Q1. Can an abstract class have static methods in Java?

Answer: Yes, static methods can be declared in an abstract class. These methods belong to the class and not to instances of the class.

```
abstract class Example {
    static void display() {
        System.out.println("Static method in an abstract class");
    }
}
```

Q2. What is the difference between an abstract class and a concrete class?

Answer:

- Abstract Class: Cannot be instantiated and may have abstract methods.

- Concrete Class: Fully implemented class that can be instantiated.

Q3. Can a constructor be declared as final in Java?

Answer:

No, constructors cannot be declared as final because they are not inherited.

Q4. Can an abstract class have a constructor in Java?

Answer: Yes, abstract classes can have constructors to initialize class variables.

```
abstract class Shape {
    int sides;
    Shape(int sides) {
        this.sides = sides;
    }
}
```

Q5. Can we make an abstract class final in Java?

Answer:

No, an abstract class cannot be declared final because it is meant to be sub classed.

Q6. What is an interface in Java?

Answer:

An interface is a reference type in Java that is similar to a class but can contain only abstract

methods (before Java 8) and constants.

Q7. Can an interface extend another interface in Java?

Answer:

Yes, an interface can extend another interface.

```
interface A {
    void methodA();
}
interface B extends A {
    void methodB();
}
```

Q8. Can an interface implement another interface in Java?

Answer:

No, interfaces cannot implement other interfaces, but they can extend them.

Q9. Can a class implement multiple interfaces in Java?

Answer:

Yes, a class can implement multiple interfaces.

```java
interface A {
    void methodA();
}
interface B {
    void methodB();
}
class Example implements A, B {
    public void methodA() {
        System.out.println("Method A");
    }
    public void methodB() {
        System.out.println("Method B");
    }
}
```

Q10. Can an interface have a constructor in Java?

Answer:

No, interfaces cannot have constructors.

Q11. What is a HashSet in Java?

Answer:

A HashSet is a collection that does not allow duplicate elements and is backed by a HashMap.

Q12. What is the purpose of the Comparator and Comparable interfaces in Java?

Answer:

Comparable: Used to define natural ordering.

Comparator: Used to define custom ordering.

Q13. What is the difference between shallow copy and deep copy?

Answer:

Shallow Copy: Copies the references of objects.

Deep Copy: Creates a new instance of each object.

Q14. What is reflection in Java?

Answer:

Reflection is a feature that allows inspection and manipulation of classes, methods, and fields at runtime.

Q15. What is the volatile keyword in Java?

Answer:

The volatile keyword ensures visibility of changes to variables across threads.

Q16. Why String Buffer and StringBuilder classes are introduced in Java when there already exist String class to represent the set of characters?

Answer:

The objects of String class are immutable in nature. i.e. you can't modify them once they are created. If you try to modify them, a new object will be created with modified content. This may cause memory and performance issues if you are performing lots of string modifications in your code. To overcome these issues, String Buffer and String Builder classes are introduced in Java.

Q17. How many objects will be created in the following code and where they will be stored in the memory?

String s1 = new String("abc");

String s2 = new String("abc");

Answer:

Two objects will be created and they will be stored in the heap memory.

Q18. How do you create mutable string objects in Java?

Answer: Using String Buffer and StringBuilder classes. These classes provide mutable string objects.

Q19. Which one will you prefer among "==" and equals () method to compare two string objects?

Answer: I prefer equals () method because it compares two string objects based on their content. That provides more logical comparison of two string objects. If you use "==" operator, it checks only the references of two objects not their content. It may not be suitable in all situations. So, rather stick to equals () method to compare two string objects.

Q20. Which is the final class in these three classes – String, String Buffer and StringBuilder?

Answer: All three are final.

Q21. What do you mean by mutable and immutable objects?

Answer: Immutable objects are like constants. You can't modify them once they are created. They are final in nature. Whereas mutable objects are concerned, you can perform modifications on them.

Q22. Is String a keyword in Java?

Answer: No. String is not a keyword in Java. String is a final class in java.lang package which is used to represent the set of characters in Java.

Q23. Is String a primitive type or derived type?

Answer: String is a derived type.

Q24. What is string constant pool?

