Lambda Expression

Bright Gee Varghese

Lambda Expression

- A *lambda expression* is, essentially, an anonymous (that is, unnamed) method.
- Lambda expressions are also commonly referred to as *closures*.
- A *functional interface* is an interface that contains one and only one abstract method.

- *lambda operator* or the *arrow operator*, is **–>**.
- The left side specifies any parameters required by the lambda expression. (If no parameters are needed, an empty parameter list is used.)
- On the right side is the *lambda body,* which specifies the actions of the lambda expression.
- This lambda expression takes no parameters, thus the parameter list is empty. It returns the constant value 123.45

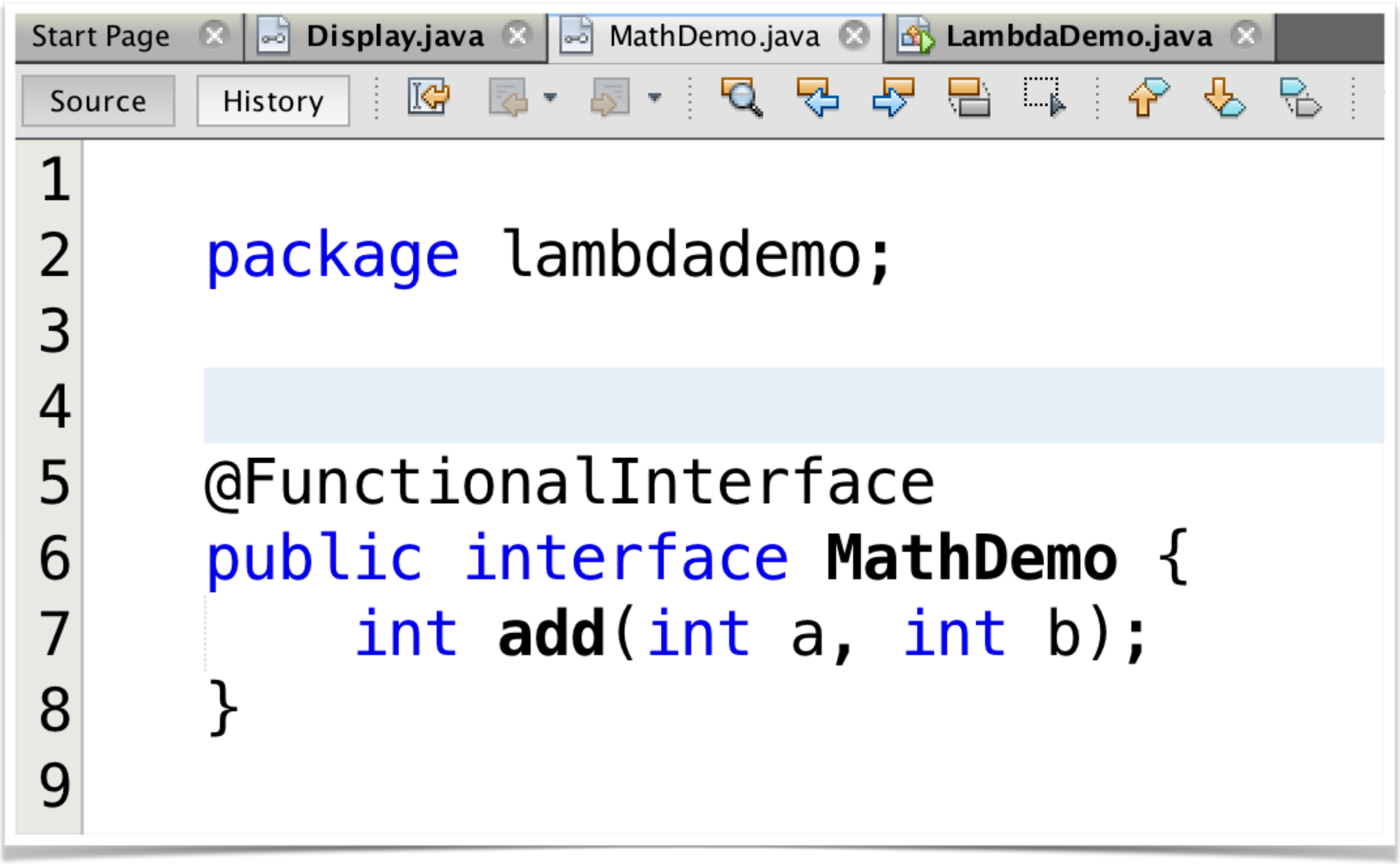
`() -> 123.45` $\Longrightarrow$ `double myMeth() { return 123.45; }`

`(n) -> (n % 2)==0` $\Longrightarrow$ This lambda expression returns **true** if the value of parameter **n** is even.

