://www.geeksforgeeks.org/strings-in-java/) refers to a sequence of characters represented as a single data type. | Character Array is a sequential collection of data type char. |
| Strings are immutable. | Character Arrays are mutable. |
| Built in functions like [substring()](https://www.geeksforgeeks.org/substring-in-java/), charAt() etc can be used on Strings. | No built in functions are provided in Java for operations on Character Arrays. |
| ‘+’ can be used to appended strings together to form a new string. | ‘+’ cannot be used to append two Character Arrays. |
| The [charAt()](https://www.geeksforgeeks.org/java-string-charat-method-example/)method can be used to access characters at a particular index in a String. | The characters in a Character Array can be accessed normally like in any other language by using []. |
| Strings can be stored in any manner in the memory. | Elements in Character Array are stored contiguously in increasing memory locations. |
| All Strings are stored in the **String Constant Pool**. | All Character Arrays are stored in the [**Heap**](https://www.geeksforgeeks.org/heap-data-structure/). |
| Not preferred for storing passwords in Java. | Preferred for storing passwords in Java. |
| A String can be converted into Character Array by using the [toCharArray()](https://www.geeksforgeeks.org/java-string-tochararray-example/) method of String class. Eg: String s = “GEEKS”; char [] ch = s.toCharArray(); | A Character Array can be converted into String by passing it into a **String Constructor**. Eg: char[] a = {‘G’, ‘E’, ‘E’, ‘K’, ‘S’}; String A = new String(a); |
|  |  |

**16. What is Upcasting and Downcasting?**

A process of converting one data type to another is known as **Typecasting** and **Upcasting** and **Downcasting** is the type of object typecasting. In Java, the object can also be typecasted like the datatypes. **Parent** and **Child** objects are two types of objects. So, there are two types of typecasting possible for an object, i.e., **Parent to Child** and **Child to Parent** or can say **Upcasting** and **Downcasting**.

**17. What is finally keyword in java?**

**Java finally block** is a block used to execute important code such as closing the connection, etc.

Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

The finally block follows the try-catch block.

**Use of finally block**

finally block in Java can be used to put "**cleanup**" code such as closing a file, closing connection, etc.

The important statements to be printed can be placed in the finally block.

18) What is partially and fully checked exception ?

Ans. A fully checked exception is a checked exception where all its child classes are also checked, like IOException, and InterruptedException. A partially checked exception is a checked exception where some of its child classes are unchecked, like an Exception.

19 ) How java Acheive platform indepency ?

Java is a platform-independent language, meaning we run the same code on multiple platforms. Java achieves this using JVM and Byte Code. Java compiler converts the programming code into byte code. Byte code is platform-independent and can be run on any processor or system.

20) What is immutable class in java ?

An object is immutable when its state doesn't change after it has been initialized. For example, String is an immutable class and, once instantiated, the value of a String object never changes.

21) When to use string , when to use buffer and when to use builder ?

When you not want to do changes the value which you assign or you don’t want to change object value use string .

When you want to implement one thread at a time mechanism use string buffer.

When you want to faster performance use builder.

22) What is threadsafe ?

Thread safety in java is the process to make our program safe to use in multithreaded environment, there are different ways through which we can make our program thread safe. Synchronization is the easiest and most widely used tool for thread safety in java.

Every immutable class by default threadsafe.

23) What is object oriented ?

The **object-oriented programming** is basically a computer programming design philosophy or methodology that organizes/ models software design around data, or objects rather than functions and logic.

It simplifies software development and maintenance by providing major concepts such as **abstraction, inheritance, polymorphism**, and **encapsulation**. These core concepts support OOP.

24) what is dependency injection ?

The general concept behind dependency injection is called Inversion of Control. A Java class has a dependency on another class, if it uses an instance of this class. We call this a class dependency.

25 ) What is tightly couple and loosely couple ?

Tight coupling does not provide the concept of interface. But loose coupling helps us follow the GOF principle of program to interfaces, not implementations. In Tight coupling, it is not easy to swap the codes between two classes.

26) What is lambda expression ?

A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

27) Difference between String object and string literal ?

 String Literal is a String created using double quotes while String Object is a String created using the new() operator.

28) When SCP object is destroy ?

Java (and JVM in particular) uses automatic garbage collection. To put it simply, whenever new objects are created, the memory is automatically allocated for them. Consequently, whenever the objects are not referenced anymore, they are destroyed and their memory is reclaimed.

29) How we can implement multithreading in java ?

i) By using thread class

ii) By using runnable interface

In case of thread class it is not compulsory to override the run method because in thread class run method is already defined but in case of implementation runnable it is compulsory to override run method bcz in runnable interface it is not defined.

30) Why run method is mendatory ?

To assign the job to the thread.

run(){

}

31) When to use thread class and when to use runnable interface ? why this two options are given in java ?

Thread class does not support multilevel inheritance so if you want use multilevel inheritance use runnable interface otherwise use thread class .

32) When data Inconsistency problem arise , and how we can solve it ?

Data inconsistency problem arise when multiple threads executing same time ,

We can solve it by using one thread at a time mechanism.

33) Why java does not support multiple inheritance and support multiple interface ?

In case of multiple inheritance ambiguity error occurs because compiler will confuse between multiple parent class method.

And in case of interface there is only declared method it is defined in child class so the always runs child class method it means here no confusion.