Answer: String objects are most used data objects in Java. Hence, Java has a special arrangement to store the string objects. String Constant Pool is one such arrangement. String Constant Pool is the memory space in the heap memory specially allocated to store the string objects created using string literals. In String Constant Pool, there will be no two string objects having the same content.

Whenever you create a string object using string literal, JVM first checks the content of the object to be created. If there exist an object in the string constant pool with the same content, then it returns the reference of that object. It doesn't create a new

object. If the content is different from the existing objects then only it creates new object.

Q25. What do you think about string constant pool? Why they have provided this pool as we can store string objects in the heap memory itself?

Answer: String constant pool increases the reusability of existing string objects. When you are creating a string object using string literal, JVM first checks string constant pool. If that object is available in string constant pool, it returns reference of that object rather than creating a new object. This will speed up your application as only reference is returned. And it also saves the memory as no two objects with same content are created.

Q26. What is the similarity and difference between String and StringBuffer class?

Answer: The main similarity between *String* and *StringBuffer* class is that both are thread safe. The main difference between them is that *String* objects are immutable where as *StringBuffer* objects are mutable.

Q27. What is the similarity and difference between StringBuffer and StringBuilder class?

Answer. The main similarity between *StringBuffer* and *StringBuilder* class is that both produces mutable string objects. The main difference between them is that *StringBuffer* class is thread safe whereas *StringBuilder* class is not thread safe.

Q28. What are the new String methods introduced in Java 11?

Answer. isBlank(), lines(), repeat(), strip(), stripLeading() and stripTrailing() are the new methods introduced to String class in Java 11.

Q29. What is method signature? What are the things it consists of?

Answer. Method signature is used by the compiler to differentiate the methods. Method signature consist of three things.

a) Method name

b) Number of arguments

c) Types of arguments

Q30. Can we declare one overloaded method as static and another one as non-static?

Answer. Yes. Overloaded methods can be either static or non-static.

Q31. How does compiler differentiate overloaded methods from duplicate methods?

Answer. Compiler uses method signature to check whether the method is overloaded or duplicated. Duplicate methods will have same method signatures i.e. same name, same number of arguments and same types of arguments. Overloaded methods will also have same name but differ in number of arguments or else types of arguments.

Q32. Is it possible to have two methods in a class with same method signature but different return types?

Answer. No, compiler will give duplicate method error. Compiler checks only method signature for duplication not the return types. If two methods have same method signature, straight away it gives compile time error.

Q33. What is JIT compiler?

Answer. The Just-In-Time (JIT) compiler is a part of the JVM that improves performance by compiling bytecode into native machine code at runtime.

Q34. What is bytecode in Java?

Answer. Bytecode is the intermediate representation of a Java program that the JVM interprets. It is platform-independent.

Q35. Difference between this() and super() in Java?

Answer:

this(): Calls another constructor in the same class.

super(): Calls a constructor of the parent class.

Q36. What is a class?

Answer:

A class is a blueprint for creating objects that encapsulates data (fields) and methods.

Q37. What is an object?

Answer:

An object is an instance of a class with state and behaviour.

Q38. What is a method in Java?

Answer:

A method is a block of code that performs a specific task. It can accept parameters and return values.

Q39. What is encapsulation?

Answer: Encapsulation is the bundling of data and methods within a class and restricting access through access modifiers.

Q40. Why is main() method public, static, and void in Java?

Answer:

public: Accessible to the JVM.

static: No need to create an object to call it.

void: Does not return a value.

Q41. Where the arrays are stored in the memory?

Answer: Arrays are nothing but the objects in Java. Hence, they are stored in heap memory like normal objects.

Q42 What are the different ways of copying an array into another array?

Answer: There are four ways available in Java to copy an array.

1) Using for loop

2) Using Arrays.copyOf() method

3) Using System.arraycopy() method

4) Using clone() method

Q43. How do you check the equality of two arrays in Java?

Answer: You can use Arrays.equals() method to compare one dimensional arrays and to compare multidimensional arrays, use Arrays.deepEquals() method.

Q44. What is ArrayIndexOutOfBoundsException in Java? When it occurs?

Answer: ArrayIndexOutOfBoundsException is a run time exception which occurs when your program tries to access invalid index of an array i.e negative index or index higher than the size of an array.