Functional Interfaces

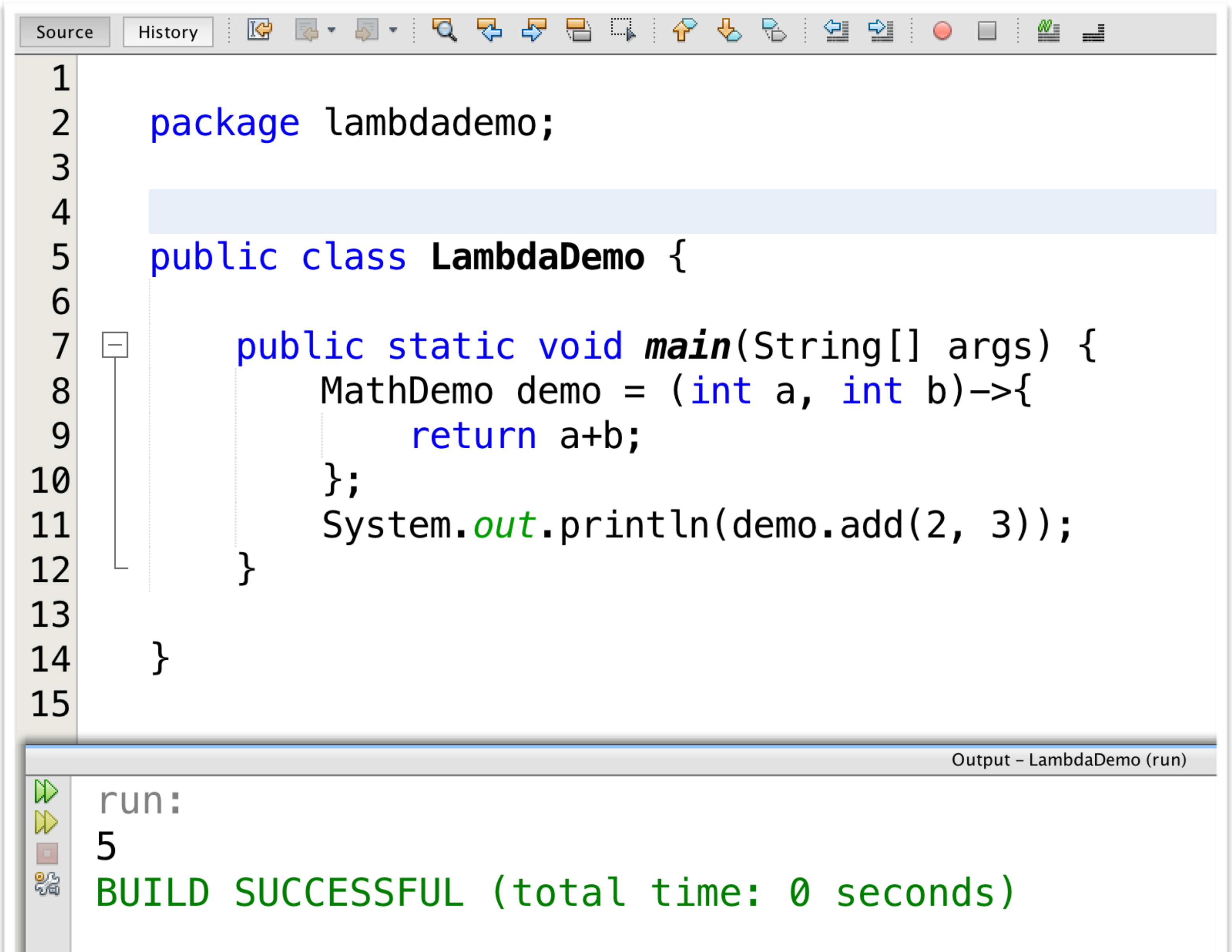
- A functional interface is an interface that specifies only one abstract method.

Create an interface which has one abstract method



```java

package lambdademo;


@FunctionalInterface
public interface MathDemo {
    int add(int a, int b);
}

