**1.What is the difference between implementing Runnable and extending Thread?**

**Difference between implementing Runnable and extending Thread.**

One difference between implementing Runnable and extending Thread is that by extending Thread, each of your threads has a unique object associated with it, whereas implementing Runnable,

many threads can share the same object instance.

A class that implements Runnable is not a thread and just a class.For a Runnable to become a Thread, You need to create an instance of Thread and passing itself in as the target.

In most cases, the Runnable interface should be used if you are only planning to override the run() method and no other Thread methods. This is important because classes should not be subclassed unless the programmer intends on modifying or enhancing the fundamental behavior of the class.

When there is a need to extend a superclass, implementing the Runnable interface is more appropriate than using the Thread class. Because we can extend another class while implementing Runnable interface to make a thread. But if we just extend the Thread class we can't inherit from any other class.

Monitor is an object or module intended to be used safely by more than one thread.

\*/

public class Program {  
 public static void main (String[] args) {  
 Runner r = new Runner();  
 Thread t1 = new Thread(r, "Thread A");  
 Thread t2 = new Thread(r, "Thread B");  
 Thread s1 = new Strider("Thread C");  
 Thread s2 = new Strider("Thread D");  
 t1.start();  
 t2.start();  
 s1.start();  
 s2.start();  
 }  
}  
  
class Runner implements Runnable {  
 private int counter;  
 public void run() {  
 try {  
 for (int i = 0; i != 2; i++) {  
 System.out.println(Thread.currentThread().getName() + ": "   
 + counter++);  
 Thread.sleep(1000);  
 }  
 }  
 catch(InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}  
  
class Strider extends Thread {   
 private int counter;  
 Strider(String name) {  
 super(name);  
 }  
 public void run() {  
 try {  
 for (int i = 0; i != 2; i++) {  
 System.out.println(Thread.currentThread().getName() + ": "   
 + counter++);  
 Thread.sleep(1000);  
 }  
 }  
 catch(InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

The output result is

Thread A: 0  
Thread B: 1  
Thread C: 0  
Thread D: 0  
Thread A: 2  
Thread B: 3  
Thread C: 1  
Thread D: 1

A class that implements *Runnable* is not a thread and just a class. For a *Runnable* to become a Thread, You need to create an instance of *Thread* and passing itself in as the target.

In most cases, the *Runnable* interface should be used if you are only planning to override the *run()* method and no other *Thread* methods. This is important because classes should not be subclassed unless the programmer intends on modifying or enhancing the fundamental behavior of the class.

When there is a need to extend a superclass, implementing the *Runnable* interface is more appropriate than using the *Thread* class. Because we can extend another class while implementing *Runnable* interface to make a thread. But if we just extend the *Thread* class we can't inherit from any other class.

Link:-<http://www.xyzws.com/Javafaq/what-is-the-difference-between-implementing-runnable-and-extending-thread/29>

2 Recommend Type…?

I would recommend using something like Runnable rather than Thread because it allows you to keep your work only loosely coupled with your choice of concurrency. For example, if you use aRunnable and decide later on that this doesn't in fact require it's own Thread, you can just call threadA.run().

3.**Difference between Thread vs Runnable interface in Java**

Thread vs Runnable in Java is always been a confusing decision for beginners in java. Thread in Java seems easy in comparison of Runnable because you just deal with one class java.lang.Thread while in case of using Runnable to implement Thread you need to deal with both Thread and Runnable two classes. though decision of using Runnable or Thread should be taken considering differences between Runnable and Thread and pros and cons of both approaches. This is also a very popular thread interview questions and most of interviewer are really interested to know what is your point of view while choosing Thread vs Runnable or opposite. In this java article we will try to point out some differences between Thread and Runnable in Java which will help you to take an informed decision.

**4.Difference between Thread and Runnable interface in Java**

**Thread vs Runnable in Java**

Difference between Thread vs Runnable in JavaHere are some of my thoughts on whether I should use Thread or Runnable for implementing task in Java, though you have another choice as "Callable" for implementing thread which we will discuss later.

1) Java doesn't support multiple inheritance, which means you can only extend one class in Java so once you extended Thread class you lost your chance and can not extend or inherit another class in Java.

2) In Object oriented programming extending a class generally means adding new functionality, modifying or improving behaviors. If we are not making any modification on Thread than use Runnable interface instead.

3) Runnable interface represent a Task which can be executed by either plain Thread or Executors or any other means. so logical separation of Task as Runnable than Thread is good design decision.

4) Separating task as Runnable means we can reuse the task and also has liberty to execute it from different means. since you can not restart a Thread once it completes. again Runnable vs Thread for task, Runnable is winner.

5) Java designer recognizes this and that's why Executors accept Runnable as Task and they have worker thread which executes those task.

6) Inheriting all Thread methods are additional overhead just for representing a Task which can can be done easily with Runnable.

These were some of notable difference between Thread and Runnable in Java, if you know any other differences on Thread vs Runnable than please share it via comments. I personally use Runnable over Thread for this scenario and recommends to use Runnable or Callable interface based on your requirement.

<http://javarevisited.blogspot.in/2012/01/difference-thread-vs-runnable-interface.html>

Read more: <http://javarevisited.blogspot.com/2012/01/difference-thread-vs-runnable-interface.html#ixzz2cb44BW4l>

**5.Java Threads via Extending Thread class, implementing Runnable interface and Callable interface**

If anybody asks - What are the two methods of using Threads in Java? The answer is on our tip of tongue- TWO, but further if it is asked what is the difference between threads implemented via extending Thread class and implementing runnable interface? we feel dificulty in explaining. Here some of my thoughts are presented to draw a line among them.

1. As we all know Java doesn't support multiple inheritance. If you want to extend the Thread class then it will make your class unable to extend other classes as java is having single inheritance feature whereas If you implement runnable interface, you can gain better object-oriented design and consistency and also avoid the single inheritance problems.
2. In OOP, we extend a class for adding some new behaviour. If we are not making any modification on Thread than its a good OOP design to use Runnable interface.
3. When we extend a thread class, all the Properties and Behaviours of Thread class and inherited by the subclass, its kind of overhead if we are not going to use it and the same can be achieved on ease with Runnable.
4. Runnable interface represent Task and the same gives flexibility to get them executed by either Executors or Thread or any other means. So it draws a line between Task as Runnable than Thread as good design decision. However Executors accept Runnable as Task and they have worker thread which executes those tasks, but that's a totally different point.
5. While we are differentiating task as Runnable exhibits it can be reused and we are also provided with liberty to execute it from different means. As we know, a thread cannot be restarted once it's complete so again Runnable seems more promising.

**A Short Note on Callable and Runnable**:

Both the interfaces are implemented by those classes who want to execute a thread of execution, but few notable differences are as follows:

1. A Thread implemented via Callable interface may return a value, whereas its not possible if we are using a Runnable interface.
2. A Thread implemented via Callable interface may throw checked exceptions, whereas its not possible if we are using a Runnable interface.

What I feel, the developers of Java felt a need of enhancing the capabilities of the Runnable interface, without changing the usage of the Runnable interface and that is why probably they've developed Callable interface.

<http://www.apoorvaprakash.in/2012/10/java-threads-via-extending-thread-class.html>