Q45. How do you search an array for a specific element?

Answer: You can search an array to check whether it contains the given element or not using Arrays.binarySearch() method. This method internally uses binary search algorithm to search for an element in an array.

Q46. What value does array elements get, if they are not initialized?

Answer: They get default values.

Q47. What are Packages in Java?

Answer: Packages in Java can be defined as the grouping of related types of classes, interfaces, etc providing access to protection and namespace management.

Q48. Why Packages are used?

Answer: Packages are used in Java in order to prevent naming conflicts, control access, and make searching/locating and usage of classes, interfaces, etc easier.

Q49. What are the advantages of Packages in Java?

Answer: There are various advantages of defining packages in Java.

- Packages avoid name clashes.
- The Package provides easier access control.
- We can also have the hidden classes that are not visible outside and are used by the package.
- It is easier to locate the related classes.

Q50. How many types of packages are there in Java?

Answer: There are two types of packages in Java

- User-defined packages
- Build In packages

Q51. Explain different data types in Java.

Answer: There are 2 types of data types in Java as mentioned below:

- Primitive Data Type
- Non-Primitive Data Type or Object Data type

Primitive Data Type: Primitive data are single values with no special capabilities. There are 8 primitive data types:

- boolean: stores value true or false
- byte: stores an 8-bit signed two's complement integer
- char: stores a single 16-bit Unicode character
- short: stores a 16-bit signed two's complement integer
- int: stores a 32-bit signed two's complement integer
- long: stores a 64-bit two's complement integer
- float: stores a single-precision 32-bit IEEE 754 floating-point
- double: stores a double-precision 64-bit IEEE 754 floating-point

Non-Primitive Data Type: Reference Data types will contain a memory address of the variable's values because it is not able to directly store the values in the memory. Types of Non-Primitive are mentioned below:

- Strings
- Array
- Class

- Object
- Interface

Q52. When a byte datatype is used?

Answer: A byte is an 8-bit signed two-complement integer. The minimum value supported by bytes is -128 and 127 is the maximum value. It is used in conditions where we need to save memory and the limit of numbers needed is between -128 to 127.

Q53. Can we declare Pointer in Java?

Answer: No, Java doesn't provide the support of Pointer. As Java needed to be more secure because which feature of the pointer is not provided in Java.

Q54. What is the default value of byte datatype in Java?

Answer: The default value of the byte datatype in Java is 0.

Q55. What is the default value of float and double datatype in Java?

Answer: The default value of the float is 0.0f and of double is 0.0d in Java.

Q56. What is the Wrapper class in Java?

Answer: Wrapper, in general, is referred to a larger entity that encapsulates a smaller entity. Here in Java, the wrapper class is an object class that encapsulates the primitive data types.

The primitive data types are the ones from which further data types could be created. For example, integers can further lead to the construction of long, byte, short, etc. On the other hand, the string cannot, hence it is not primitive.

Getting back to the wrapper class, Java contains 8 wrapper classes. They are Boolean, Byte, Short, Integer, Character, Long, Float, and Double. Further, custom wrapper classes can also be created in Java which is similar to the concept of Structure in the C programming language. We create our own wrapper class with the required data types.

Q57. Why do we need wrapper classes?

Answer: The wrapper class is an object class that encapsulates the primitive data types, and we need them for the following reasons:

- Wrapper classes are final and immutable
- Provides methods like valueOf(), parseInt(), etc.
- It provides the feature of autoboxing and unboxing.

Q58. What is the default value stored in Local Variables?

Answer: There is no default value stored with local variables. Also, primitive variables and objects don't have any default values.

Q59. Explain the difference between instance variable and a class variable.

Answer: **Instance Variable:** A class variable without a static modifier known as an instance variable is typically shared by all instances of the class. These variables can have distinct values among several objects. The contents of an instance variable are completely independent of one object instance from another because they are related to a specific object instance of the class.
Class Variable: Class Variable can be declared anywhere at the class level using the keyword static. These variables can only have one value when applied to various objects. These variables can be shared by all class members since they are not connected to any specific object of the class.

Q60. Which Java operator is right associative?

Answer: There is only one operator which is right associative which is = operator.