```

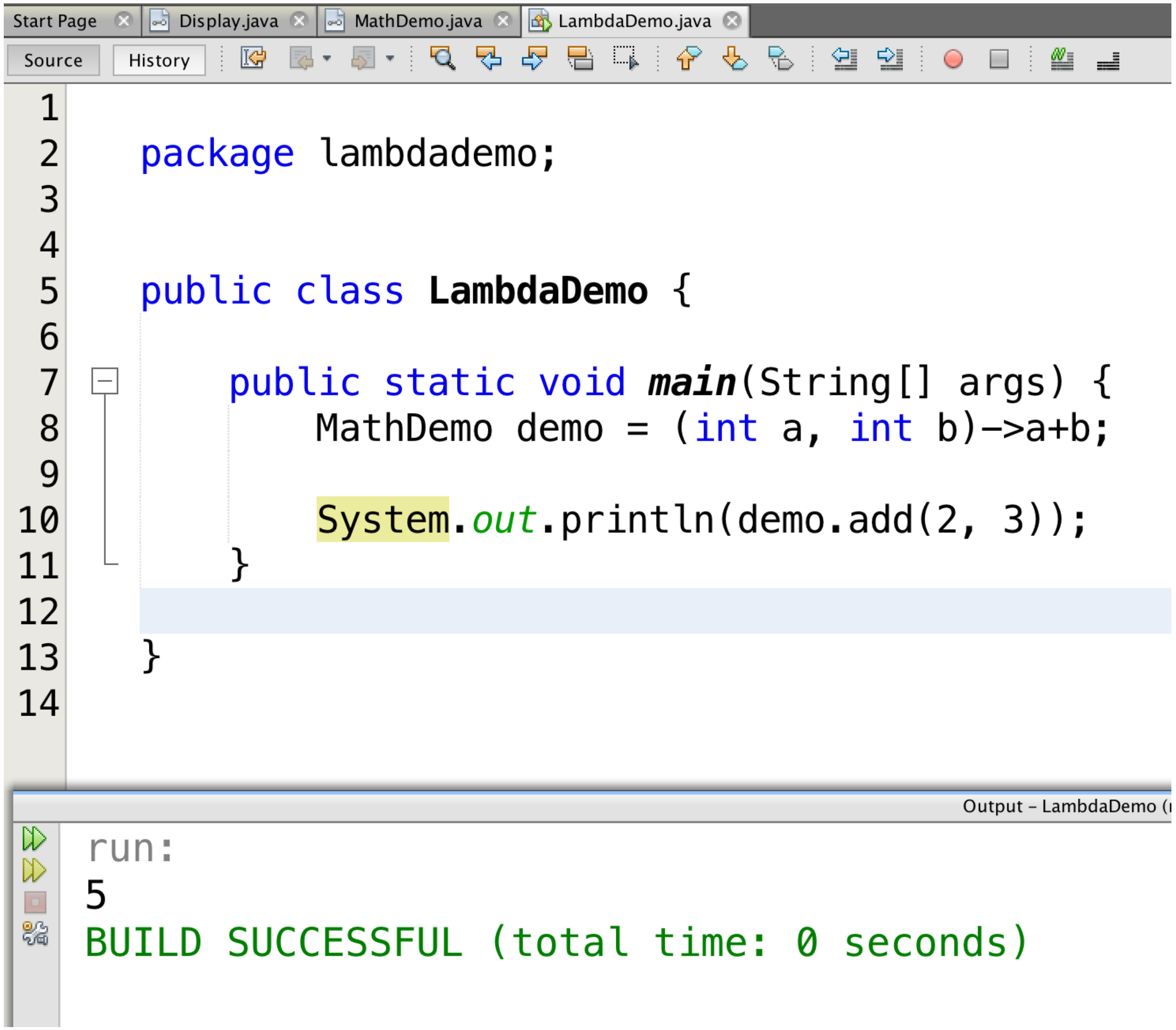
With braces and return



```java

package lambdademo;


public class LambdaDemo {

    public static void main(String[] args) {
        MathDemo demo = (int a, int b)->{
            return a+b;
        };
        System.out.println(demo.add(2, 3));
    }

}

