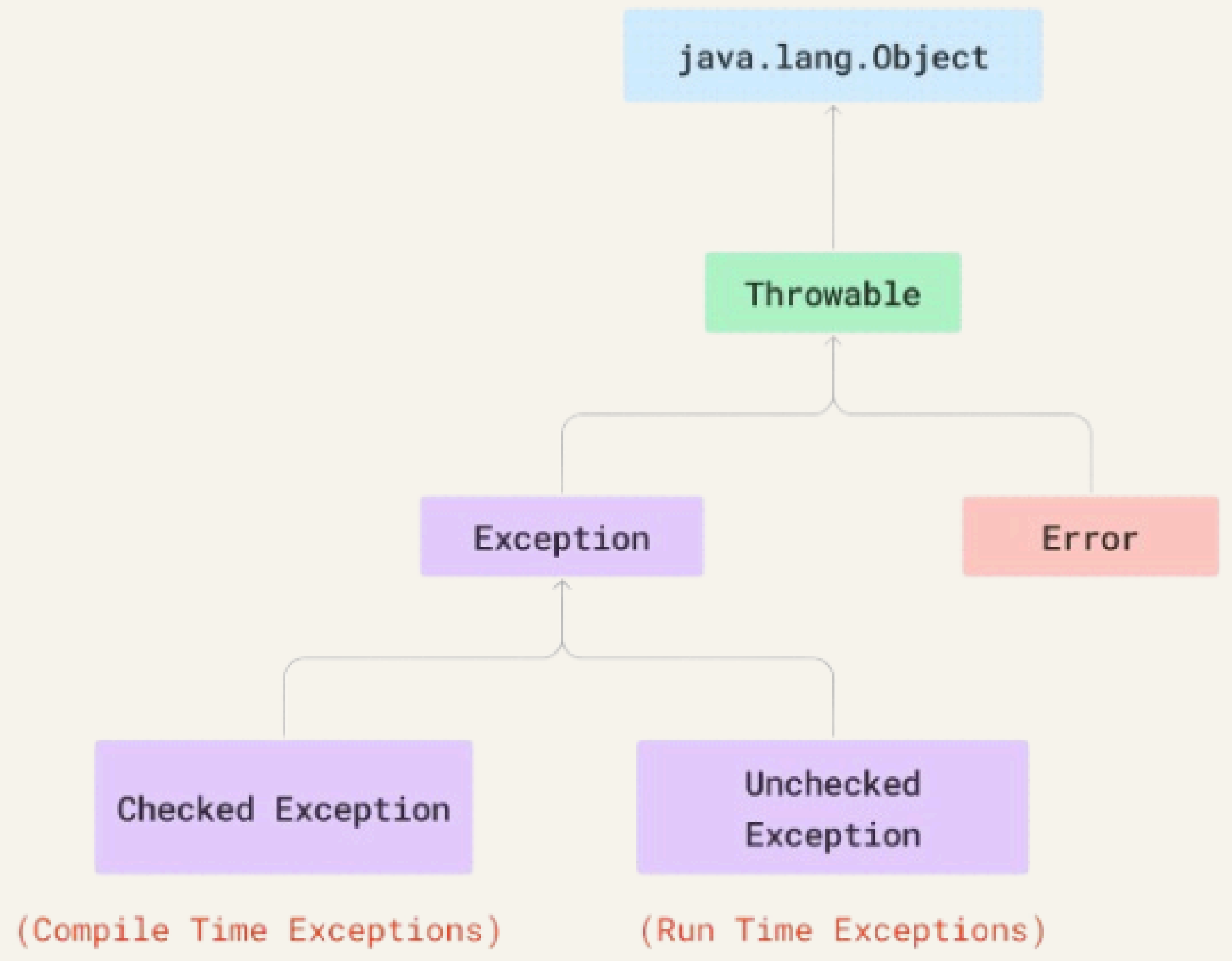
PARALLEL COMPUTATION

SECTION 4

ESLAM AHMED MOHAMED

INTRODUCTION TO EXCEPTIONS

- An Exception is an event that disrupts the normal flow of a program during execution.
- In Java, all exceptions are derived from the Throwable class, which has two main branches:
  - Exception – recoverable issues that can be handled.
  - Error – serious problems that usually should not be handled.



