데이터융합SW과
김규석 교수

JAVA

기본 프로그래밍 08

Objective

Thread

- Two ways to create and run a thread

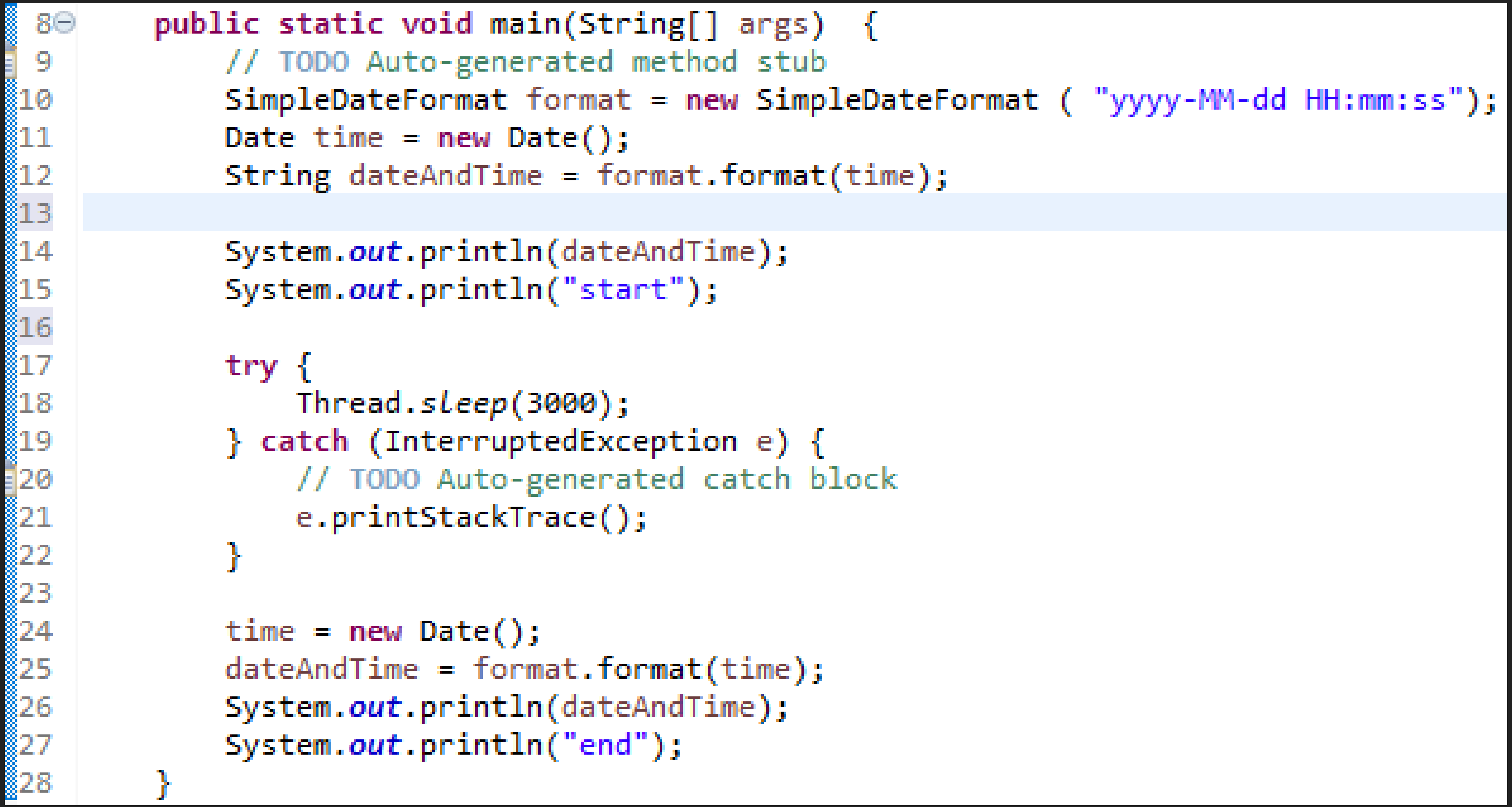
Exception

- try … catch

Thread(Cont'd)

Thread.sleep()