```

```
run:
5
BUILD SUCCESSFUL (total time: 0 seconds)
```

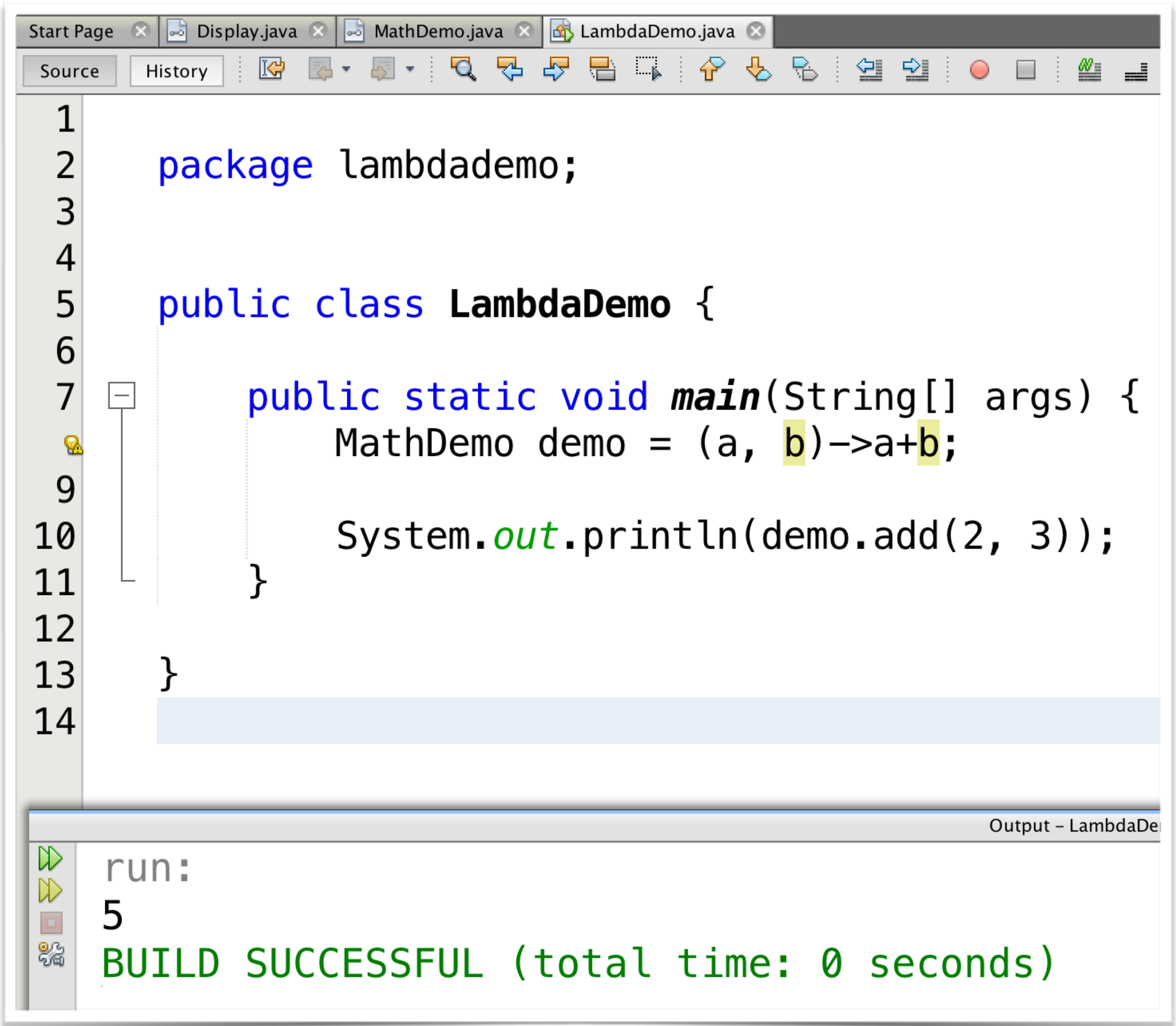
Without braces and no return



```java

package lambdademo;


public class LambdaDemo {

    public static void main(String[] args) {
        MathDemo demo = (int a, int b)->a+b;

        System.out.println(demo.add(2, 3));
    }

}

