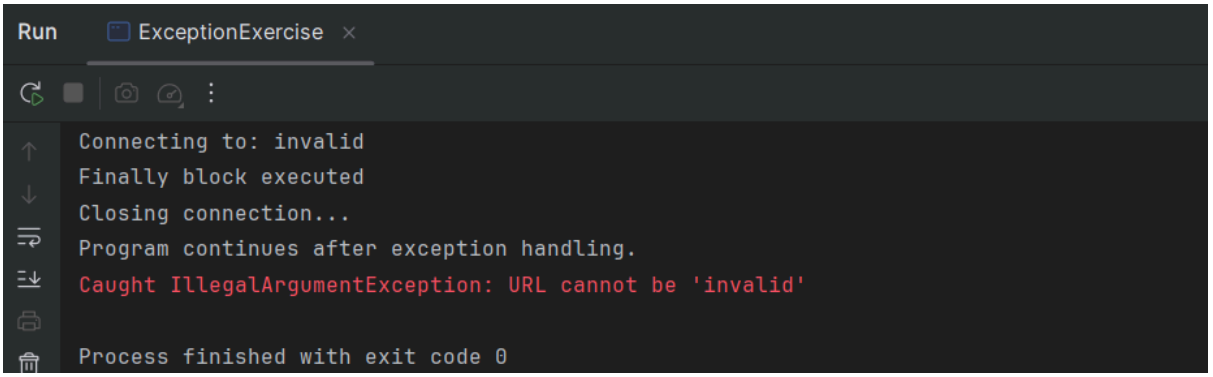
Step 1 — Run Starter Code ("invalid")

Code:

connection.connect("invalid");

Output:



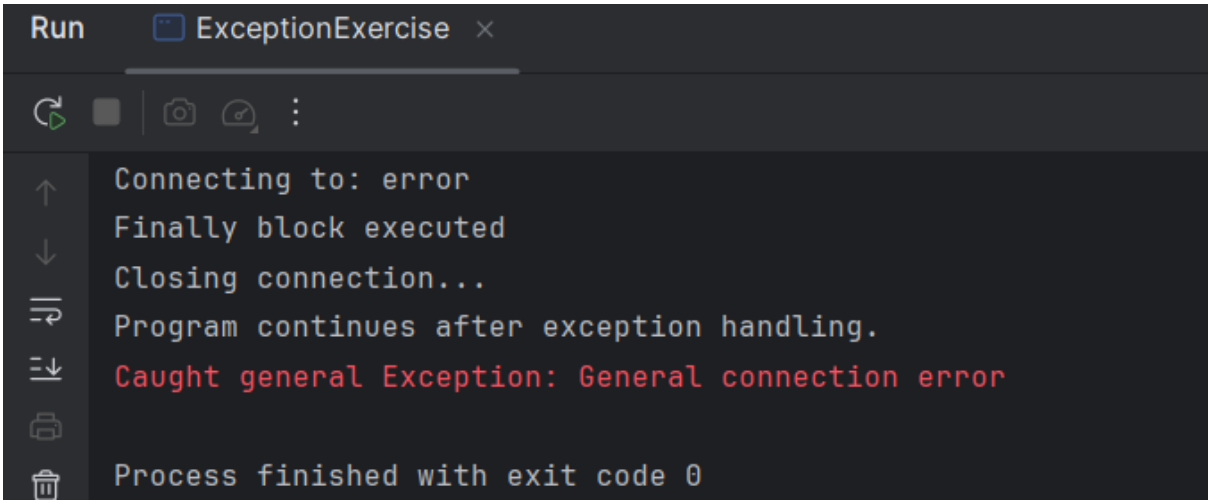
> When we use `"invalid"`, the program throws an **Unchecked Exception** (`IllegalArgumentException`).
>
> The program immediately goes to the **first catch block** that matches the exception type.
>
> The **finally block** as we Know always runs no matter what.
>
> After handling the exception, the program continues running normally.

Step 2 — Use "error"

Code:

connection.connect("error");

Output:



When we use `"error"`, the program throws a **Checked Exception** (`Exception`).

It is caught by the **general catch block**.

The **finally block** still runs no matter what.

After handling the exception, the program continues normally.

Step 3 — Use a Valid URL ("abc")

Code:

connection.connect("abc");