- Pause the execution of current thread for specific time in milliseconds

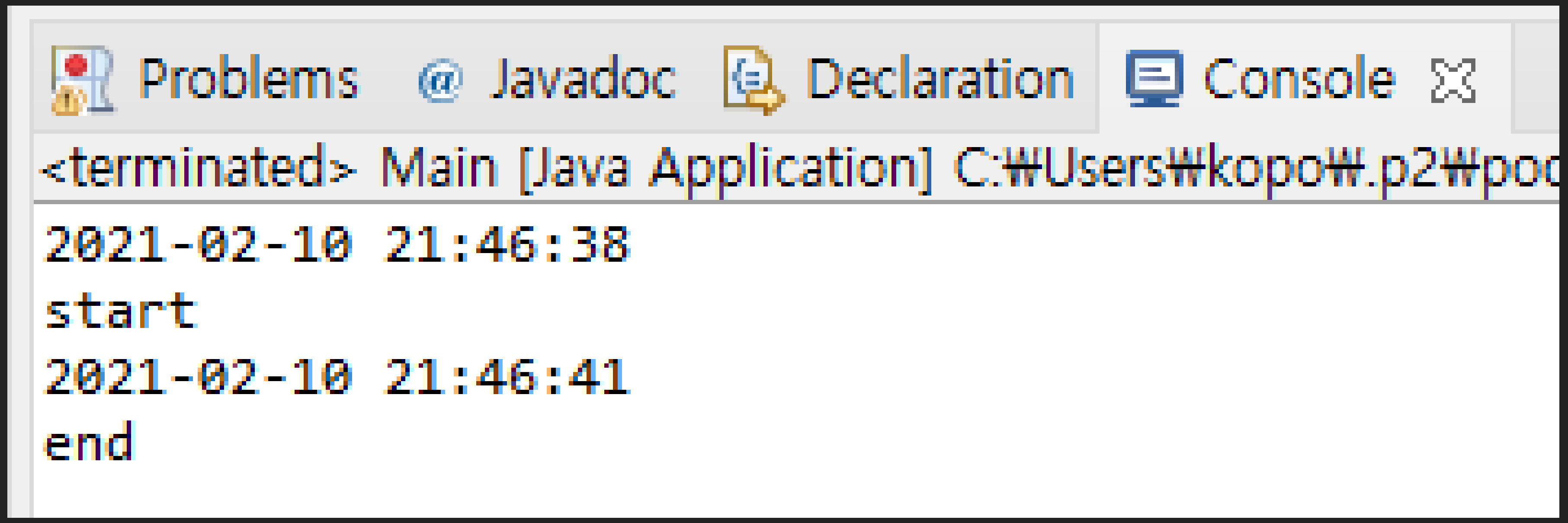


```java
	public static void main(String[] args)  {
		// TODO Auto-generated method stub
		SimpleDateFormat format = new SimpleDateFormat ( "yyyy-MM-dd HH:mm:ss");
		Date time = new Date();
		String dateAndTime = format.format(time);

		System.out.println(dateAndTime);
		System.out.println("start");

		try {
			Thread.sleep(3000);
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		time = new Date();
		dateAndTime = format.format(time);
		System.out.println(dateAndTime);
		System.out.println("end");
	}
```



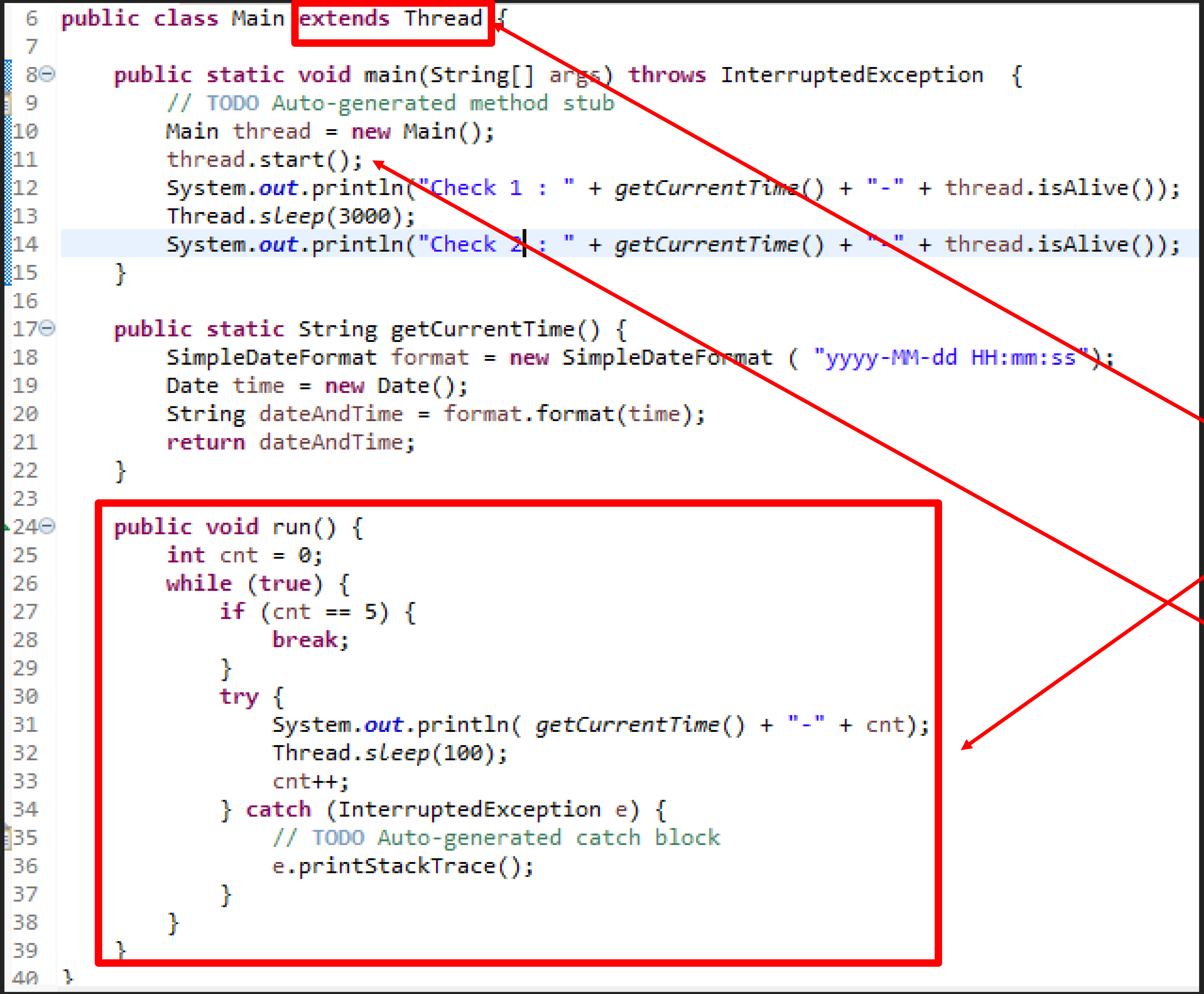
```
<terminated> Main [Java Application] C:₩Users₩kopo₩.p2₩poo
2021-02-10 21:46:38
start
2021-02-10 21:46:41
end
```

P1 : Print a number from 1 to 100 every second

Thread(Cont'd)