```

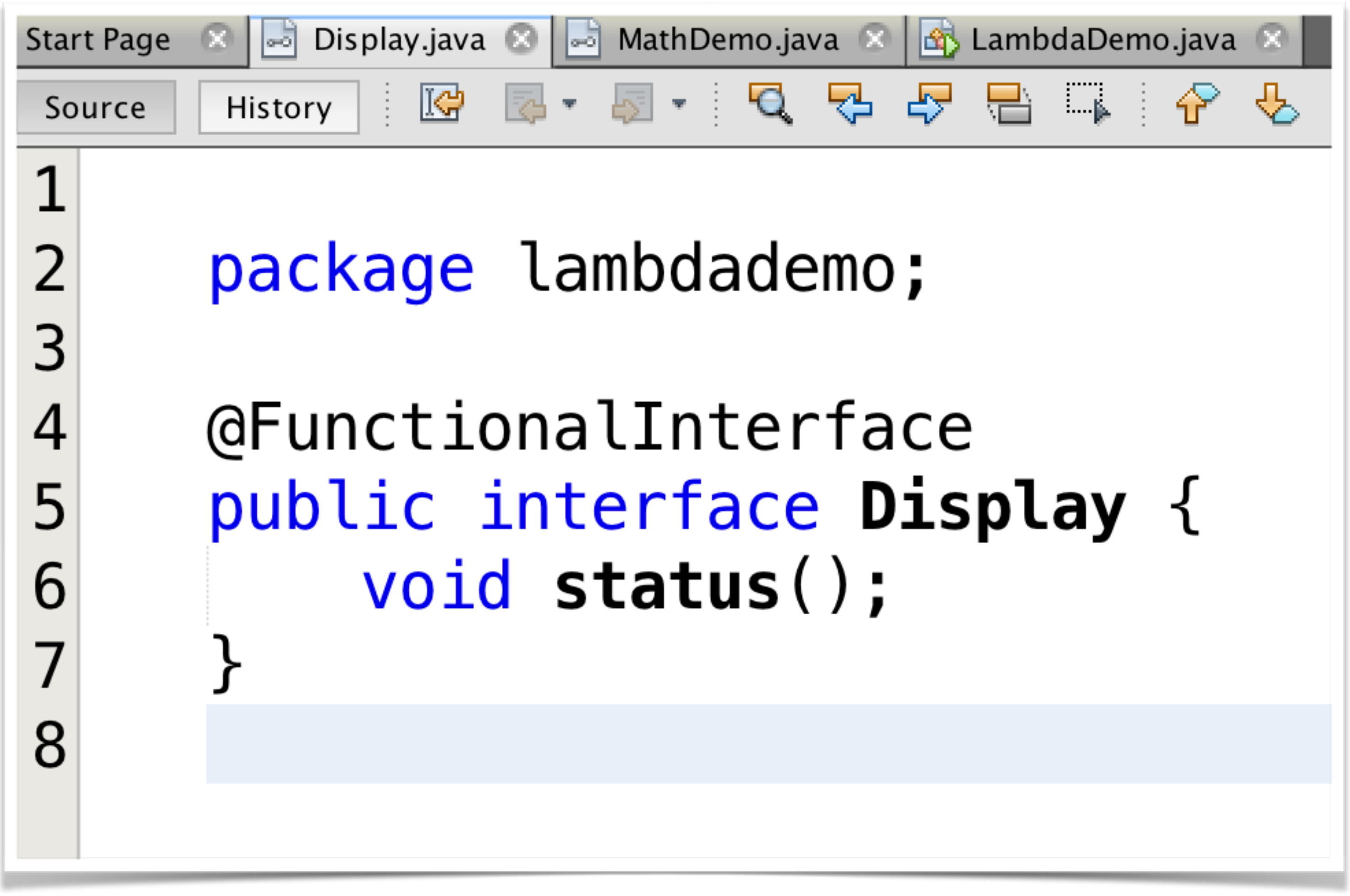
```
run:
5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Without braces, no return and no parameter type



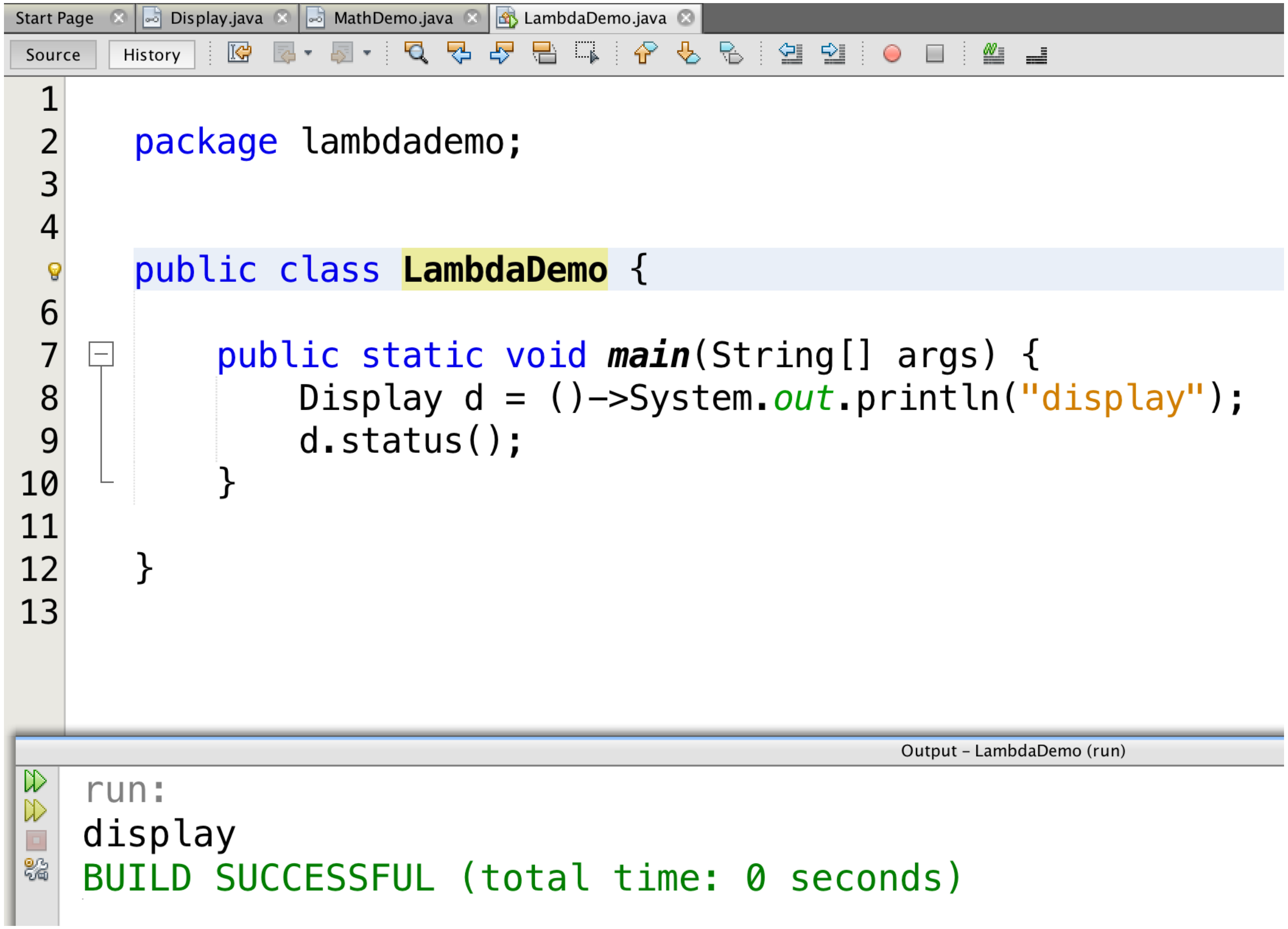
```