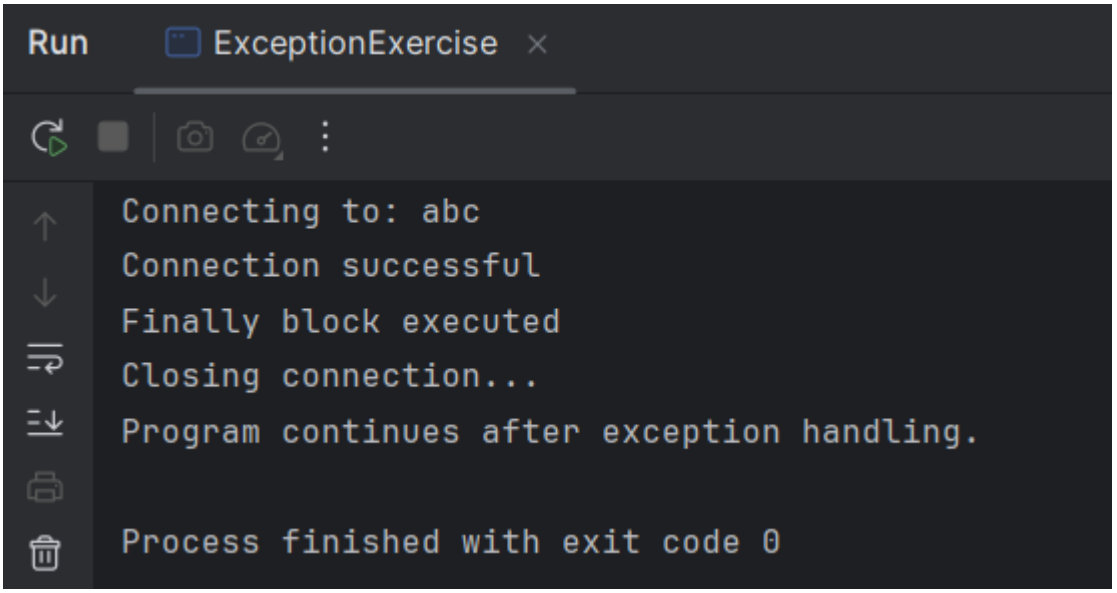
Output:

Connecting to: abc
Connection successful
Finally block executed
Closing connection...
Program continues after exception handling.

No exceptions are thrown when we use a valid URL like `"abc"`.

The program runs normally, and the **finally block** still executes.

After that, the program continues running as usual.



Step 4 — Swap Catch Order

Code:

```
try {
    connection.connect("invalid");
} catch (Exception e) {
    System.err.println("Caught general Exception: " + e.getMessage());
} catch (IllegalArgumentException e) {
    System.err.println("Caught IllegalArgumentException: " + e.getMessage());
} finally {
    System.out.println("Finally block executed");
    connection.close();
}
```

Compiler Error:

error: exception IllegalArgumentException has already been caught

`IllegalArgumentException` is a **subclass of** `Exception`.

If we catch the general `Exception` first, the specific `IllegalArgumentException` catch **can never run**, so the compiler gives an error.

```
C:\Users\ReactUser\Raghad_java-github-lab\assignment-3\src\ExceptionExercise.java:12:9
java: exception java.lang.IllegalArgumentException has already been caught
```

Step 5 — Remove `throws Exception` from connect

Code:

```
public void connect(String url) {
    if("error".equals(url)) {
        throw new Exception("General connection error");
    }
}
```

Compiler Error:

```
C:\Users\ReactUser\Raghad_java-github-lab\assignment-3\src\ExceptionExercise.java:45:13
java: unreported exception java.lang.Exception; must be caught or declared to be thrown
```

Checked exceptions must be either **declared** with `throws` or **handled** using a try-catch block.

If we remove `throws Exception` and don't handle it inside the method, the **compiler gives an error**.

Step 6 — Add Method Throwing Checked Exception

Code:

```
public void testChecked() throws Exception {
    System.out.println("Testing checked exception...");
    throw new Exception("This is a test checked exception");
}
```

In main:

```
try {
    connection.connect("abc");
    connection.testChecked();
} catch (IllegalArgumentException e) {
    System.err.println("Caught IllegalArgumentException: " + e.getMessage());
} catch (Exception e) {
    System.err.println("Caught general Exception: " + e.getMessage());
} finally {
    System.out.println("Finally block executed");
    connection.close();
}
```

Output:

The method `testChecked()` **throws a checked exception**.

When we call it in `main`, it **must be inside a try-catch** block.

The exception is **successfully caught** by the general catch block.

Key Concepts

throw vs throws

`throw` → actually **throws the exception object now**.

`throws` → **declares** that the method may throw an exception later.

`throw` يرمّي الاستثناء مباشرة، `throws` يخبر الكومبايلر إن الميثود ممكن ترمي استثناء.

Exception hierarchy

`Throwable` → the top class.

`Exception` (checked) & `RuntimeException` (unchecked) →

`Error` (serious problems).

كل Exception ترجع في النهاية إلى `Throwable`.

Checked vs Unchecked Exceptions

Checked → must **handle** or **declare** (e.g., `Exception`).

Unchecked → runtime only, no declaration needed (e.g., `IllegalArgumentException`).

Checked exceptions لازم نكتبها أو نمسكها بالكود، Unchecked ممكن تصير وقت التشغيل بس ما يلزم تصريح.

finally block

Always executed **after try/catch**.

Ensures cleanup like **closing connections**.

return أو exception: ينفذ دائمًا مهما حصل

Order of catch blocks

Catch **specific exceptions first**, then general later.

Catching parent first makes child **unreachable**, compiler error.

لازم الترتيب يكون من الخاص إلى العام.