Q61. What is dot operator?

Answer: The Dot operator in Java is used to access the instance variables and methods of class objects. It is also used to access classes and sub-packages from the package.

Q62. What is covariant return type?

Answer: The covariant return type specifies that the return type may vary in the same direction as the subclass. It's possible to have different return types for an overriding method in the child class, but the child's return type should be a subtype of the parent's return type and because of that overriding method becomes variant with respect to the return type.

We use covariant return type because of the following reasons:

- Avoids confusing type casts present in the class hierarchy and makes the code readable, usable, and maintainable.
- Gives liberty to have more specific return types when overriding methods.
- Help in preventing run-time ClassCastExceptions on returns.

Q63. What is the transient keyword?

Answer: The transient keyword is used at the time of serialization if we don't want to save the value of a particular variable in a file. When JVM comes across a transient keyword, it ignores the original value of the variable and saves the default value of that variable data type.

Q64. What is the Java Collections Framework?

Answer: The Java Collections Framework (JCF) provides a set of classes and interfaces to manage a group of objects as a single unit. It includes various data structures like List, Set, Map, and utility classes like Collections for operations like sorting and searching.

```
List<String> list = new ArrayList<>();
list.add("A");
list.add("B");
System.out.println(list); // Output: [A, B]
```

Q65. What are the main interfaces in the Collections Framework?

Answer:

1. **List**: Ordered collection (e.g., ArrayList, LinkedList).
2. **Set**: Unordered collection of unique elements (e.g., HashSet, TreeSet).
3. **Queue**: Follows FIFO order (e.g., PriorityQueue).
4. **Map**: Key-value pairs (e.g., HashMap, TreeMap).

Q66. What is Iterable interface?

Answer:

Iterable interface is a member of java.lang package which is extended by java.util.Collection interface which is nothing but the root level interface of the Java collection framework. Iterable interface has only one method called iterator() which returns an Iterator object, using that object you can iterate over the elements of Collection. (forEach() and spliterator() methods are added to this interface from Java 8). That means these methods will be available in all collection types which are inherited from Collection interface.

Q67. What are the characteristics of List?

Answer:

- List Interface represents an ordered or sequential collection of objects.
- Elements of the lists are ordered using Zero based index.
- Elements of the lists can be randomly accessed. i.e. elements can be inserted at or removed from or retrieved from a specific position using integer index.
- A list may contain duplicate elements.
- A list may have multiple null elements.

Q68. What are the major implementations of List interface?

Answer:

- ArrayList
- Vector
- LinkedList

Q69. What are the characteristics of ArrayList?

Answer:

- Size of the ArrayList is not fixed. It can increase and decrease dynamically as we add or delete the elements.
- Elements are placed according to Zero-based index. That means, first element will be placed at index 0 and last element at index n-1, where 'n' is the size of the ArrayList.
- ArrayList can have any number of null elements.
- ArrayList can have duplicate elements.
- As ArrayList implements RandomAccess, you can get, set, insert and remove elements of the ArrayList from any arbitrary position.
- ArrayList is not synchronized. That means, multiple threads can use same ArrayList simultaneously.

Q70. What are the three marker interfaces implemented by ArrayList?

Answer:

- RandomAccess, Cloneable and Serializable.

Q71. What is the default initial capacity of ArrayList?

Answer:

- Default initial capacity of an ArrayList is 10. This capacity increases automatically as we add more elements to ArrayList. You can also specify initial capacity of an ArrayList while creating it.

Q72. What is the PriorityQueue?

Answer:

- PriorityQueue is a class in Java collection framework which implements Queue interface.
- The PriorityQueue is a queue in which elements are ordered according to specified Comparator. You have to specify this Comparator while creating a PriorityQueue itself. If no Comparator is specified, elements will be placed in their natural order.
- The PriorityQueue is a special type of queue because it is not a First-In-First-Out (FIFO) as in the normal queues. But, elements are placed according to supplied Comaparator.
- The PriorityQueue does not allow null elements. Elements in the PriorityQueue must be of Comparable type, If you insert the elements which are not Comparable, you will get ClassCastException at run time.
- The head element of the PriorityQueue is always the least element and tail element is always the largest element according to specified Comparator.