package lambdademo;


public class LambdaDemo {

    public static void main(String[] args) {
        MathDemo demo = (a, b)->a+b;

        System.out.println(demo.add(2, 3));
    }

}

```

```
run:
5
BUILD SUCCESSFUL (total time: 0 seconds)
```



```java

package lambdademo;

@FunctionalInterface
public interface Display {
    void status();
}

```

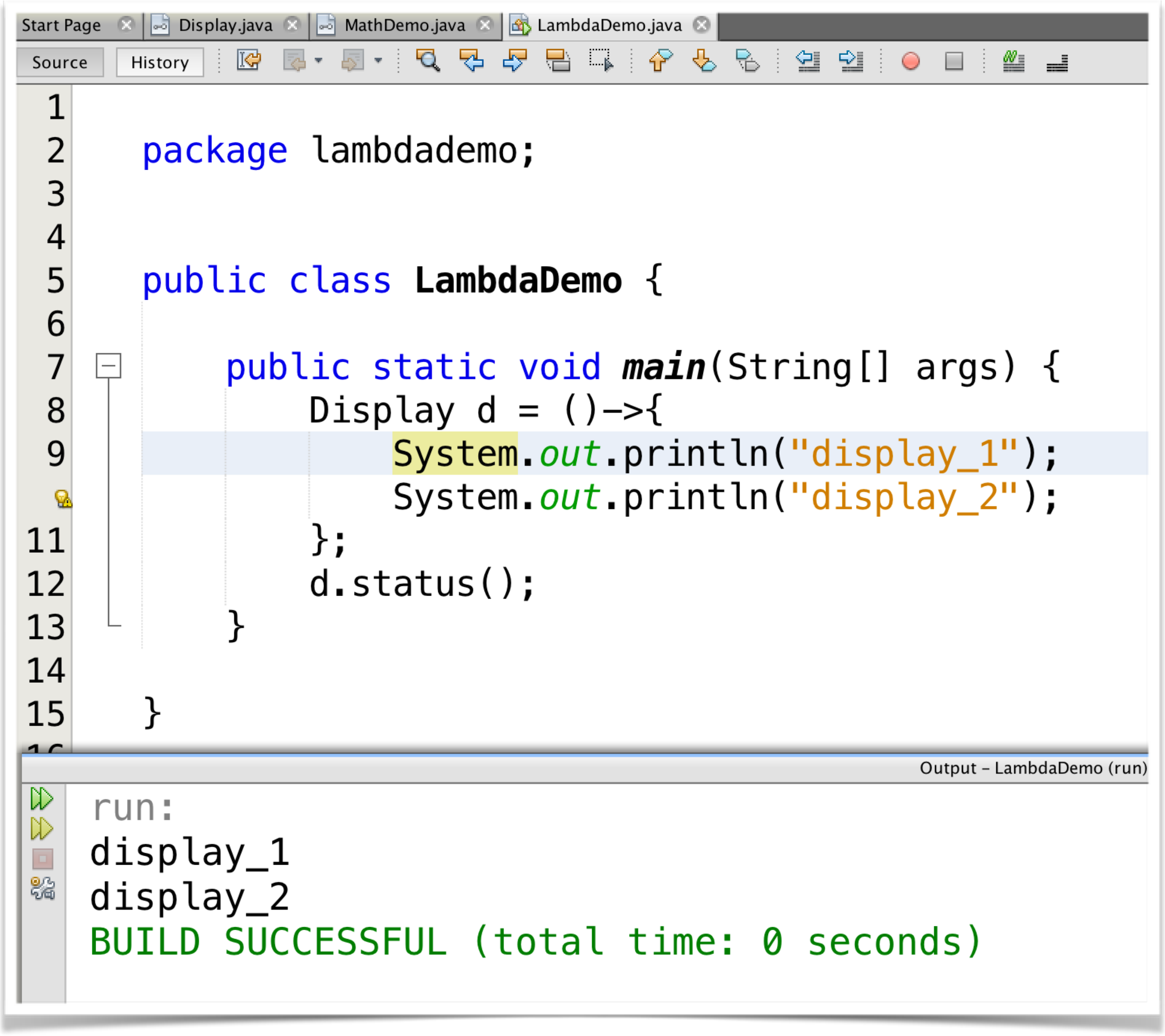
optional braces - only if one statement



