<https://www.codejava.net/java-core/the-java-language/java-8-lambda-runnable-example>

**How to use Java Lambda expression to create thread via Runnable**

Lambda expressions (Project Lambda - [JSR 335](https://www.jcp.org/en/jsr/detail?id=335)) are a new and important feature of the upcoming Java SE 8 platform ([JSR 337](https://www.jcp.org/en/jsr/detail?id=337)). They can be used to represent one method interface (also known as *functional interface*) in a clear and concise way using an expression in the form of:

***(argument list) -> body***

In this article, we see how Lambda expressions can simplify the creation of a new thread.

## 1. Create a Java thread via Runnable using Classic Code

Before Java 8, we [create and start a thread](https://www.codejava.net/java-core/concurrency/how-to-use-threads-in-java-create-start-pause-interrupt-and-join) by creating an anonymous class that implements the Runnable interface, as shown in the following code:

Runnable task1 = new Runnable(){

    @Override

    public void run(){

        System.out.println("Task #1 is running");

    }

};

Thread thread1 = new Thread(task1);

thread1.start();

Or pass the anonymous class into the Thread’s constructor:

Thread thread1 = new Thread(new Runnable() {

    @Override

    public void run(){

        System.out.println("Task #1 is running");

    }

});

thread1.start();

## 2. Create a Java thread via Runnable using Lambda expression

With Lambda expressions come with Java 8, the above code can be re-written more concisely. For example:

// Lambda Runnable

Runnable task2 = () -> { System.out.println("Task #2 is running"); };

// start the thread

new Thread(task2).start();

It’s much more simple, isn’t it? By using Lambda expression, you don’t have to write the boilerplate code: declarations of the anonymous class and the **run()** method.

And the following code snippet is for test program that demonstrates creating threads using both classic and Lambda approaches:

package net.codejava.lambda;

/\*\*

 \* This simple program demonstrates how to use Lambda expressions to create

 \* and run threads.

 \*

 \* @author www.codejava.net

 \*/

public class RunnableLambdaExample {

    public static void main(String[] args) {

        System.out.println(Thread.currentThread().getName() + ": RunnableTest");

        // Anonymous Runnable

        Runnable task1 = new Runnable(){

          @Override

          public void run(){

            System.out.println(Thread.currentThread().getName() + " is running");

          }

        };

        // Passing a Runnable when creating a new thread

        Thread thread2 = new Thread(new Runnable() {

            @Override

            public void run(){

                System.out.println(Thread.currentThread().getName() + " is running");

            }

        });

        // Lambda Runnable

        Runnable task3 = () -> {

            System.out.println(Thread.currentThread().getName() + " is running");

        };

        Thread thread1 = new Thread(task1);

        thread1.start();

        thread2.start();

        new Thread(task3).start();

    }

}

Output of this program may be different on each run because there are 3 threads started simultaneously. For example:

**Run #1:**

main: RunnableTest

Thread-1 is running

Thread-0 is running

Thread-2 is running

**Run #2:**

main: RunnableTest

Thread-1 is running

Thread-2 is running

Thread-0 is running

<https://stackoverflow.com/questions/31416784/thread-with-lambda-expression>

Q:

I have an error at line 42 and 43 : Thread t1=new Thread(()->prod.test()); , Thread t2=new Thread(()->cons.test()); Unhandled exception type InterruptedException . If I try to quickfix it will created the try catch with an catch Exception, it will have the same error and will try to fix it in the same way continuing to surround it with try catch.

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

interface Predicate {

public void test() throws InterruptedException;

}

class MyClass {

int num = 0;

Lock lock = new ReentrantLock();

public void produce() throws InterruptedException {

lock.lock();

for (int i = 0; i < 1000; i++) {

num++;

Thread.sleep(1);

}

lock.unlock();

}

public void consume() throws InterruptedException {

lock.lock();

for (int i = 0; i < 1000; i++) {

num--;

Thread.sleep(1);

}

lock.unlock();

}

public int getNum() {

return num;

}

}

public class Main00 {

public static void main(String[] args) throws InterruptedException {

MyClass c = new MyClass();

Predicate prod = c::produce;

Predicate cons = c::consume;

Thread t1 = new Thread(() -> prod.test());

Thread t2 = new Thread(() -> cons.test());

long start = System.currentTimeMillis();

t1.start();

t2.start();

t1.join();

t2.join();

long end = System.currentTimeMillis();

System.out.println("time taken " + (end - start) + " num = "

+ c.getNum());

}

}

A:

You have created a functional interface Predicate whose method is declared to throw an InterruptedException, which is a checked exception. However, you call test() in the body of a lambda expression as the parameter to [the Thread constructor that takes a Runnable](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html#Thread-java.lang.Runnable-), whose [run() method is not declared to throw any checked exceptions](https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html#run--). Therefore, because the exception is not caught in the body, a compiler error occurs.

Incidentally, it may be confusing to name your own interface Predicate, because of the [built-in functional interface java.util.function.Predicate](https://docs.oracle.com/javase/8/docs/api/java/util/function/Predicate.html) whose functional method returns a boolean.

Because run() can't throw an Exception, you must catch the exception and handle it. You might log the exception and its stack trace. You might wrap the exception in a RuntimeException. Either way, catching the checked exception will allow the code to compile. Example:

Thread t1 = new Thread(() -> {

try {

prod.test();

} catch (InterruptedException e) {

// handle: log or throw in a wrapped RuntimeException

throw new RuntimeException("InterruptedException caught in lambda", e);

}

});