Thread

- Allows the program to operate multiple things at the same time

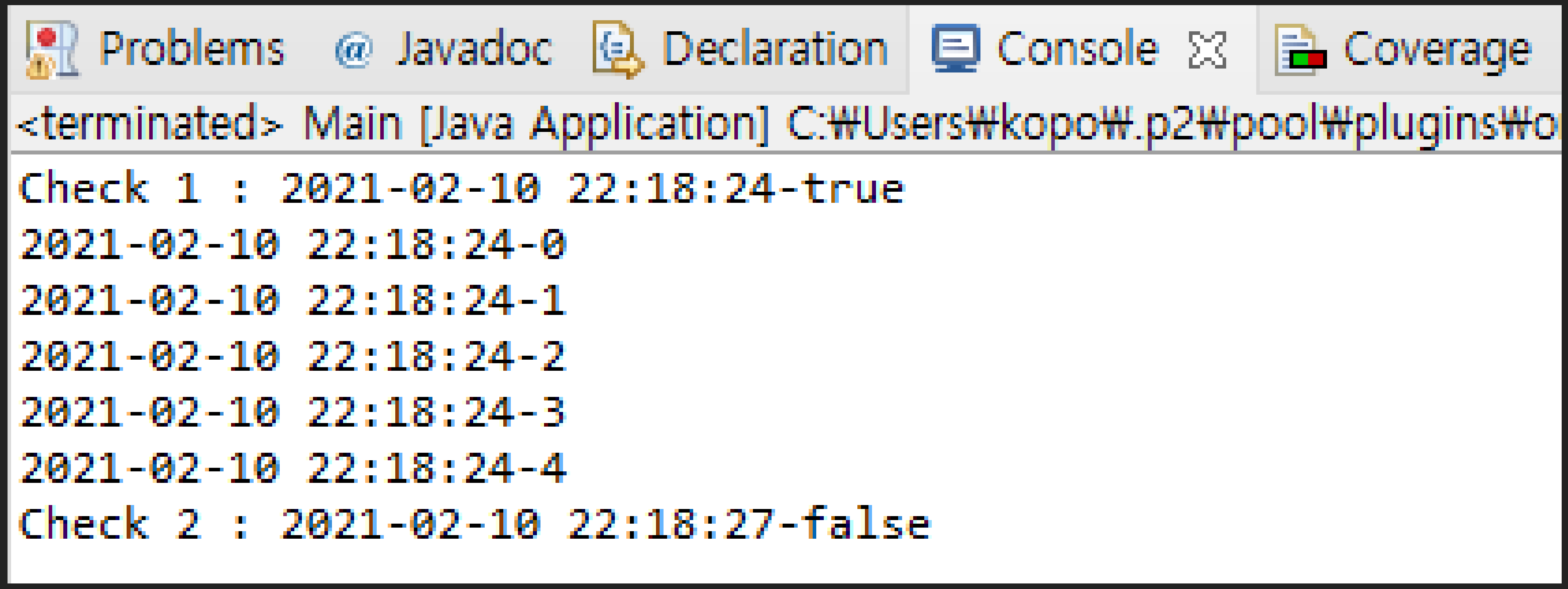


```java
public class Main extends Thread {

    public static void main(String[] args) throws InterruptedException  {
        // TODO Auto-generated method stub
        Main thread = new Main();
        thread.start();
        System.out.println("Check 1 : " + getCurrentTime() + "-" + thread.isAlive());
        Thread.sleep(3000);
        System.out.println("Check 2 : " + getCurrentTime() + "-" + thread.isAlive());
    }

    public static String getCurrentTime() {
        SimpleDateFormat format = new SimpleDateFormat ( "yyyy-MM-dd HH:mm:ss");
        Date time = new Date();
        String dateAndTime = format.format(time);
        return dateAndTime;
    }

    public void run() {
        int cnt = 0;
        while (true) {
            if (cnt == 5) {
                break;
            }
            try {
                System.out.println( getCurrentTime() + "-" + cnt);
                Thread.sleep(100);
                cnt++;
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```