```
package lambdademo;


public class LambdaDemo {

    public static void main(String[] args) {
        Display d = ()->System.out.println("display");
        d.status();
    }

}
```

```
run:
display
BUILD SUCCESSFUL (total time: 0 seconds)
```

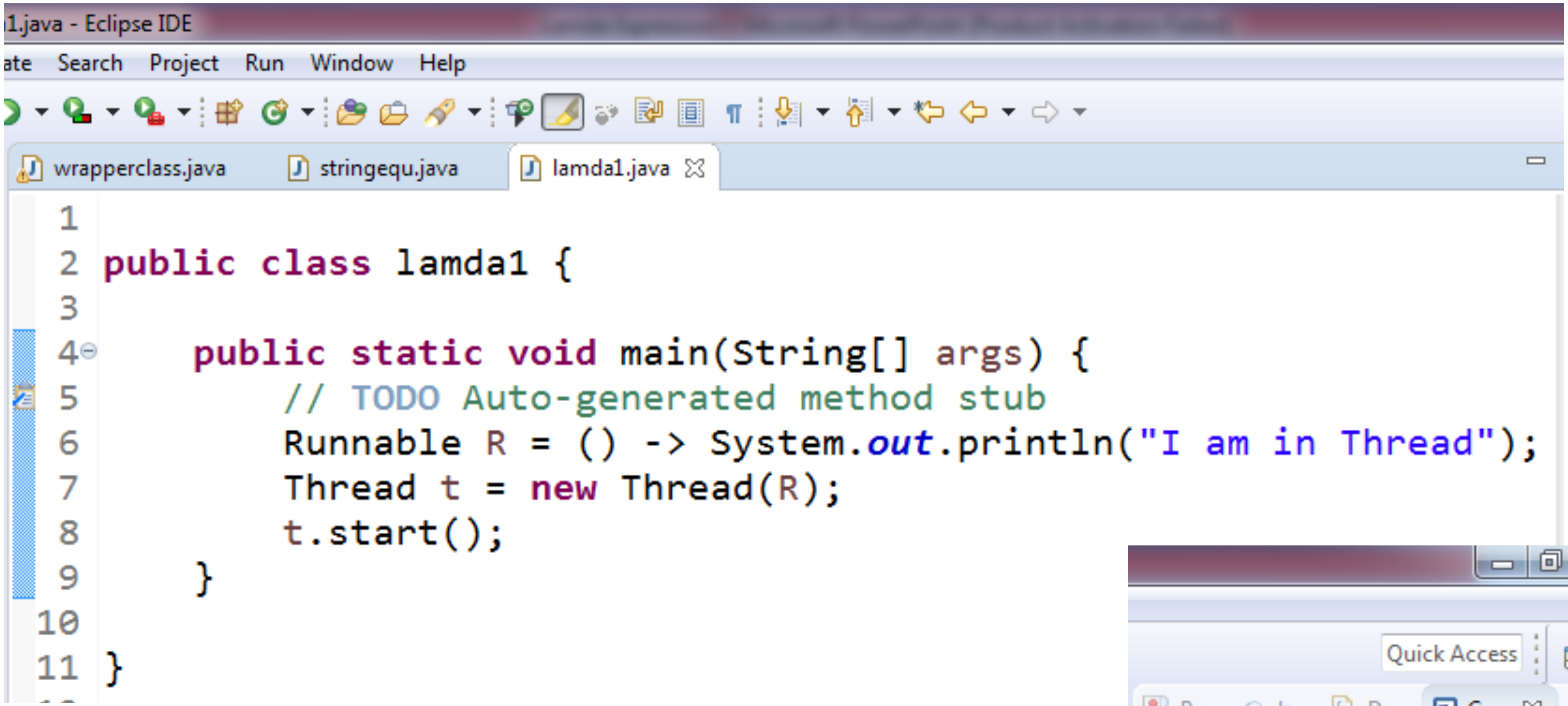
Mandatory braces - if more than one statement



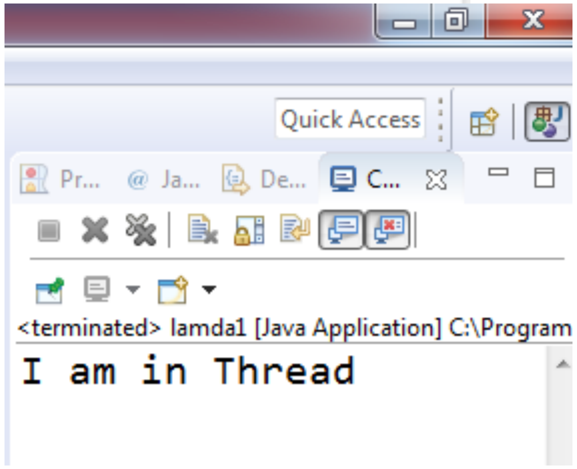
```java

package lambdademo;


public class LambdaDemo {

    public static void main(String[] args) {
        Display d = ()->{
            System.out.println("display_1");
            System.out.println("display_2");
        };
        d.status();
    }