Q73. What are Deque and ArrayDeque? When they are introduced in Java?

Answer:

- Deque is an interface which extends the Queue interface and ArrayDeque is the class which implements Deque interface. Both are introduced from Java 6.
- The Deque is the short name for “Double Ended Queue“. As the name suggest, Deque is a linear collection of objects which supports insertion and removal of elements from both the ends. The Deque interface defines the

methods needed to insert, retrieve and remove the elements from both the ends.

- The main advantage of Deque is that you can use it as both **Queue** (FIFO) as well as **Stack** (LIFO). The Deque interface has all those methods required for FIFO and LIFO operations. ArrayDeque class provides implementations for all these methods.

Q 74. What are the characteristics of sets?

Answer:

- Set contains only unique elements. It does not allow duplicates.
- Set can have maximum one null element.
- Random access of elements is not possible.
- Order of elements in a set is implementation dependent. HashSet maintains no order. TreeSet elements are ordered according to supplied Comparator (If no Comparator is supplied, elements will be placed in their natural order) and LinkedHashSet maintains insertion order.
- Set interface contains only methods inherited from Collection interface. It does not have it's own methods. But, applies restriction on methods so that duplicate elements are always avoided.
- One more good thing about Set interface is that the stronger contract between equals() and hashCode() methods. According to this contract, you can compare two Set instances of different implementation types (HashSet, TreeSet and LinkedHashSet).
- Two set instances, irrespective of their implementation types, are said to be equal if they contain same elements.

Q75. What are the major implementations of Set interface?

Answer:

There are three major implementations of Set interface.

- HashSet
- LinkedHashSet
- TreeSet

Q76. What are the differences between List and Set?

Answer:

List	Set
List can have duplicate elements.	Set doesn't allow duplicate elements. It allows only unique elements.
List elements are ordered according zero-based index.	Order of elements in a set is implementation dependent. HashSet maintains no order. TreeSet elements are ordered according to supplied Comparator (If no Comparator is supplied, elements will be placed in their natural ascending order) and LinkedHashSet maintains insertion order.
List can have any number of null elements.	Set can have maximum one null element.
List elements can be accessed randomly.	Set elements can't be accessed randomly.
Ex : ArrayList, LinkedList	Ex : HashSet, LinkedHashSet, TreeSet

Q77.What are the characteristics of LinkedHashSet?

Answer:

- LinkedHashSet internally uses LinkedHashMap to store it's elements just like HashSet which internally uses HashMap to store it's elements.
- LinkedHashSet maintains insertion order. This is the main difference between LinkedHashSet and HashSet.
- LinkedHashSet also gives constant time performance for insertion, removal and retrieval operations. The performance of LinkedHashSet is slightly less than the HashSet as it has to maintain linked list internally to order it's elements.
- LinkedHashSet doesn't allow duplicate elements and allows maximum one null element.
- Iterator returned by LinkedHashSet is fail-fast. i.e if the LinkedHashSet is modified at any time after the Iterator is created, it throws ConcurrentModificationException.
- LinkedHashSet is not synchronized. To get the synchronized LinkedHashSet, use Collections.synchronizedSet() method.

Q78. When you prefer LinkedHashSet over HashSet?

Answer:

LinkedHashSet is preferred over HashSet if you want a unique collection of objects in an insertion order.

Q79. What is SortedSet? Give one Example?

Answer:

The SortedSet is an interface which extends Set interface. It's elements are sorted, that's why name SortedSet. The elements of the SortedSet are sorted according to supplied Comparator. This Comparator is supplied while creating a SortedSet. If you don't supply Comparator, elements will be placed in their natural order.

TreeSet is the SortedSet.

Q80. What is NavigableSet? Give one example?

Answer:

- The NavigableSet is an interface which extends SortedSet interface which in turn extends Set interface.
- The NavigableSet is a SortedSet with navigation facilities.
- The NavigableSet interface provides many methods through them you can easily find closest matches of any given element. It has the methods to find out less than, less than or equal to, greater than and greater than or equal of any element in a SortedSet.
- TreeSet is also of type NavigableSet.

Q81. What are the characteristics of TreeSet?

