Bugs the Java compiler can’t catch

(1) Misunderstanding how an API works

(2) Not getting all your cases right (“a list is either an empty-list or ...”)

(3) Using the wrong values in the wrong places

(4) Inheritance / overriding not going where you expected
Write more unit tests

(1) Misunderstanding how an API works
You can write unit tests for system-provided code, not just your own. Convince yourself you know how it’s supposed to work; experiment!

(2) Not getting all your cases right (“a list is either an empty-list or ...”)
Write tests for each case

(3) Using the wrong values in the wrong places
Write tests that exercise your “internal” helper functions

(4) Inheritance / overriding not going where you expected
Not helpful on its own, but...
Debug printing / logging

(1) Misunderstanding how an API works
Print what goes in and what comes out
Most things in Java do something useful with .toString()

(2) Not getting all your cases right ("a list is either an empty-list or ...")
Print something as you go

(3) Using the wrong values in the wrong places
Print the values at each stage

(4) Inheritance / overriding not going where you expected
Print something on the way into each method
Single-step debugging

(1) Misunderstanding how an API works
Step through mysterious code, gain zen understanding of how it works
Of course, you’d rather code be so beautiful it’s easy to read

(2) Not getting all your cases right (”a list is either an empty-list or...”)
Make unit tests, set breakpoints

(3) Using the wrong values in the wrong places
Set breakpoints, inspect values manually
IntelliJ calls your toString() methods and shows you the results

(4) Inheritance / overriding not going where you expected
Set breakpoints, see if program ever gets there
Printing vs. logging

Old school
System.out.println("running constructor");

New school (we borrow from Android style logs; you’ll get this next week)
import edu.rice.Log;

class Foo {
    private static final String TAG = "Foo";
    public Foo() {
        Log.i(TAG, "running constructor");
    }
}
Log levels, etc.

Information, warning, error
Log.i(), Log.w(), Log.e()
You can adjust the level via Log.setLogLevel() and see more or less output
Optional lambda “lazy” form if you want “fast” when logging is disabled
Log.i(TAG, () -> "current input: " + input.toString());

Why private static final String TAG in each Java class?
If you copy-and-paste the code somewhere else, the TAG will always be correct

Why structured Log vs. unstructured System.out.println()?
Android Studio (IntelliJ for Android) has built-in search features, color-coding
Optional plug-ins for your IntelliJ

Sometimes, logs are all you get (cloud computing, smartphone apps, etc.)
Live coding

The midpoint line drawing algorithm; debugging its implementation