# S4 Unit testing

Testing
Creating unit tests
JUnit

# Why test?

- Well-designed tests are a key tool to detect and isolate errors in code.
- Rigorous testing can give some confidence that code will work correctly outside the specific cases that we have tested.

# Types of Tests

- **Unit tests**: Test individual “units” / “modules” for correctness.
  - In Java (OO program) a unit is typically a **method** or **class**.
  - Check that “building blocks” are functioning correctly.
  - Essential in bug hunting: narrow down what unit is faulty.
- **Integration tests**: Test integration of multiple modules.
  - Expose problems with interface of modules and interactions between them.
- **System tests**: End-to-end complete system
  - Checking it meets its requirements (not only correctness).

# Types of Tests

- **Black box**: Tests are written from the specification of how the unit or system should work, without knowledge of its implementation.
  - Can expose flaws in the (shared understanding of the) specification.
- **White box**: Tests are written with knowledge of the unit/system implementation.
  - Can ensure coverage of branches/cases within the code.

# What makes **good** unit tests?

- Respect documented assumptions and limitations.
  - Behaviour outside of documented limitations is *undefined*: it cannot be tested.
- Simple
  - Be able to determine the correct answer/output without relying on implementation!
  - Avoid false positives (test fails on correct code).
- Deterministic (repeatable)
  - Same test of same code always yields same result.
  - If test/code depends on environment (e.g., computer's workload, time of day), make the dependencies explicit (and controllable, if possible).

# What makes **good** unit tests?

- Coverage
  - Tests exercise all code components (for example, all paths in a branching statement).
  - Combinations (for example, equal and opposite signs, equal and different subclasses).
- Cover edge cases
  - Values at the limit (empty sequence or string, zero, very large or very small) – but still within limitations of the class/method!
  - Any value that requires special handling the code.

# JUnit

- Unit testing framework for Java
  - Useful for bug detection, isolation and regression testing.
  - `https://junit.org/junit5/`
- Integrated into IntelliJ
  - But requires some setup!

# JUnit

- Instance methods annotated with `@Test` define tests.
  - Within test methods, **assertions** define conditions to be checked.
  - Other annotations can be used to configure tests, e.g., `@Order`, `@Timeout`
- When JUnit is run on a class, all tests are run and a report is generated.
- Test class vs. Test-in-class:
  - Test methods can be embedded in a class: when the class is used normally, test methods are ignored.
  - The class must be instantiable.