TYPES OF EXCEPTIONS

● Checked Exceptions

- These exceptions must be either handled using try-catch or declared using throws in the method signature.
- They represent errors that can be anticipated and recovered from during program execution.

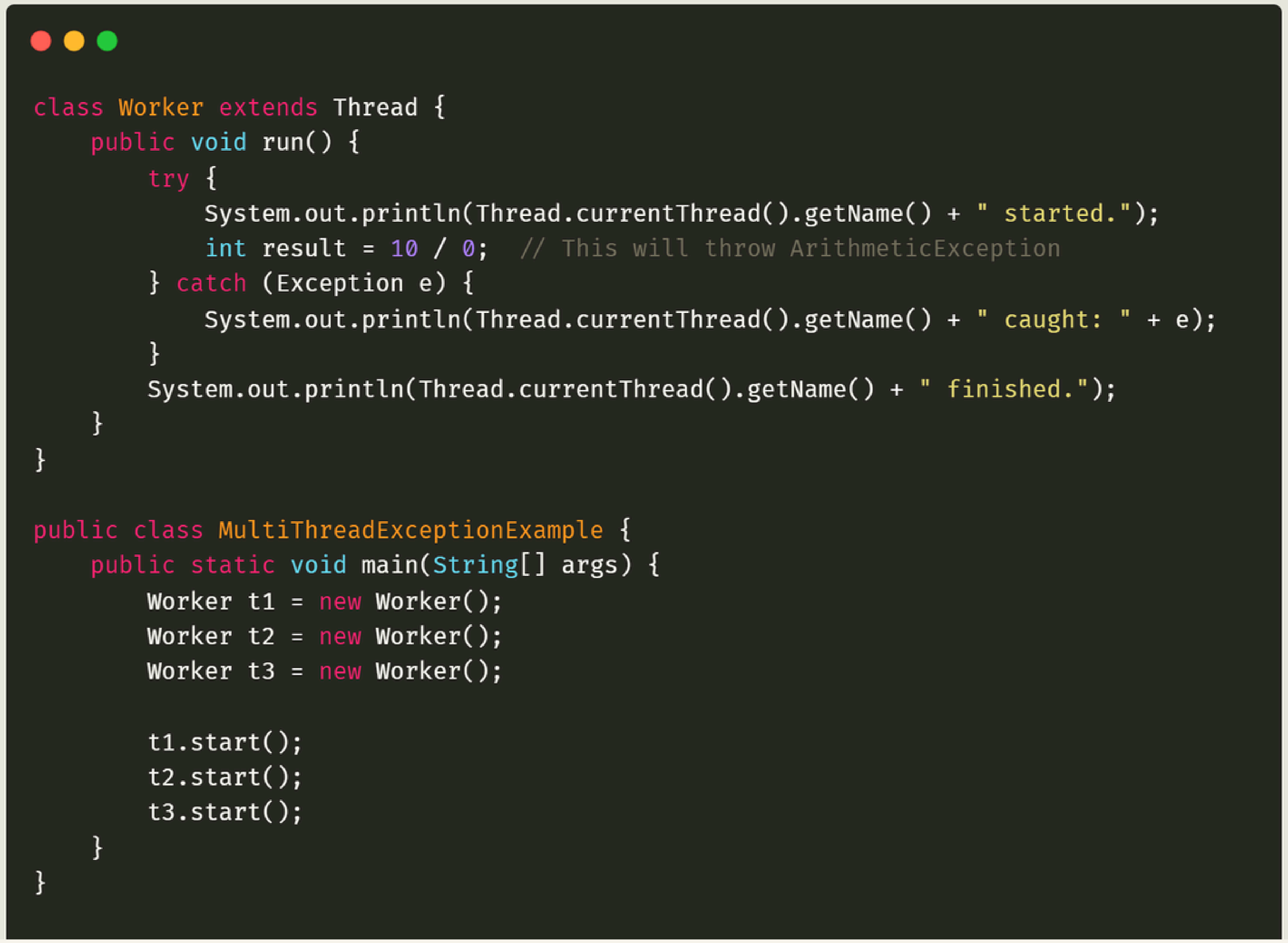
● Unchecked Exceptions

- These occur at runtime and are not checked at compile time.
- They usually indicate programming logic errors.