```
<terminated> Main [Java Application] C:₩Users₩kopo₩.p2₩pool₩plugins₩o
Check 1 : 2021-02-10 22:18:24-true
2021-02-10 22:18:24-0
2021-02-10 22:18:24-1
2021-02-10 22:18:24-2
2021-02-10 22:18:24-3
2021-02-10 22:18:24-4
Check 2 : 2021-02-10 22:18:27-false
```

Create a thread

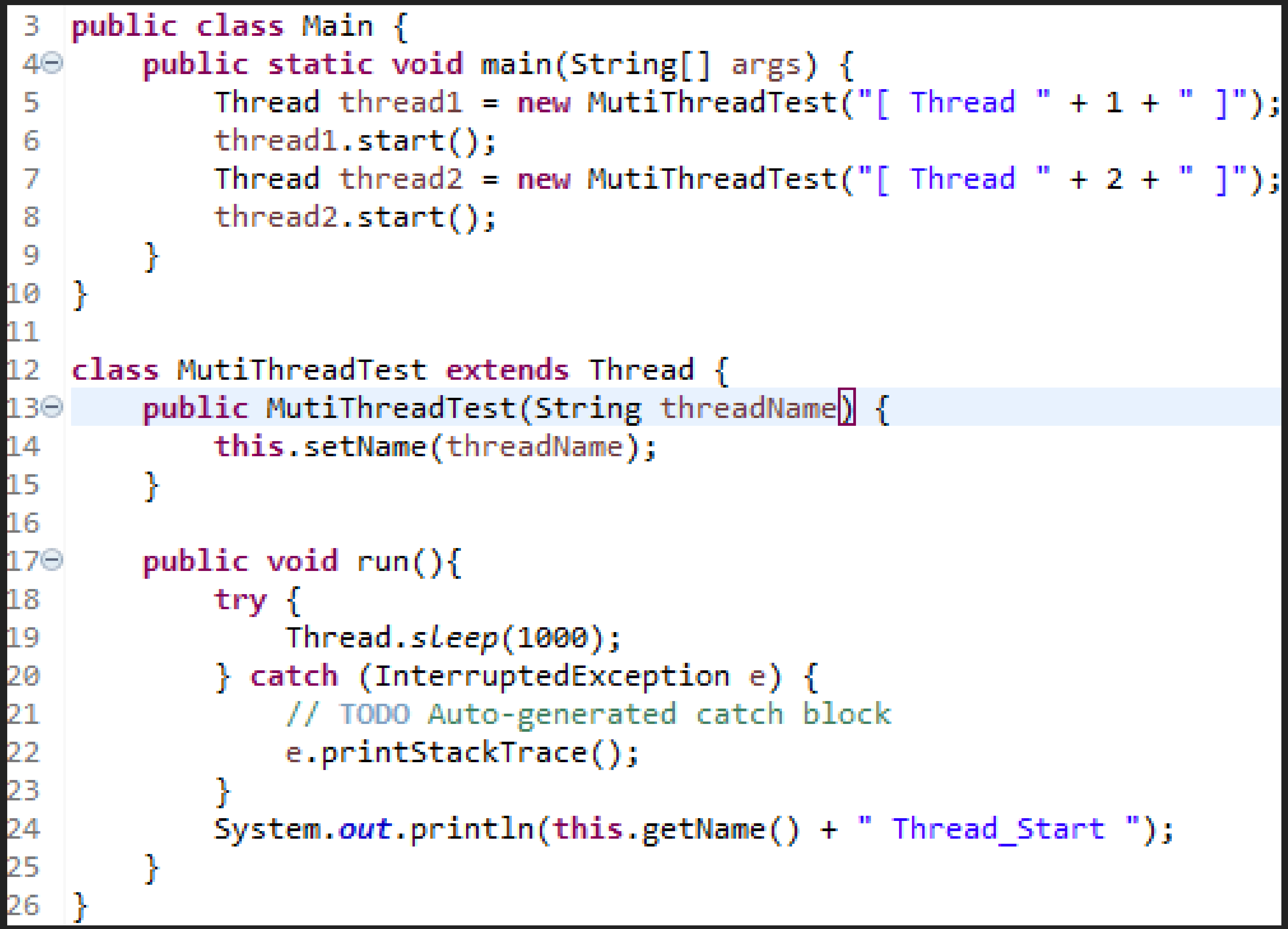
Run a thread

P2 : A thread is naturally destroyed

Thread(Cont'd)

setPriority()

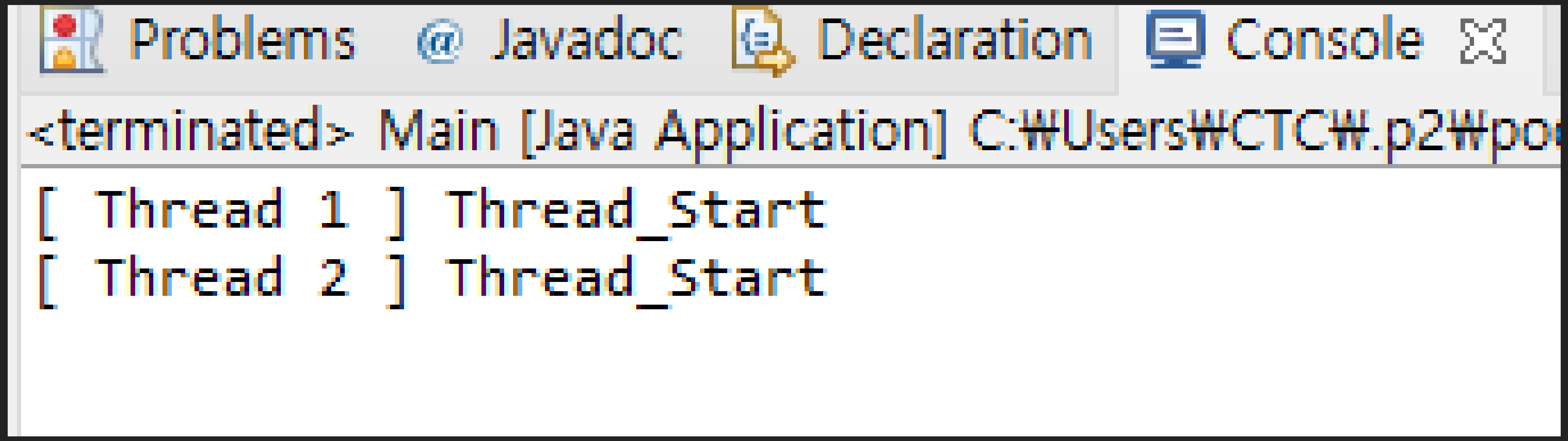
- This method is used to change the thread's priority



