线程未捕获异常uncaughtException的处理

为什么需要UncauthtExceptionHandler?

- **reason1:主线程**可以轻松发现异常，而**子线程却不行**
    - 代码演示:

      ```java
      /**
       * @desc: 单线程，抛出，有异常堆栈
       *          多线程呢？子线程发生异常，会有什么不同？
       * @author: Mr.Han
       */
      public class ExceptionInChildThread implements Runnable{
          public static void main(String[] args) {
              new Thread(new ExceptionInChildThread()).start();
              for (int i = 0; i < 1000; i++) {
                  System.out.println(i);
              }
          }

          @Override
          public void run() {
              throw new RuntimeException();
          }
      }
      ```

      > 运行我们可以发现：虽然子线程可以抛出异常，但是主线程的运行丝毫不受影响，在实际的生产环境中我们其实很难在茫茫日志中发现打印出的子线程的异常信息，这样就很容易忽略子线程发生的异常。因此有了**UncauthtExceptionHandler**的登场。

      ps: 运行结果

      ```
      244
      245
      246
      Exception in thread "Thread-0" java.lang.RuntimeException
          at uncaughtexception.ExceptionInChildThread.run(ExceptionInChildThread.java:18)
          at java.lang.Thread.run(Thread.java:748)
      247
      248
      249
      250
      ```

    - **reason2: 子线程异常无法用传统方式捕获**
        - try-catch 只能捕获当前线程发生的异常。而当前线程（如main thread）中的子线程发生异常时无法捕获的。
        - 代码演示:

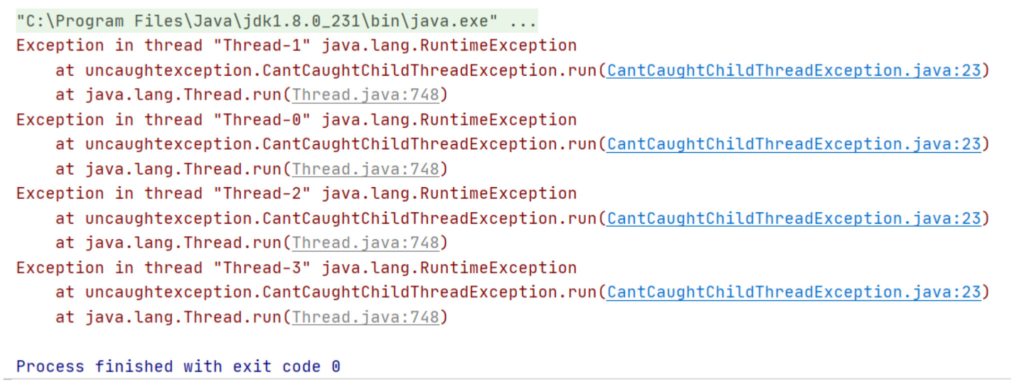
          ```java
          package uncaughtexception;

          /**
           * @desc: 演示通过try-catch 无法捕获子线程的异常
           * @author: Mr.Han
          ```

```java
 */
public class CantCaughtChildThreadException implements Runnable{
    public static void main(String[] args) {
        try {
            new Thread(new CantCaughtChildThreadException()).start();
            new Thread(new CantCaughtChildThreadException()).start();
            new Thread(new CantCaughtChildThreadException()).start();
            new Thread(new CantCaughtChildThreadException()).start();
        } catch (RuntimeException e){
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        throw new RuntimeException();
    }
}
```

运行结果



```
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
Exception in thread "Thread-1" java.lang.RuntimeException
    at uncaughtexception.CantCaughtChildThreadException.run(CantCaughtChildThreadException.java:23)
    at java.lang.Thread.run(Thread.java:748)
Exception in thread "Thread-0" java.lang.RuntimeException
    at uncaughtexception.CantCaughtChildThreadException.run(CantCaughtChildThreadException.java:23)
    at java.lang.Thread.run(Thread.java:748)
Exception in thread "Thread-2" java.lang.RuntimeException
    at uncaughtexception.CantCaughtChildThreadException.run(CantCaughtChildThreadException.java:23)
    at java.lang.Thread.run(Thread.java:748)
Exception in thread "Thread-3" java.lang.RuntimeException
    at uncaughtexception.CantCaughtChildThreadException.run(CantCaughtChildThreadException.java:23)
    at java.lang.Thread.run(Thread.java:748)

Process finished with exit code 0
```

解决方法

（一）在run 方法中 try - catch (不推荐)

（二）使用UncaughtExceptionHandler

- UncaughtExceptionHandler是Thread类内部的一个Interface，代码如下：

```java
@FunctionalInterface
public interface UncaughtExceptionHandler {
    /**
     * Method invoked when the given thread terminates due to the
     * given uncaught exception.
     * <p>Any exception thrown by this method will be ignored by
the
     * Java Virtual Machine.
     * @param t the thread
     * @param e the exception
     */
    void uncaughtException(Thread t, Throwable e);
}
```

- 异常处理器调用策略

  1. 实现自定义的UncaughtExceptionHandler

```java
package uncaughtexception;

import java.util.logging.Logger;

/**
 * @desc:
 * @author: Mr.Han
 */
public class MyOwnUncaughtException implements
Thread.UncaughtExceptionHandler {
    private String name;

    public MyOwnUncaughtException(){

    }

    public MyOwnUncaughtException(String name) {
        this.name = name;
    }

    @Override
    public void uncaughtException(Thread t, Throwable e) {
        Logger logger = Logger.getAnonymousLogger();
        logger.warning(t.getName() + "发生异常");
        System.out.println(this.name + "捕获了异常");
    }
}
```

  2. 测试

```java
package uncaughtexception;

/**
 * @desc: 演示通过try-catch 无法捕获子线程的异常
 * @author: Mr.Han
 */
public class UseMyOwnCaughtExceptionToCaughtException implements
Runnable{
    public static void main(String[] args) {
        Thread.setDefaultUncaughtExceptionHandler(new MyOwnUncaughtException("myOwnUncaughtException"));
        new Thread(new UseMyOwnCaughtExceptionToCaughtException()).start();
        new Thread(new UseMyOwnCaughtExceptionToCaughtException()).start();
        new Thread(new UseMyOwnCaughtExceptionToCaughtException()).start();
        new Thread(new UseMyOwnCaughtExceptionToCaughtException()).start();
    }

    @Override
    public void run() {
        throw new RuntimeException();
    }
}
```

3. 运行结果

```
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" "-javaagent:C:\Program gram core\target\classes" uncaughtexception.UseMyOwnCaughtExceptionToCaughtException
六月 27, 2020 11:56:28 上午 uncaughtexception.MyOwnUncaughtException uncaughtException
警告: Thread-3发生异常
六月 27, 2020 11:56:28 上午 uncaughtexception.MyOwnUncaughtException uncaughtException
警告: Thread-1发生异常
六月 27, 2020 11:56:28 上午 uncaughtexception.MyOwnUncaughtException uncaughtException
警告: Thread-2发生异常
六月 27, 2020 11:56:28 上午 uncaughtexception.MyOwnUncaughtException uncaughtException
警告: Thread-0发生异常
myOwnUncaughtException捕获了异常
myOwnUncaughtException捕获了异常
myOwnUncaughtException捕获了异常
myOwnUncaughtException捕获了异常
```