Answer:

- The elements in TreeSet are sorted according to specified Comparator. If no Comparator is specified, elements will be placed according to their natural ascending order.
- Elements inserted in the TreeSet must be of Comparable type and elements must be mutually comparable. If the elements are not mutually comparable, you will get ClassCastException at run time.
- TreeSet does not allow even a single null element.
- TreeSet is not synchronized. To get a synchronized TreeSet, use Collections.synchronizedSortedSet() method.
- TreeSet gives performance of order log(n) for insertion, removal and retrieval operations.
- Iterator returned by TreeSet is of fail-fast nature. That means, If TreeSet is modified after the creation of Iterator object, you will get ConcurrentModificationException.

- TreeSet internally uses TreeMap to store it's elements just like HashSet and LinkedHashSet which use HashMap and LinkedHashMap respectively to store their elements.

Q82. How HashSet, LinkedHashSet and TreeSet differ from each other?

Answer:

HashSet	LinkedHashSet	TreeSet
HashSet uses HashMap internally to store it's elements.	LinkedHashSet uses LinkedHashMap internally to store it's elements.	TreeSet uses TreeMap internally to store it's elements.
HashSet doesn't maintain any order of elements.	LinkedHashSet maintains insertion order of elements. i.e elements are placed as they are inserted.	TreeSet orders the elements according to supplied Comparator. If no Comparator is supplied, elements will be placed in their natural ascending order.
HashSet gives better performance than the LinkedHashSet and TreeSet.	The performance of LinkedHashSet is between HashSet and TreeSet. It's performance is almost similar to HashSet. But slightly in the slower side as it also maintains LinkedList internally to maintain the insertion order of elements.	TreeSet gives less performance than the HashSet and LinkedHashSet as it has to sort the elements after each insertion and removal operations.
HashSet gives performance of order O(1) for insertion, removal and retrieval operations.	LinkedHashSet also gives performance of order O(1) for insertion, removal and retrieval operations.	TreeSet gives performance of order O(log(n)) for insertion, removal and retrieval operations.
HashSet uses equals() and hashCode() methods to compare the elements and thus removing the possible duplicate elements.	LinkedHashSet also uses equals() and hashCode() methods to compare the elements.	TreeSet uses compare() or compareTo() methods to compare the elements and thus removing the possible duplicate elements. It doesn't use equals() and hashCode() methods for comparison of elements.
HashSet allows maximum one null element.	LinkedHashSet also allows maximum one null element.	TreeSet doesn't allow even a single null element. If you try to insert null element into TreeSet, it throws NullPointerException.

Q83. How Map interface is different from other three primary interfaces of Java collection framework – List, Set and Queue?

Answer: The main difference between Map interface and other three top level interfaces is that it doesn't inherit from Collection interface. Instead it starts it's own interface hierarchy for maintaining the key-value associations.

Map stores the data as key-value pairs where each key is associated with a value where as other three interfaces – List, Set and Queue – store only values.

Q84. What are the popular implementations of Map interface?

Answer:

- HashMap
- LinkedHashMap
- TreeMap

Q85. What are the characteristics of HashMap?

Answer:

- HashMap holds the data in the form of key-value pairs where each key is associated with one value.
- HashMap doesn't allow duplicate keys. But it can have duplicate values.
- HashMap can have multiple null values and only one null key.
- HashMap is not synchronized. To get the synchronized HashMap, use Collections.synchronizedMap() method.
- HashMap maintains no order.
- HashMap gives constant time performance of O(1) for the operations like get() and put() methods.
- Default initial capacity of HashMap is 16.

Q86. What is the blank final field?

Answer:

Uninitialized final field is called blank final field.

Q87. Can we change the state of an object to which a final reference variable is pointing?

Answer:

Yes, we can change the state of an object to which a final reference variable is pointing, but we can't re-assign a new object to this final reference variable.

Q88. What is the main difference between abstract methods and final methods?

Answer:

Abstract methods must be overridden in the sub classes and final methods are not at all eligible for overriding.

Q89. What is the use of final class?

Answer:

A final class is very useful when you want a high level of security in your application. If you don't want inheritance of a particular class, due to security reasons, then you can declare that class as a final.

Q90. Can we change the value of an interface field? If not, why?