```java
public class Main {
    public static void main(String[] args) {
        Thread thread1 = new MutiThreadTest("[ Thread " + 1 + " ]");
        thread1.start();
        Thread thread2 = new MutiThreadTest("[ Thread " + 2 + " ]");
        thread2.start();
    }
}

class MutiThreadTest extends Thread {
    public MutiThreadTest(String threadName) {
        this.setName(threadName);
    }

    public void run(){
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println(this.getName() + " Thread_Start ");
    }
}
```

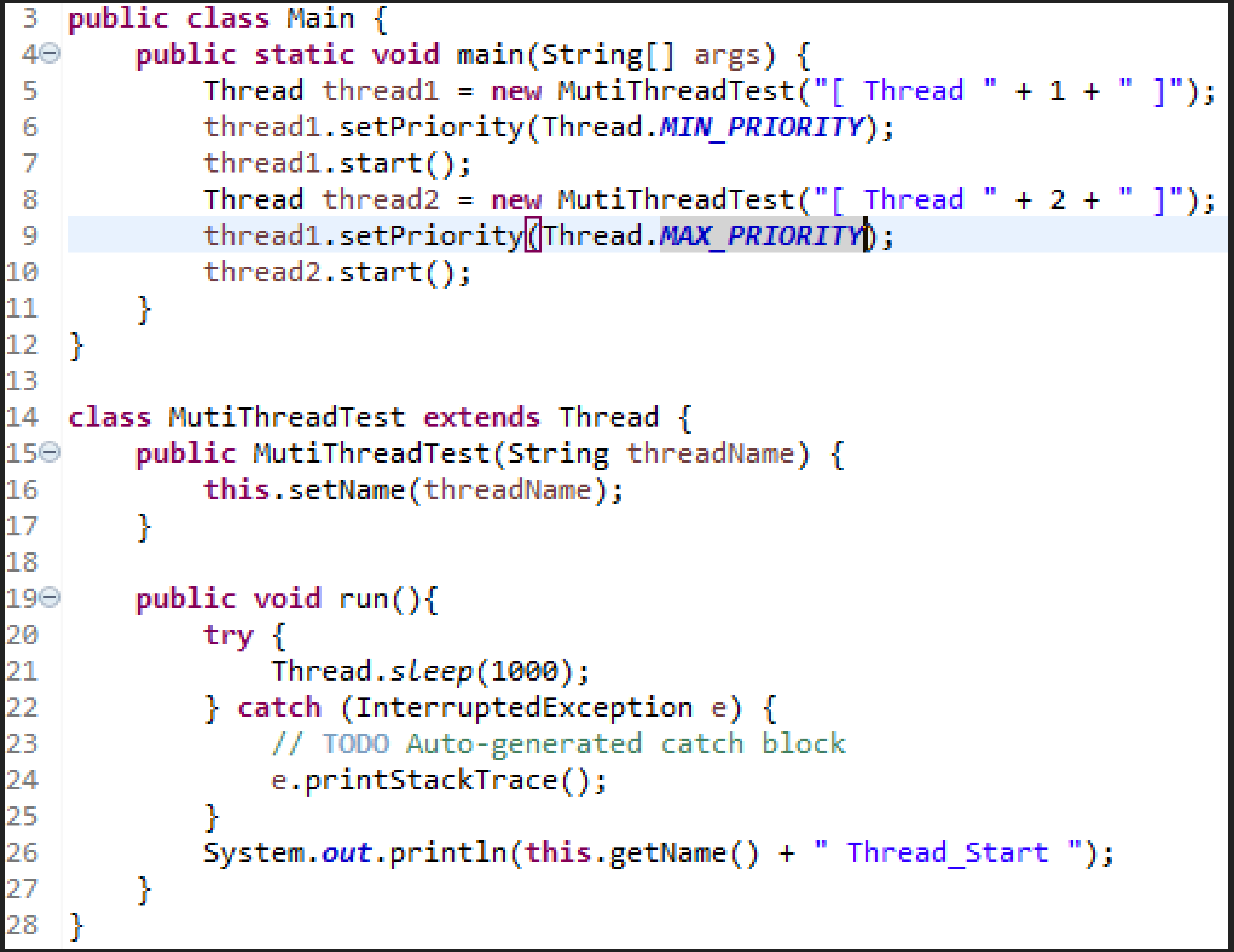


```
<terminated> Main [Java Application] C:₩Users₩CTC₩.p2₩poo
[ Thread 1 ] Thread_Start
[ Thread 2 ] Thread_Start
```

Thread(Cont'd)

setPriority()

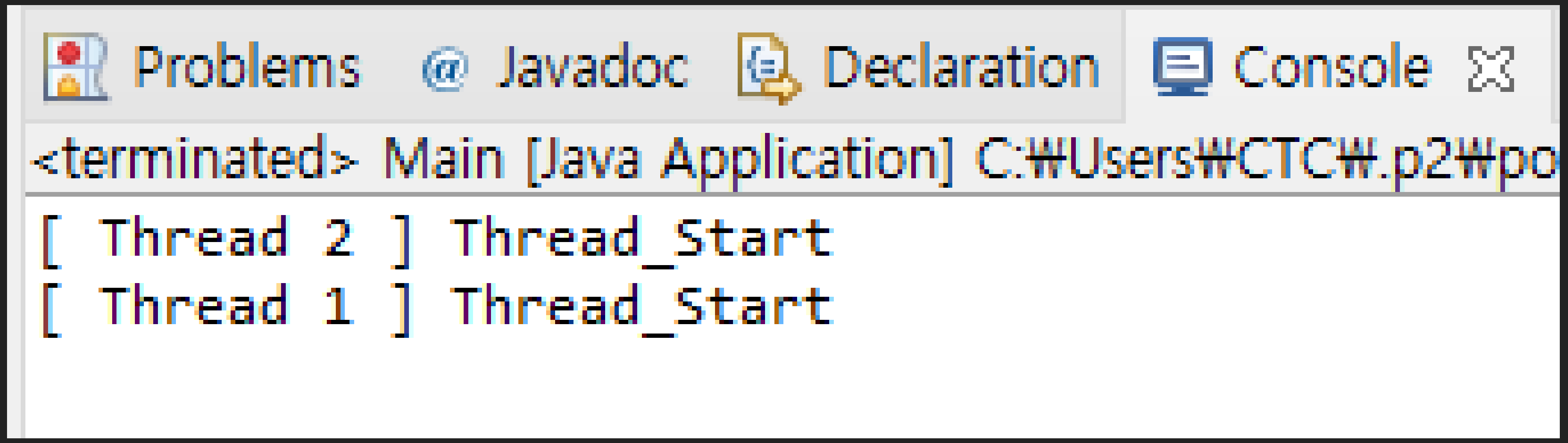
- This method is used to change the thread's priority