HANDLING EXCEPTIONS WITH TRY-CATCH

- When multiple threads are running, each thread operates independently.
- An exception in one thread does not stop or affect other threads.
- To handle exceptions safely, we should use a try–catch block inside the run() method of the thread.

HANDLING EXCEPTIONS WITH TRY-CATCH



```java
class Worker extends Thread {
    public void run() {
        try {
            System.out.println(Thread.currentThread().getName() + " started.");
            int result = 10 / 0;  // This will throw ArithmeticException
        } catch (Exception e) {
            System.out.println(Thread.currentThread().getName() + " caught: " + e);
        }
        System.out.println(Thread.currentThread().getName() + " finished.");
    }
}

public class MultiThreadExceptionExample {
    public static void main(String[] args) {
        Worker t1 = new Worker();
        Worker t2 = new Worker();
        Worker t3 = new Worker();

        t1.start();
        t2.start();
        t3.start();
    }
}
```

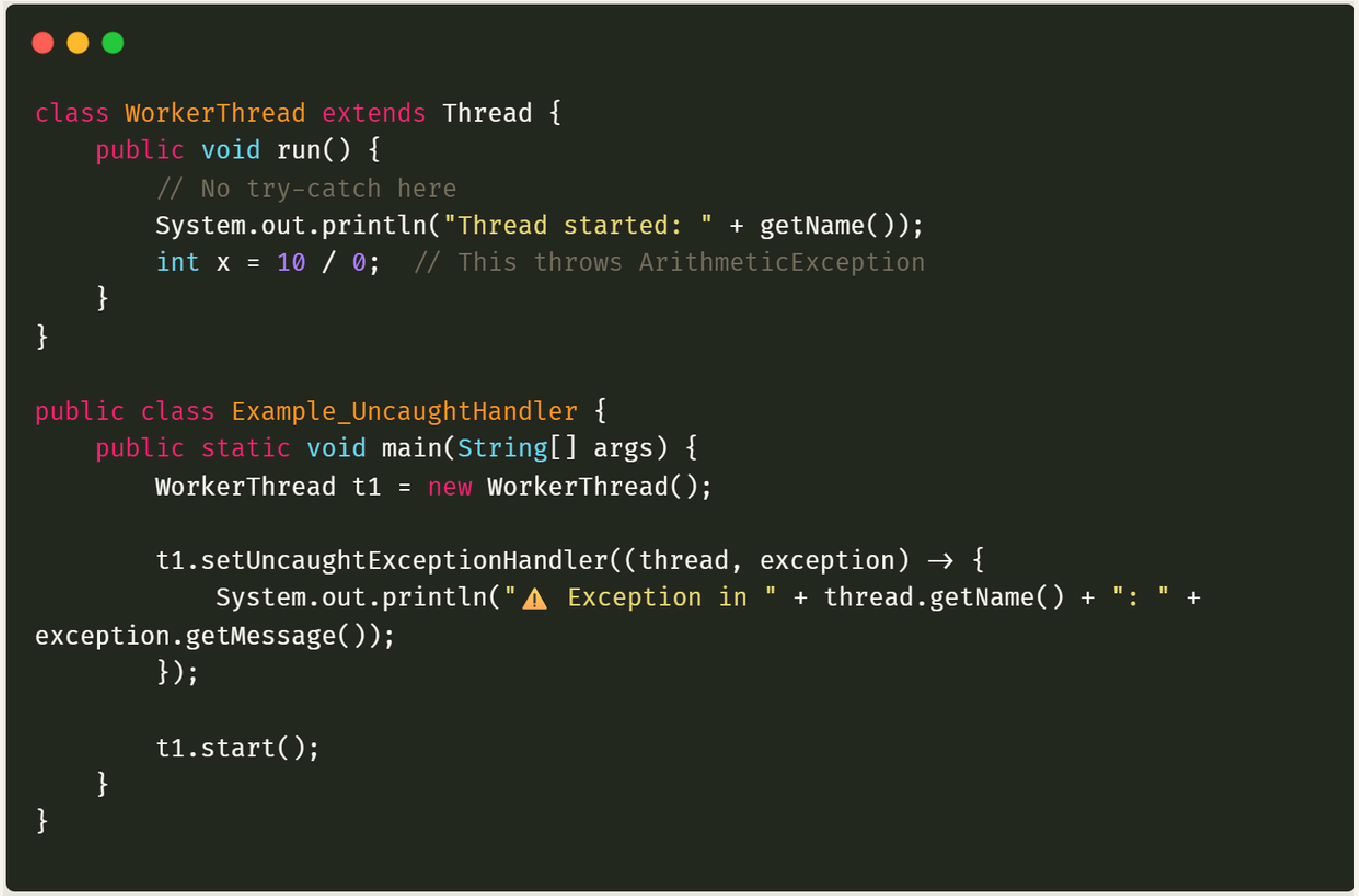
ADVANCED EXCEPTION HANDLING IN THREADS

- Instead of adding try-catch in every thread, you can use:
