Unit Testing

1. junit5 api

Foundation of Quality Assurance and Testing

1. static testing(Spot bugs) and dynamic testing(Randoop)
2. software faults, errors and failures
    a. faults: software design faults, static
    b. errors: internal incorrect, when executing the faults, dynamic
    c. failures: external incorrect, different with the requirements, dynamic
3. fault and failure model: reachability, infection(internal), propagation(external)
4. **coverage**:
    a. strctual coverage
        i. **node coverage, edge coverage, edge–pair coverage, complete path coverage**
        ii. **test requirements for edge–pair coverage**: three nodes
        iii. **test path**: start at initial node, end with terminated node
    b. simple path and prime path
        i. simple path: A path from node ni to nj is simple if no node appears more than once, except possibly the first and last nodes are the same
        ii. prime path: A simple path that does not appear as a proper subpath of any other simple path(it also means the longest terminates and cycles) === (in cycle write cycle and terminate, otherwise write cycles)
        iii. **prime path coverage**
    c. Date flow
        i. DU pair: from def to use
        ii. DU path: a **simple path** from def to use with no other def
            1. du(i,j,x) = []
            2. du(i,x) = {[],[]}
        iii. All–defs coverage: every def reach a use
        iv. All–users coverage: every def reaches all possible use
        v. All–dupath coverage: all pathes between defs and uses

Object–Oriented Software Development

1. **activity diagram**
2. **user case diagram**:
3. **unified software development process**:
    a. incremental and iterative
    b. user case driven
    c. architectural centric
4. **use the below relationship to build ?**: todo
    a. association, aggregation(a special relationship of association), composition(a special relationship of aggregation)
5. **covariance Problem**: todo

Automated Fault Detection and Diagnosis Technology

1. automated software test
    a. random testing(dynamic): Randoop
    b. symbolic execution(static)
        i. using liner constraint solver
    c. concolic execution(dynamic) : todo
        i. evosuite
2. automated fault location
    a. tarantula
    b. ochiai
3. automated program repair
    a. mutation–driven
    b. semanics–driven: ?
    c. prioritization of plausible fixes
        i. heuristics
        ii. machine learning

Execise

1. a test case does not execute the fault: **means execute the fault code**
2. a test case execute the fault, does not result in an error state: **means execute the fault code but not cause a unexcept state.**
3. a test case results in an error, does not a failure: **means causing an unexcept state but repaired in the subsequence processing and the result is correct.**
4. activity diagram