```java
public class Main {
    public static void main(String[] args) {
        Thread thread1 = new MutiThreadTest("[ Thread " + 1 + " ]");
        thread1.setPriority(Thread.MIN_PRIORITY);
        thread1.start();
        Thread thread2 = new MutiThreadTest("[ Thread " + 2 + " ]");
        thread1.setPriority(Thread.MAX_PRIORITY);
        thread2.start();
    }
}

class MutiThreadTest extends Thread {
    public MutiThreadTest(String threadName) {
        this.setName(threadName);
    }

    public void run(){
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println(this.getName() + " Thread_Start ");
    }
}
```



```
<terminated> Main [Java Application] C:₩Users₩CTC₩.p2₩po
[ Thread 2 ] Thread_Start
[ Thread 1 ] Thread_Start
```

P3 : Change the parameter of "setPriority()"

Thread(Cont’d)

Syncronization

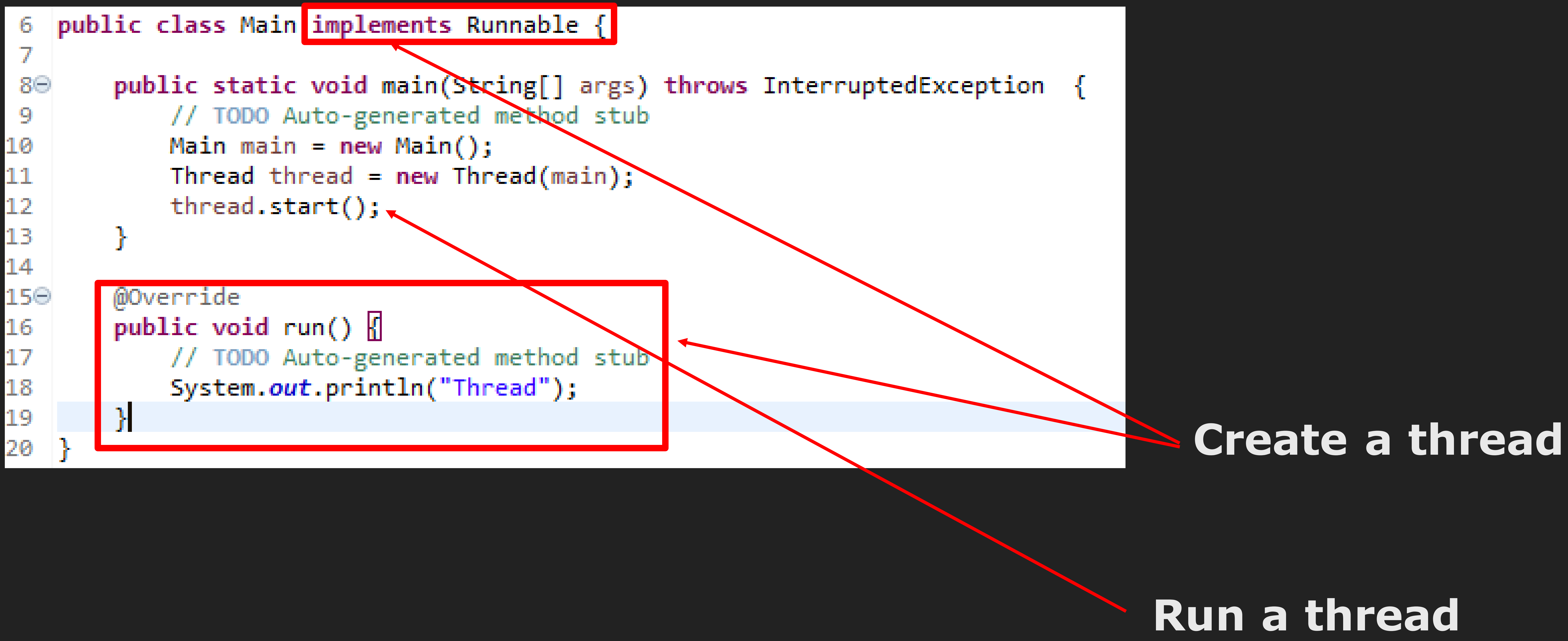
- A synchronized blocks can only have one thread executing at the same time
- “synchronized” < Keyword

P4 : Find out and execute an example of “synchronized”

Thread

Thread II

- The following code is the other way to use a thread



```java
public class Main implements Runnable {

    public static void main(String[] args) throws InterruptedException  {
        // TODO Auto-generated method stub
        Main main = new Main();
        Thread thread = new Thread(main);
        thread.start();
    }

    @Override
    public void run() {
        // TODO Auto-generated method stub
        System.out.println("Thread");
    }
}
```

P5 : Implement the same program as P1

P6

Timer

- Print the current time on the screen
- User inputs a time in second
- The program prints out “time is over” after the input time