    - UncaughtExceptionHandler – handles uncaught exceptions for a specific thread.
    - DefaultUncaughtExceptionHandler – applies to all threads in the program.

UNCAUGHTEXCEPTIONHANDLER

- **Each thread in Java can throw an exception during its execution.**
- **If that exception is not handled inside the run() method (i.e., no try-catch), it becomes an uncaught exception.**
- **However, we can customize this behavior by assigning a specific UncaughtExceptionHandler to that thread.**

UNCAUGHTEXCEPTIONHANDLER

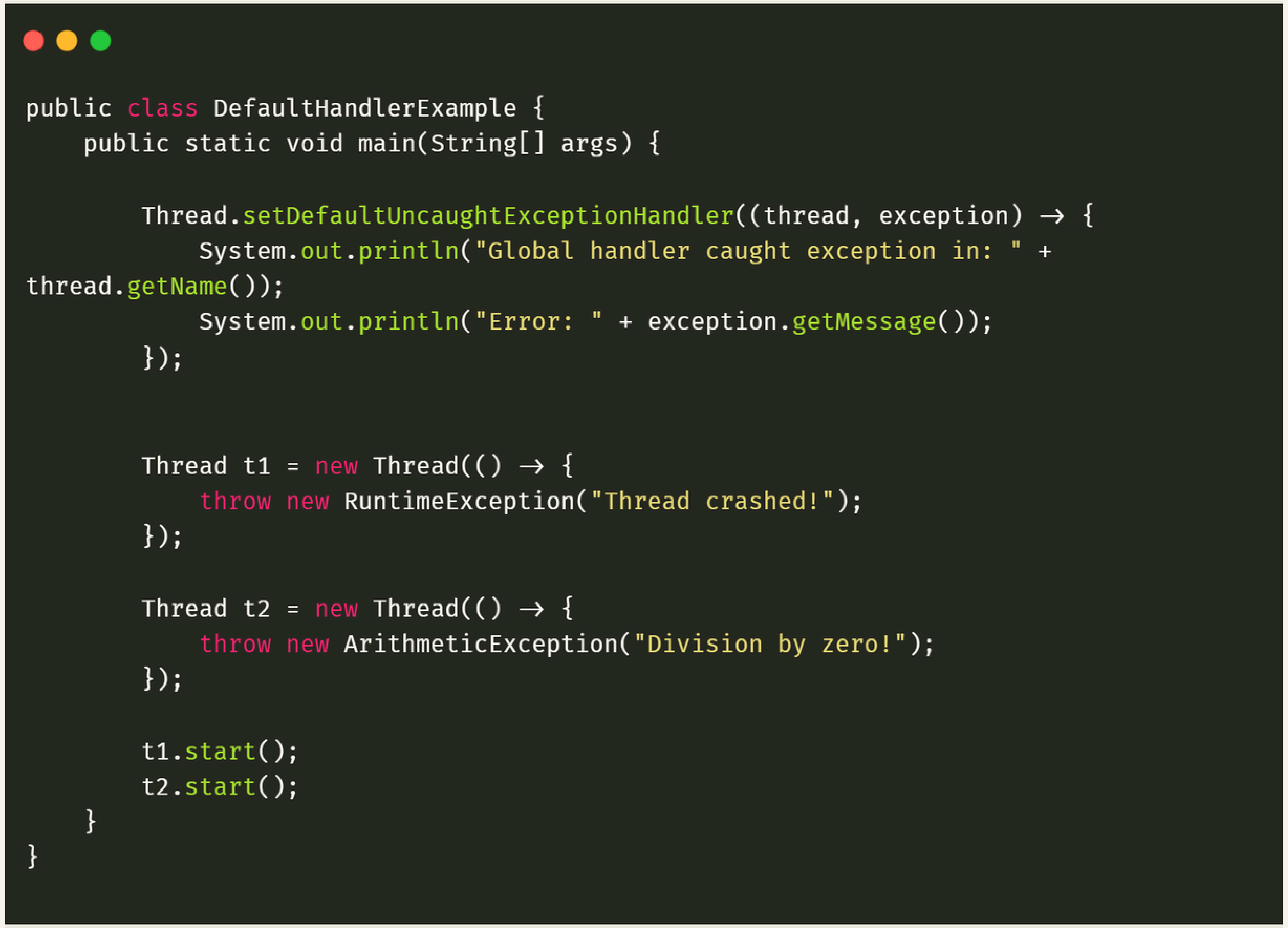


```
class WorkerThread extends Thread {
    public void run() {
        // No try-catch here
        System.out.println("Thread started: " + getName());
        int x = 10 / 0;  // This throws ArithmeticException
    }
}

public class Example_UncaughtHandler {
    public static void main(String[] args) {
        WorkerThread t1 = new WorkerThread();

        t1.setUncaughtExceptionHandler((thread, exception) -> {
            System.out.println("⚠ Exception in " + thread.getName() + ": " +
exception.getMessage());
        });

        t1.start();
    }
}
```

