1. Object Class Methods :

|  |  |
| --- | --- |
| **Method** | **Description** |
| 1. public final Class getClass() | returns the Class class object of this object. The Class class can further be used to get the metadata of this class. |
| 1. public int hashCode() | returns the hashcode number for this object. |
| 1. public boolean equals(Object obj) | compares the given object to this object. |
| 1. protected Object clone() throws CloneNotSupportedException | creates and returns the exact copy (clone) of this object. |
| 1. public String toString() | returns the string representation of this object. |
| 1. public final void notify() | wakes up single thread, waiting on this object's monitor. |
| 1. public final void notifyAll() | wakes up all the threads, waiting on this object's monitor. |
| 1. public final void wait(long timeout)throws InterruptedException | causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| 1. public final void wait(long timeout,int nanos)throws InterruptedException | causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| 1. public final void wait()throws InterruptedException | causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method). |
| 1. protected void finalize()throws Throwable | is invoked by the garbage collector before object is being garbage collected. |

1. hashcode() Method :

* It returns unique integer value for the object at runtime.
* That integer value is the memory address of that object in heap (It is not mandatory).
* It is used to determining the index of the object when the data stored into the some HashTable like datastructure.

1. equals() :

* It verifies the equality of the two objects.
* By default, two objects are equal if and only if they are refer to the same memory location.

1. Contract Between equals() and hashcode() Methods :

* **JAVA 8**

1. What is functional Interface?

Ans : i] Functional interface is an interface that contains only one abstract method.

ii] They can have only one functionality to exhibit.

iii] Runnable,ActionListner,comparable are some functional interface.

Iv] @FunctionalInterface is not mandatory to use.but it ensures that functional interface can’t more than one abstract method.

1. Why functional Interface is developed as there already have the abstract class is avaialable in java?

Ans : in case of abstract class – It allows you to create functionality that subclasses can implement or override.

In case of functional interfcae – It is introduce because objects are base of java programming language and we can never have a function without an object that’s why java language provide support for using lambda expression only with functional interface.

1. Difference between Stream and Collection ?

Ans :

| **STREAMS** | **COLLECTIONS** |
| --- | --- |
| It doesn’t store data, it operates on the souvrce data structure i.e collection. | It stores/holds all the data that the data structure currently has in a particular data structure like Set, List or Map, |
| They use functional interfaces like lambda which makes it a good fit for programming language. | They don’t use functional interfaces. |
| Java Streams are consumable i.e; to traverse the stream, it needs to be created every time. | They are non-consumable i.e; can be traversable multiple times without creating it again. |
| Java streams support both sequential and parallel processing. | It supports parallel processing and parallel processing can be very helpful in achieving high performance. |
| All the Java stream API interfaces and classes are in j**ava.util.stream** package. | Specific classes for primitive types such as **IntStream**, **LongStream**, and **DoubleStream** are used in collections since primitive data types such as int, long in the collections using auto-boxing and these operations could take a lot of time. |
| Streams are not modifiable i.e one can’t add or remove elements from streams. | These are modifiable i.e one can easily add to or remove elements from collections. |
| Streams are iterated internally by just mentioning the operations. | Collections are iterated externally using loops. |