```
12:34:45
30
time is over(12:35:15)
```

P7

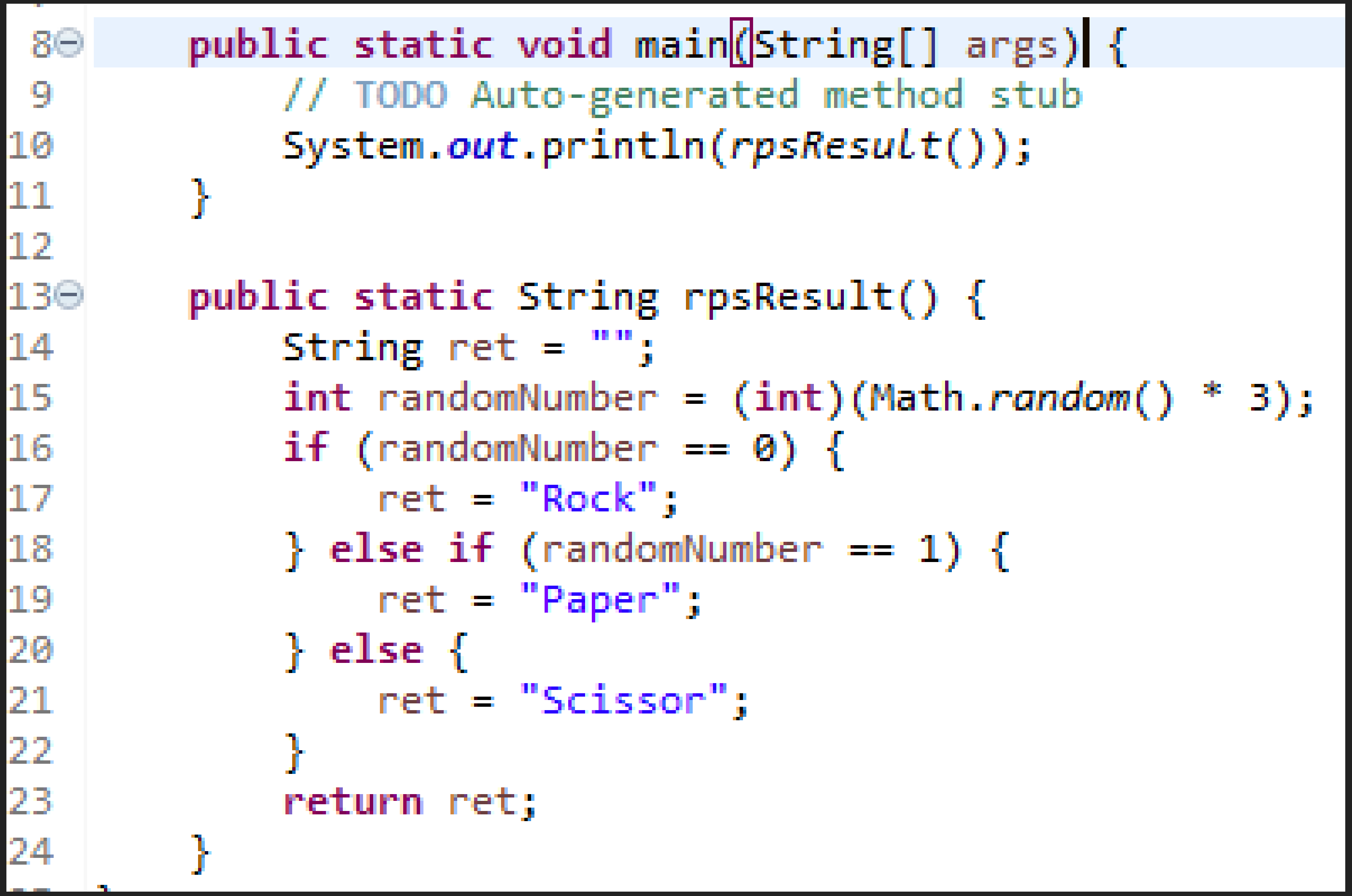
Timer II

- Print the current time on the screen
- User inputs a time in second
- The program prints out “time is over” after the input time
- Add an exceptional statement that waits for the user input again when the time is already over

```
12:34:45
30
time is already over(It’s 12:36:10)
100
time is over(12:37:50)
```

Rock-Paper-Scissors Game

- User inputs a number(0 : Rock, 1 : Paper, 2 : Scissor)
- User plays the Rock-Paper-Scissors game with the computer
- Print the result
- Hint, Math.random() returns 0.0 <= number < 1.0



```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println(rpsResult());
    }

