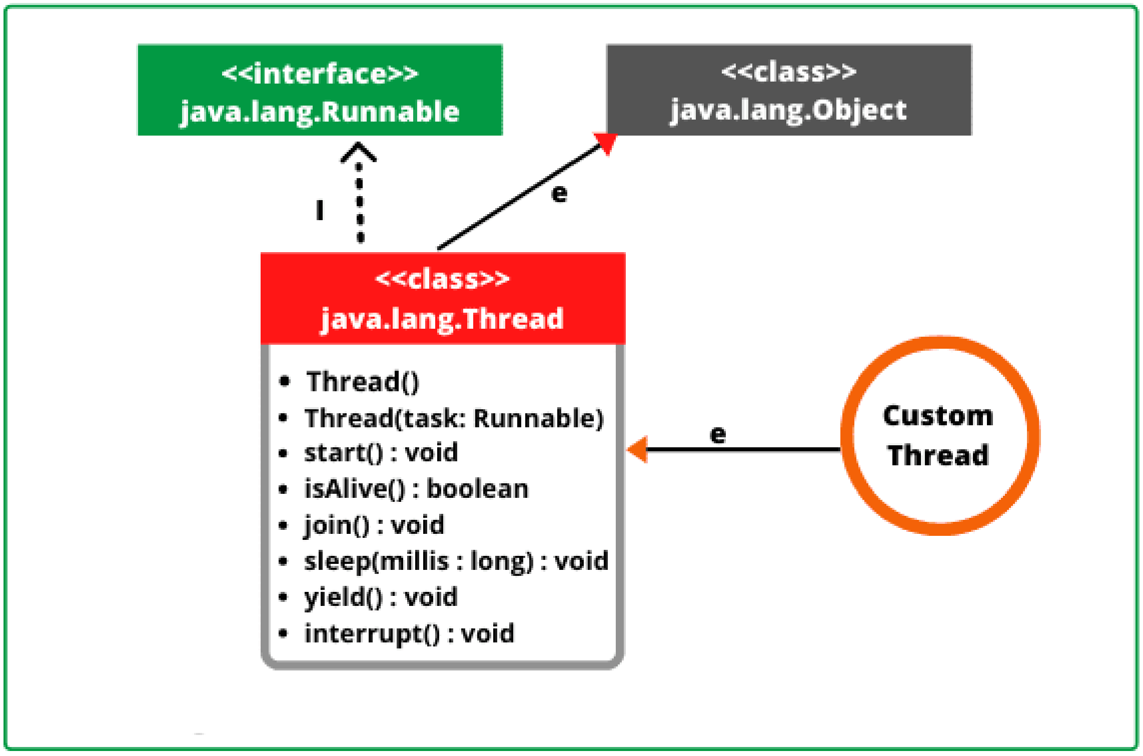
Thread Class in Java

In Java, Thread class contains several constructors for creating threads for tasks and methods for controlling threads. It is a predefined class declared in java.lang default package.

Each thread in Java is created and controlled by a unique object of the Thread class. An object of thread controls a thread running under JVM.

Thread class contains various methods that can be used to start, control, interrupt the execution of a thread, and for many other thread related activities in a program.



Thread Class Constructor

The various constructors of thread class are defined in java.lang package that can be used to create
 an object of thread .

1. **Thread():** This is a basic and default constructor without parameters. It simply creates an object of Thread class.

2. **Thread(String name):** It creates a new thread object with specified name to a thread.

3. **Thread(Runnable r):**It creates a thread object by passing a parameter r as a reference to
an object of the class that implements Runnable interface.

4.**Thread(Runnable r, String name):**This constructor creates a thread object by passing two arguments r and name. Here, variable r is a reference of an object of class that
implements Runnable interface.

Methods of Thread Class

currentThread(): The currentThread() returns the reference of currently executing thread. Since this is a static method, so we can call it directly using the class name.

```
public static Thread currentThread()
```

sleep(): The sleep() method puts currently executing thread to sleep for specified number of milliseconds. This method is used to pause the current thread for specified amount of time in
milliseconds.

```
public static void sleep(long milliseconds) throws InterruptedException

public static void sleep(long milliseconds, int nanoseconds ) throw InterruptedException
```

yield(): The yield() method pauses the execution of current thread and allows another thread of equal or higher priority that are waiting to execute. Currently executing thread give up the
control of the CPU.

```
public static void yield()
```

activeCount(): This method returns the number of active threads.

```
public static int activeCount()
```

The instance methods of Thread class

start(): The start() method is used to start the execution of a thread by calling its run() method. JVM calls the run() method on the thread.

```
public void start()
```

run(): The run() method moves the thread into running state. It is executed only after the start() method has been executed.

```
public void run()
```

getName(): This method returns the name of the thread. The return type of this method is String.

```
public final String getName()
```

setName(): This method is used to set the name of a thread. It takes an argument of type String.

```
public final void setName(String name)
```

getPriority(): This method returns the priority of a thread. It returns priority in the form of an integer value ranging from 1 to 10. The maximum priority is 10, the minimum priority is 1, and
normal priority is 5

```
public final int getPriority() // Return type is an integer.
```

setPriority(): This method is used to set the priority of a thread. It accepts an integer value as an argument.

```
public final void setPriority(int newPriority)
```

isAlive(): This method is used to check the thread is alive or not. It returns a boolean value (true or false) that indicates thread is running or not. The isAlive() method is final and native.

```
public final native boolean isAlive()
```

join(): The join() method is used to make a thread wait for another thread to terminate its process.

```
public final void join() throw InterruptedException
```

stop(): This method is used to stop the thread.

```
public final void stop()
```

suspend(): The suspend() method is used to suspend or pause a thread.

```
public final void suspend()
```

resume(): This method is used to resume the suspended thread. It neither accepts anything nor returns anything.

```
public final void resume()
```

isDaemon(): This method is used to check the thread is daemon thread or not.

```
public final boolean isDaemon()
```