Answer:

No, we can't change the value of an interface field. Because interface fields, by default, are final and static. They remain constant for whole execution of a program.

Q91. Where all we can initialize a final non-static global variable if it is not initialized at the time of declaration?

Answer:

In all constructors or in any one of instance initialization blocks.

Q92. What are the differences between HashSet and HashMap?

Answer:

HashSet	HashMap
HashSet implements Set interface.	HashMap implements Map interface.
HashSet stores the data as objects.	HashMap stores the data as key-value pairs.
HashSet internally uses HashMap.	HashMap internally uses an array of *Entry<K, V>* objects.
HashSet doesn't allow duplicate elements.	HashMap doesn't allow duplicate keys, but allows duplicate values.
HashSet allows only one null element.	HashMap allows one null key and multiple null values.
Insertion operation requires only one object.	Insertion operation requires two objects, key and value.
HashSet is slightly slower than HashMap.	HashMap is slightly faster than HashSet.

Q93. What are the differences between HashMap and HashTable?

Answer:

HashMap	HashTable
HashMap is not synchronized and therefore it is not thread safe.	HashTable is internally synchronized and therefore it is thread safe.
HashMap allows maximum one null key and any number of null values.	HashTable doesn't allow null keys and null values.
Iterators returned by the HashMap are fail-fast in nature.	Enumeration returned by the HashTable are fail-safe in nature.
HashMap extends AbstractMap class.	HashTable extends Dictionary class.
HashMap returns only iterators to traverse.	HashTable returns both Iterator as well as Enumeration for traversal.
HashMap is fast.	HashTable is slow.
HashMap is not a legacy class.	HashTable is a legacy class.
HashMap is preferred in single threaded applications. If you want to use HashMap in multi threaded application, wrap it using Collections.synchronizedMap() method.	Although HashTable is there to use in multi threaded applications, now a days it is not at all preferred. Because, ConcurrentHashMap is better option than HashTable.

Q94. Can we use a field or a method declared without access modifiers outside the package.?

Answer:

No, we can't use a field or a method with no-access (default) specifiers outside the package in which their class is defined.

Q95. Can a method or a class be final and abstract at the same time.?

Answer:

No, it is not possible. A class or a method cannot be final and abstract at the same time. final and abstract are totally opposite in nature. final class or final method must not be modified further whereas abstract class or abstract method must be modified further.

Q96. Can we declare a class as private.?

Answer:

We can't declare an outer class as private. But, we can declare an inner class (class as a member of another class) as private.

Q97. Can we declare an abstract method as private also.?

Answer: No, abstract methods cannot be private. They must be public or protected or default so that they can be modified further.

Q98. Can we declare a class as protected.?

Answer: We can't declare an outer class as protected. But, we can declare an inner class (class as a member of another class) as protected.

Q99. What are access modifiers in java.?

Answer: These are the modifiers which are used to restrict the visibility of a class or a field or a method or a constructor. Java supports 4 access modifiers.

a) private: private fields or methods or constructors are visible within the class in which they are defined.

b) protected: Protected members of a class are visible within the package but they can be inherited to sub classes outside the package.

c) public: public members are visible everywhere.

d) default or No-access modifiers: Members of a class which are defined with no access modifiers are visible within the package in which they are defined.

Q100. What are the types of inheritance.?

Answer: There are 5 types of inheritance.

1). Single Inheritance: One class is extended by only one class.

2). Multilevel Inheritance: One class is extended by a class and that class in turn is extended by another class thus forming a chain of inheritance.

3). Hierarchical Inheritance: One class is extended by many classes.

4). Hybrid Inheritance: It is a combination of above types of inheritance.

5). Multiple Inheritance: One class extends more than one classes. (Java does not support multiple inheritance.)

Q101. Can a class extend more than one classes or does java support multiple inheritance? If not, why?

Answer:

No, a class in java cannot extend more than one classes or java does not support multiple inheritance. To avoid ambiguity, complexity and confusion, java does not support multiple inheritance. For example, If Class C extends Class A and Class B which have a method with same name, then Class C will have two methods with same name. This causes ambiguity and confusion for which method to use. To avoid this, java does not support multiple inheritance.