DEFAULTUNCAUGHTEXCEPTIONHANDLER

- In Java, when a thread throws an exception that is not caught inside its run() method, it becomes an uncaught exception.
- By default, such exceptions print a stack trace to the console and terminate the thread.To control this behavior globally across all threads, Java provides the method
- Thread.setDefaultUncaughtExceptionHandler() — a global handler that catches uncaught exceptions from any thread that does not have its own custom handler.

DEFAULTUNCAUGHTEXCEPTIONHANDLER



```
public class DefaultHandlerExample {
    public static void main(String[] args) {

        Thread.setDefaultUncaughtExceptionHandler((thread, exception) → {
            System.out.println("Global handler caught exception in: " +
thread.getName());
            System.out.println("Error: " + exception.getMessage());
        });


        Thread t1 = new Thread(() → {
            throw new RuntimeException("Thread crashed!");
        });

        Thread t2 = new Thread(() → {
            throw new ArithmeticException("Division by zero!");
        });

        t1.start();
        t2.start();
    }
}
```

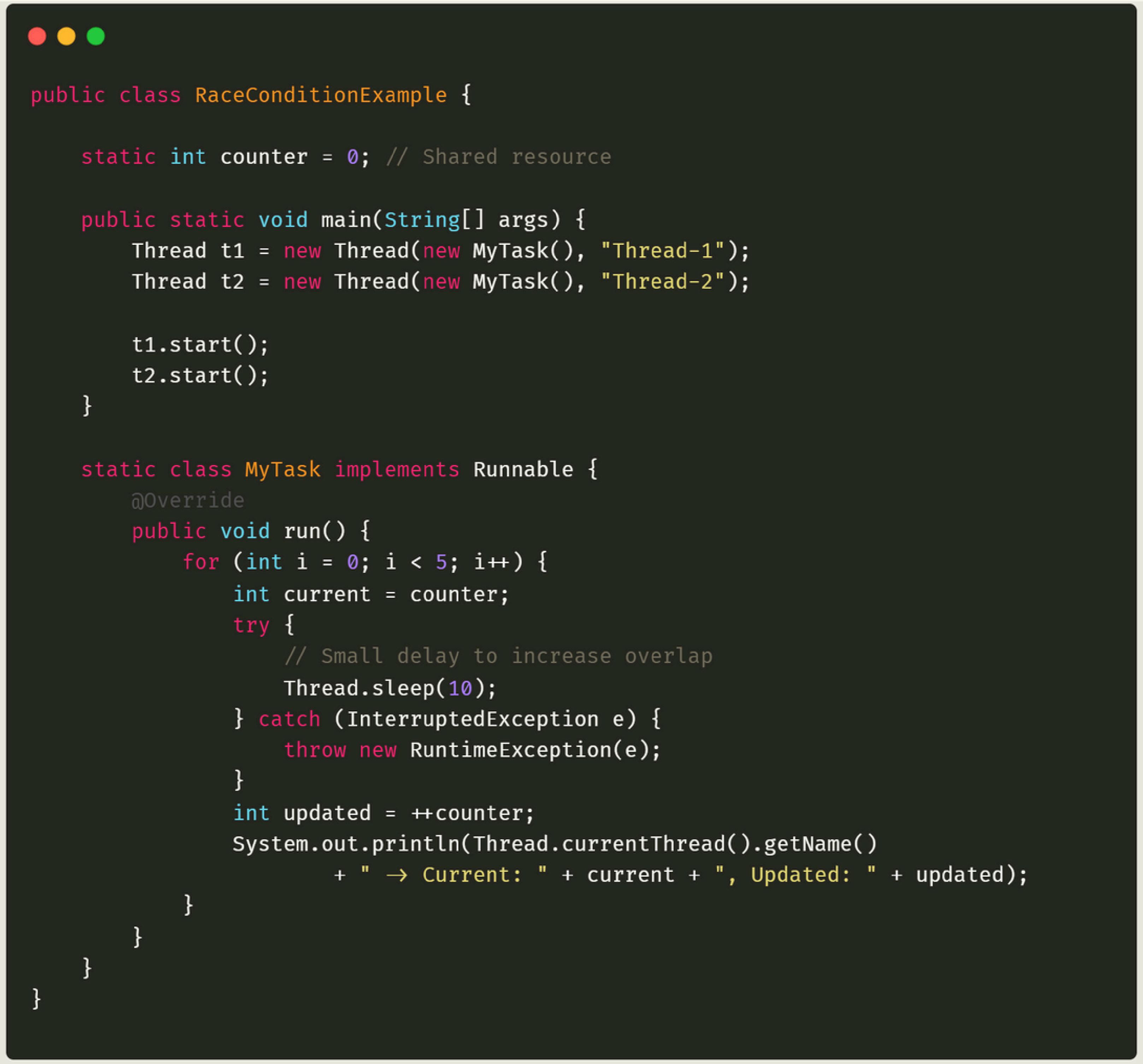
UNCAUGHTEXCEPTIONHANDLER VS DEFAULTUNCAUGHTEXCEPTIONHANDLER

Feature	UncaughtExceptionHandler	DefaultUncaughtExceptionHandler
Scope	Assigned to a single thread only	Global — applies to all threads
Assignment Method	thread.setUncaughtExceptionHandler()	Thread.setDefaultUncaughtExceptionHandler()
Priority	Has higher priority (used first if defined)	Used only when no thread-specific handler exists
Typical Use Case	Custom handling for specific threads	Centralized handling for all threads
Flexibility	High — can define behavior per thread	Lower — one global behavior for all threads

RACE CONDITION IN JAVA

- A race condition occurs when two or more threads access and modify a shared resource (such as a variable) concurrently without proper synchronization.
- Since thread execution order is unpredictable, the final result becomes inconsistent or incorrect.

RACE CONDITION IN JAVA



```java
public class RaceConditionExample {

    static int counter = 0; // Shared resource

    public static void main(String[] args) {
        Thread t1 = new Thread(new MyTask(), "Thread-1");
        Thread t2 = new Thread(new MyTask(), "Thread-2");

        t1.start();
        t2.start();
    }

    static class MyTask implements Runnable {
        @Override
        public void run() {
            for (int i = 0; i < 5; i++) {
                int current = counter;
                try {
                    // Small delay to increase overlap
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
                int updated = ++counter;
                System.out.println(Thread.currentThread().getName()
                        + " -> Current: " + current + ", Updated: " + updated);
            }
        }
    }
}
```

THANK YOU