**Basic Java Multihreaing – 2022**

**How to define a thread in Java 8**

public String m1() {  
 try {  
 System.*out*.println("Executing m1 ...");  
 TimeUnit.*SECONDS*.sleep(5);  
 } catch (InterruptedException ie) {  
 ie.printStackTrace();  
 }  
 return "m1";  
}  
  
public void check() {  
 **Runnable r = () -> m1();  
 Thread th = new Thread(r); or Thread th = new Thread( () -> m1() );  
 th.start();**}

**How to use Executor properly**

public void check() {  
 ExecutorService executor = Executors.*newSingleThreadExecutor*();  
 executor.submit(() -> m1());  
 try {  
 System.*out*.println("attempt to shutdown executor");  
 executor.shutdown();  
 executor.awaitTermination(5, TimeUnit.*SECONDS*);  
 } catch (InterruptedException e) {  
 System.*err*.println("tasks interrupted");  
 } finally {  
 if (!executor.isTerminated()) {  
 System.*err*.println("cancel non-finished tasks");  
 }  
 executor.shutdownNow();  
 System.*out*.println("shutdown finished");  
 }  
}

**Executor with Callable in Java 8**

public String m1() {  
 try {  
 System.*out*.println("Executing m1 ...");  
 TimeUnit.*SECONDS*.sleep(5);  
 } catch (InterruptedException ie) {  
 ie.printStackTrace();  
 }  
 return "m1";  
}  
  
public String m2() {  
 try {  
 System.*out*.println("Executing m2 ...");  
 TimeUnit.*SECONDS*.sleep(3);  
 } catch (InterruptedException ie) {  
 ie.printStackTrace();  
 }  
 return "m2";  
}  
  
public String m3() {  
 try {  
 System.*out*.println("Executing m3 ...");  
 TimeUnit.*SECONDS*.sleep(2);  
 } catch (InterruptedException ie) {  
 ie.printStackTrace();  
 }  
 return "m3";  
}

All the above methods should return value.

public void check() throws InterruptedException {  
 ExecutorService executor = Executors.*newWorkStealingPool*();  
 List<Callable<String>> callables = Arrays.*asList*(() -> m1(), () -> m2(), () -> m3());  
 executor.invokeAll(callables)  
 .stream()  
 .map(future -> {  
 try {  
 return future.get();  
 } catch (Exception e) {  
 throw new IllegalStateException(e);  
 }  
 })  
 .forEach(System.*out*::println);  
}