class Object

clone(), equals(), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), and

wait()

class Object

- Class Object is the root of the class hierarchy.
- Object class is defined in java.lang package.
- Every class has Object as a superclass directly or indirectly.
- Direct Child: If a class does not extend any other class then it is a direct child class of Object class.
- Indirect Child: if extends another class then it is indirectly derived.
- Object class has one default constructor and several methods.

class Object

- clone()

  `Protected Object `**`clone`**`() throws CloneNotSupportedException`

- Object cloning refers to the creation of an exact copy of an object. It creates a new instance of the class of the current object and initializes all its fields with exactly the contents of the corresponding fields of this object.

class Object

- equals()

  ```
  public boolean equals(Object obj)
  ```

- The equals method implements the most discriminating possible equivalence relation on objects
- For any non-null reference values x and y, this method returns true if and only if x and y refer to the same object (x == y has the value true).
- Note that it is generally necessary to override the hashCode method whenever this method is overridden, so as to maintain the general contract for the hashCode method, which states that equal objects must have equal hash codes.

class Object

- finalize()

```
protected void finalize() throws
              Throwable
```

- Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
- A subclass can overrides the finalize method to dispose of system resources or to perform other cleanup.
- Finalize() is called **just before an object is garbage collected**. It is called the Garbage Collector on an object when the garbage collector determines that there are no more references to the object.

class Object

- Garbage collection in Java is the process by which Java programs perform automatic memory management.
- Java programs run on the JVM, objects are created on the **heap**, which is a portion of memory dedicated to the program.
- In C/C++, a programmer is responsible for both the creation and destruction of objects.
- In Java, the programmer need not care for all those objects which are no longer in use. Garbage collector destroys these objects.
- The main objective of Garbage Collector is to free heap memory by destroying unreachable objects.

class Object

- getClass()

  ```
  public final Class<?> getClass()
  ```

- It returns the class object of the object and is used to get the actual runtime class of the object.
- It can also be used to get metadata of this class.
- The returned Class object is the object that is locked by static synchronized methods of the represented class.
- As it is final so we don’t override it.

class Object

```
public class abc
{
  public static void main(String
  arg[]){
      abc o=new abc();
      Class oc=o.getClass();
      System.out.print(oc);
  }
}
```

class Object

```
public class abc
{
  public static void main(String
  arg[]){
     abc o=new abc();
     Class oc=o.getClass();
     System.out.print(oc);
  }
}
```

```
Output:
class abc
```

class Object

- hashCode()

```
public int hashCode()
```

- Returns a hash code value for the object.
- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer.
- This integer need not remain consistent from one execution of an application to another execution of the same application.

class Object

- hashCode()

```
public int hashCode()
```

- Returns a hash code value for the object.
- If two objects are equal according to the equals(Object) method, then calling the hashCode() on each of the two objects must produce the same integer result.
- It is not required that if two objects are unequal according to the equals(Object) method, then calling the hashCode() on each of the two objects must produce distinct integer results.

class Object

- notify()

```
public final void notify()
```

- Wakes up a single thread that is waiting on this object's monitor.
- If any threads are waiting on this object, one of them is chosen to be awakened.
- The choice is arbitrary and occurs at the discretion of the implementation.
- A thread waits on an object's monitor by calling one of the wait methods.

class Object

- notify()

```
        public final void notify()
```

- The awakened thread will not be able to proceed until the current thread relinquishes the lock on this object.
- The awakened thread will compete in the usual manner with any other threads that might be actively competing to synchronize on this object..

class Object

- notifyAll()

```
public final void notifyAll()
```

- Wakes up all threads that are waiting on this object's monitor. A thread waits on an object's monitor by calling one of the wait methods..
- The awakened threads will compete in the usual manner with any other threads that might be actively competing to synchronize on this object.

class Object

- toString()

```
public String toString()
```

- Returns a string representation of the object.
- In general, the toString method returns a string that "textually represents" this object.
- It is recommended that all subclasses override this method.

class Object

- toString()

```
public String toString()
```

- The toString method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object.
- In other words, this method returns a string equal to the value of:

```
getClass().getName() + '@' +
Integer.toHexString(hashCode())
```

class Object

```
public class abc
{
  public static void main(String
  arg[]) {
      int a=10, b=20,c;
      c=a+b;
     System.out.print(c);
     System.out.println(c.toString());
  }
}
```

class Object

- wait()

```
public final void wait(long timeout)
      throws InterruptedException
```

- It causes the current thread to wait until either another thread invokes the notify() or notifyAll() for this object, or a specified amount of time has elapsed.
- timeout - the maximum time to wait in milliseconds.

class Object

- wait()

```
public final void wait(long timeout,
            int nanos) throws
        InterruptedException
```

- This method is similar to the wait method of one argument, but it allows finer control over the amount of time to wait for a notification before giving up.
- The amount of real time, measured in nanoseconds, is given by:

```
        1,000,000*timeout+nanos.
```

class Object

- wait()

```
public final void wait() throws
        InterruptedException
```

- It causes the current thread to wait until another thread invokes the notify() or notifyAll() method for this object.
- The specified amount of real time has elapsed, more or less.
- If timeout is zero, however, then real time is not taken into consideration and the thread simply waits until notified.

```
wait(0), wait(0,0) and wait()
```