}
```

```
run:
display_1
display_2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Thread – Runnable Interface

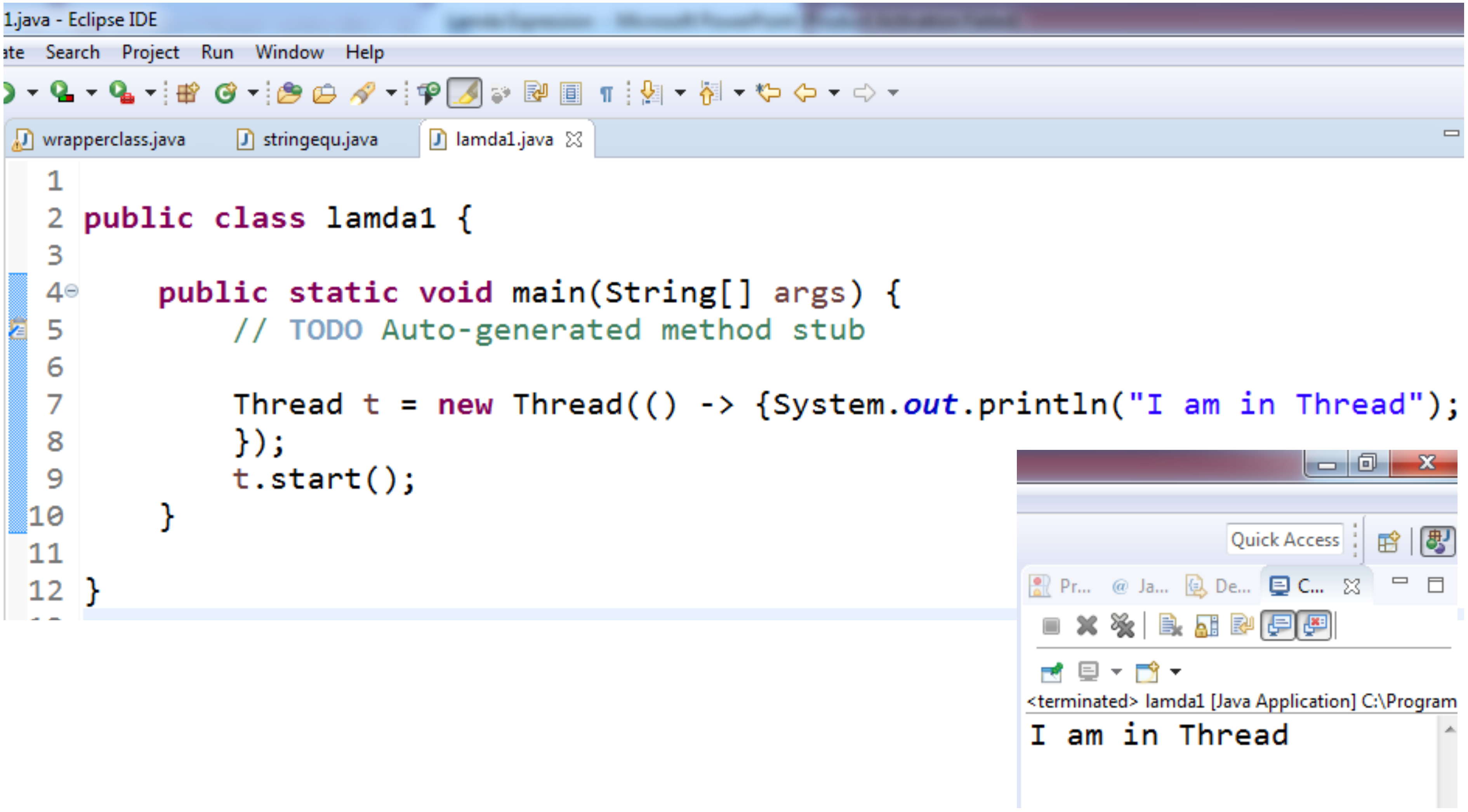


```java

public class lamda1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Runnable R = () -> System.out.println("I am in Thread");
        Thread t = new Thread(R);
        t.start();
    }

}
```



```
<terminated> lamda1 [Java Application] C:\Program
I am in Thread
```

Runnable Interface within Thread



```
public class lamda1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Thread t = new Thread(() -> {System.out.println("I am in Thread");
        });
        t.start();
    }

}
```

```
<terminated> lamda1 [Java Application] C:\Program
I am in Thread
```