    public static String rpsResult() {
        String ret = "";
        int randomNumber = (int)(Math.random() * 3);
        if (randomNumber == 0) {
            ret = "Rock";
        } else if (randomNumber == 1) {
            ret = "Paper";
        } else {
            ret = "Scissor";
        }
        return ret;
    }
```

P9

Running Race Game

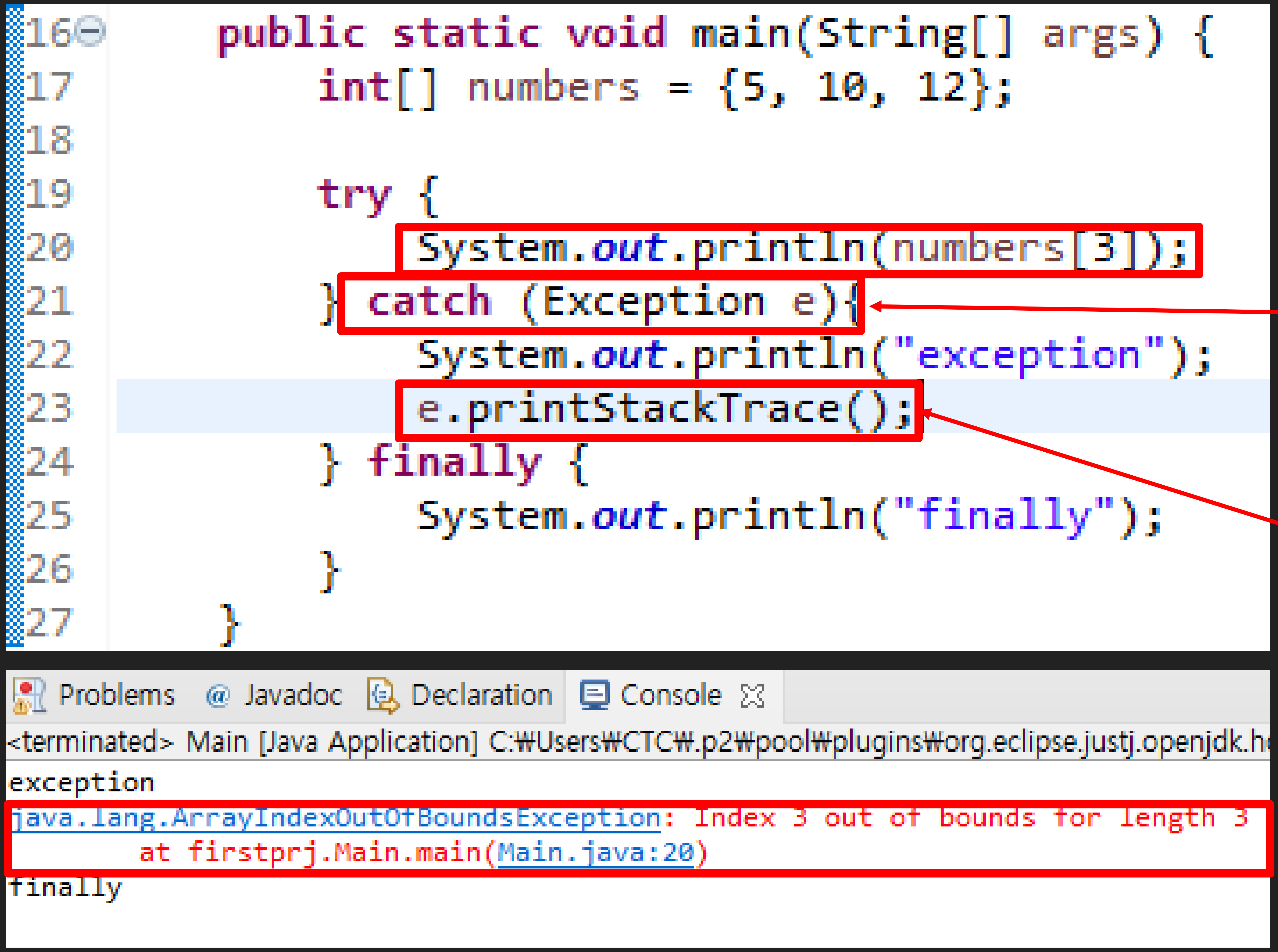
- User inputs a number N(number of runners)
- N of threads are created
- Print the running progresses of runners every second
- Use the function, Math.random() to update the progresses (Maximum is less than 10m and running distance is 50m)

```
3
(1s)    (2s)    (3s)    (4s)    (5s)    (6s)    (7s)       (8s)       (9s)
6.5m    12.3m   17.2m   26.4m   34.5m   42.1m   (Finished)                       1
3.2m    8.9m    16.8m   24.6m   31.2m   40.0m   45.1m      49.1m      (Finished) 3
3.1m    10.2m   17.1m   23.8m   32.4m   39.7m   46.2m      (Finished)            2
```

Exception(Cont'd)

try … catch

- try : a block of code to be executed
- catch : a block of code when an error occurs
- finally : a block of code that is executed, regardless of the result



```java
    public static void main(String[] args) {
        int[] numbers = {5, 10, 12};

        try {
            System.out.println(numbers[3]);
        } catch (Exception e){
            System.out.println("exception");
            e.printStackTrace();
        } finally {
            System.out.println("finally");
        }
    }
```

```
<terminated> Main [Java Application] C:₩Users₩CTC₩.p2₩pool₩plugins₩org.eclipse.justj.openjdk.h
exception
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
	at firstprj.Main.main(Main.java:20)
finally
```

P10 : According to the error logs, change the error type

Error Type

Return Value

Exception(Cont’d)

Type of Exceptions

- **ArithmeticException** : when an exceptional condition has occurred in an arithmetic operation.

- **ArrayIndexOutOfBoundsException :** when n array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

- **ClassNotFoundException :** when trying to access a class whose definition is not found

- **FileNotFoundException :** when a file is not accessible or does not open.

- **IOException :** when an input-output operation failed or interrupted

- **InterruptedException :** when a thread is waiting, sleeping, or doing some processing, and it is interrupted.

Source : https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/

Exception

Type of Exceptions

- **NoSuchFieldException :** when a class does not contain the field specified
- **NoSuchMethodException :** when accessing a method which is not found.
- **NullPointerException :** when referring to the members of a null object. Null represents nothing
- **NumberFormatException :** when a method could not convert a string into a numeric format.
- **RuntimeException :** any exception which occurs during runtime.
- **StringIndexOutOfBoundsException :** thrown by String class methods to indicate that an index is either negative or greater than the size of the string

Source : https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/

P11

Exception Handling

- User inputs a number and the number defines the size of an array
- User inputs numbers more than the defined size
- In this case, this program prints “error” and starts again from the beginning, Otherwise, assign the numbers into the array, sort and print them

```
5
1 3 4 5 7 8
error
4
1 22 6 12
1 6 12 22
```

P12

Fibonacci Numbers

- The formula for this is as below

$F_0 = 0, F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 …

- User inputs a number
- Print the number of elements from the Fibonacci Numbers
- Add exception statements

```
7
0 1 1 2 3 5 8
```

P13

Comparing Strings

- User inputs the two words and assign them to the arrays

- Print the following information

1. Size of the first word
2. Size of the second word
3. Number of the same characters

- Add exception statements

```
school
scholar
1 : 6
2 : 7
3 : 5
```