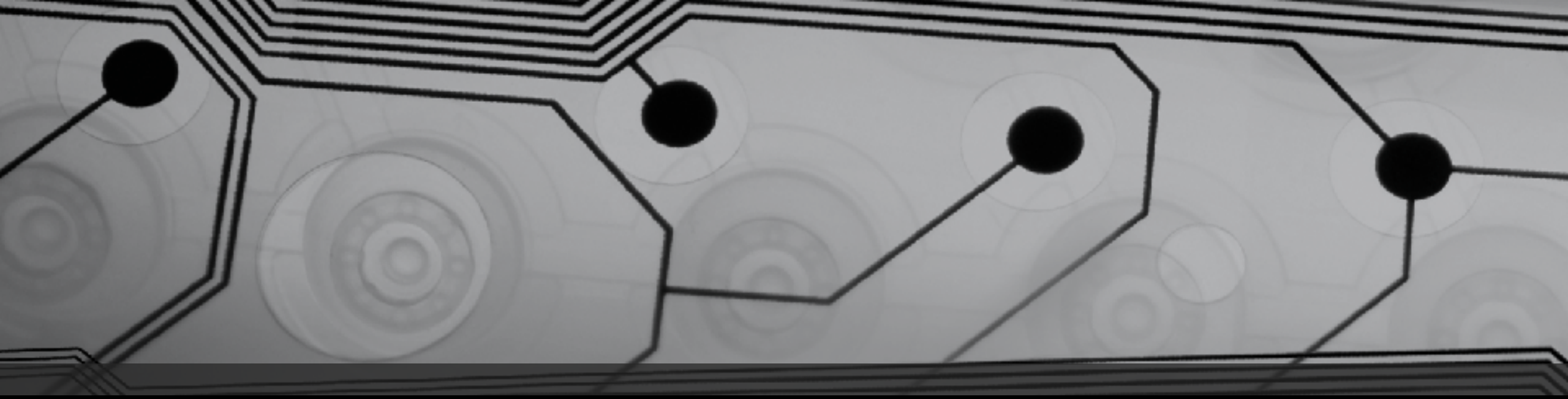
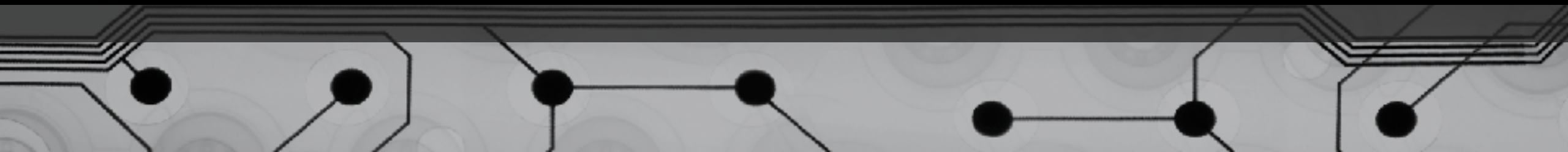


CMSC 131

Fall 2018



Break

```
while(…) {
    …
     if (…) {
        break;
     }
    …
}
```

- Used with loops
- Why might this be “questionable” style?
- What happens with nested loops?
- Are there examples where “break” works well?

Continue

```
while(…) {
    …
     if (…) {
        continue;
     }
    …
}
```

- Used with loops
- Why might this be “questionable” style?
- What happens with nested loops?
- How does this work with for-loops?
- Are there examples where “continue” works well?

Example

BreakContinueExample.java

Runtime Errors

Examples of Runtime Errors

Let's consider a concrete case:

Example: TriangleAreaCalculator.java

What could go wrong here?

What should be done?

- Error message and terminate?
- Return an "error code"?
- "Throwing an exception".

Exceptions

- What is an “exception”?

In Java:

1. When something unexpected occurs, we “throw” an exception
   - Demonstrate with TriangleCalculator.java
2. JVM looks for “handler”
   - Looks in current method. If not found, pops this frame off the call stack and looks in the next one. Etc.

   If “handler” is found it runs.

   If “handler” is never found, program terminates.

How does a “Handler” Look?

```
try {
    <Code that might throw an exception>
} catch(ExceptionClass e) {
    <Put handler here>
}
```

Common Types for Exceptions

- NullPointerException
- ArithmeticException
- IllegalArgumentException
- RuntimeException (plain vanilla one)
- Many others
- You can create your own!

Examples

- RandomTriangleMaker.java
- CalorieCounter.java

Observations

Where are exceptions thrown?

- In code you have written
- In code written by someone else that you are calling
- By the JVM itself, internally

Why is this better than just returning an “error code”?