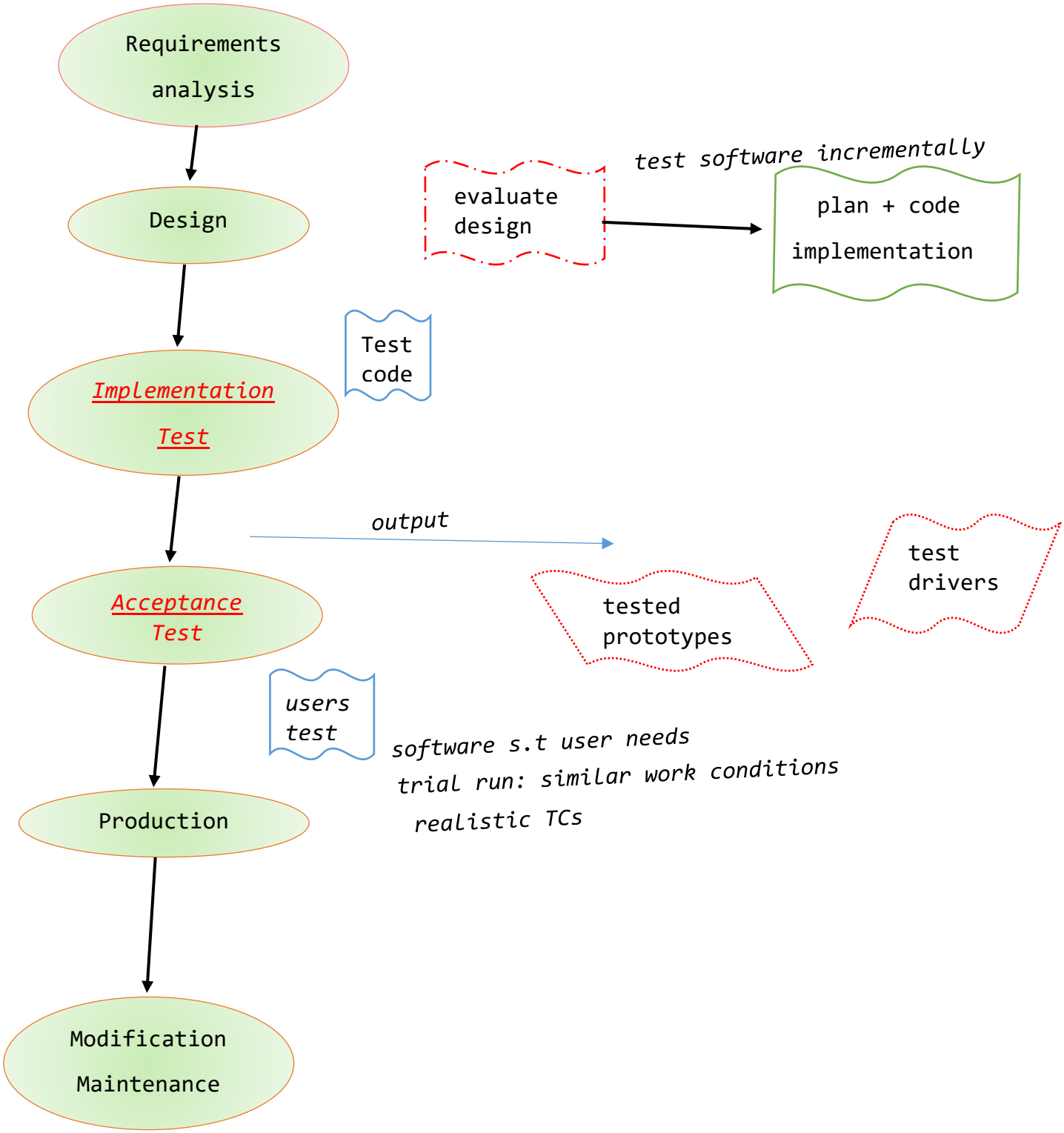

Requirements analysis
Design
Implementation Test
Acceptance Test
Production
Modification Maintenance
evaluate design
test software incrementally
plan + code implementation
Test code
output
tested prototypes
test drivers
users test
software s.t user needs
trial run: similar work conditions
realistic TCs

Code Testing

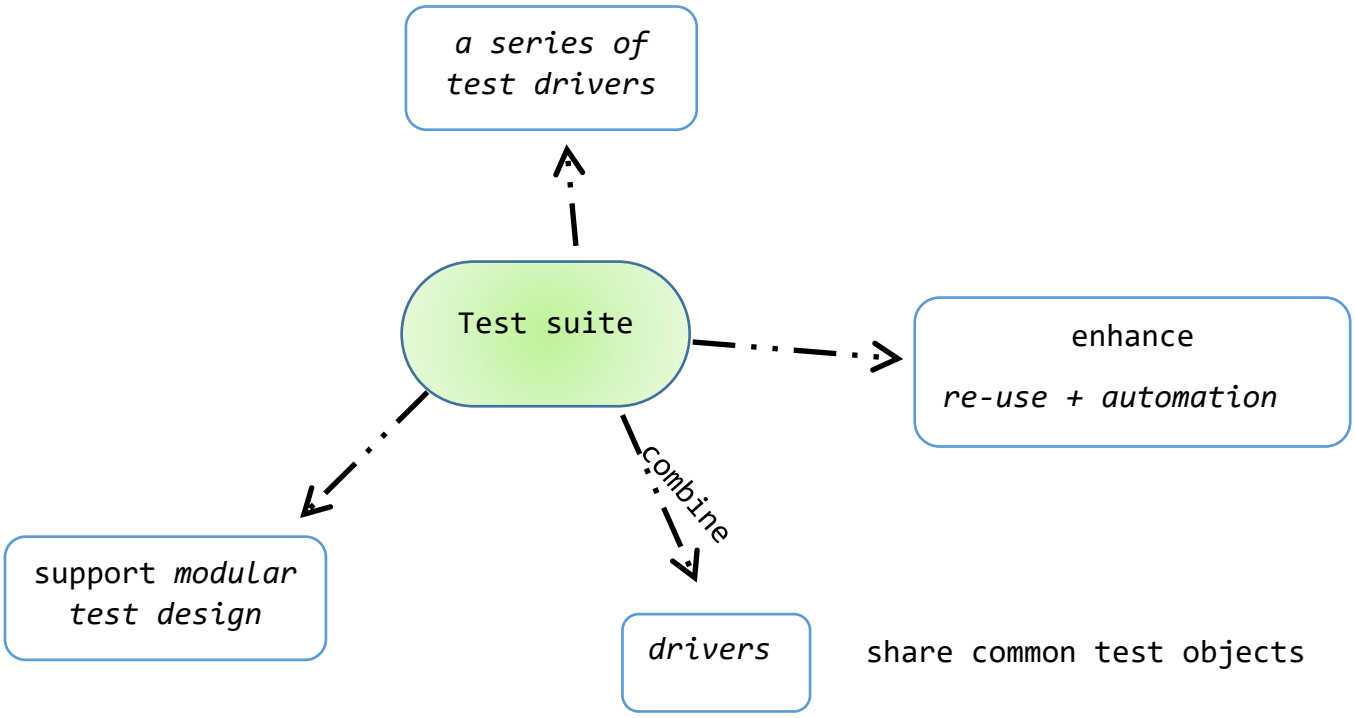
Unit testing: each module (in solation), can re-use test drivers
(esp. top-down plan)

Integration testing: a group of modules ← unit test procedure
can use top-level modules

Regression testing: each modification → re-run tests
(error correction)
➔ ensure code integrity

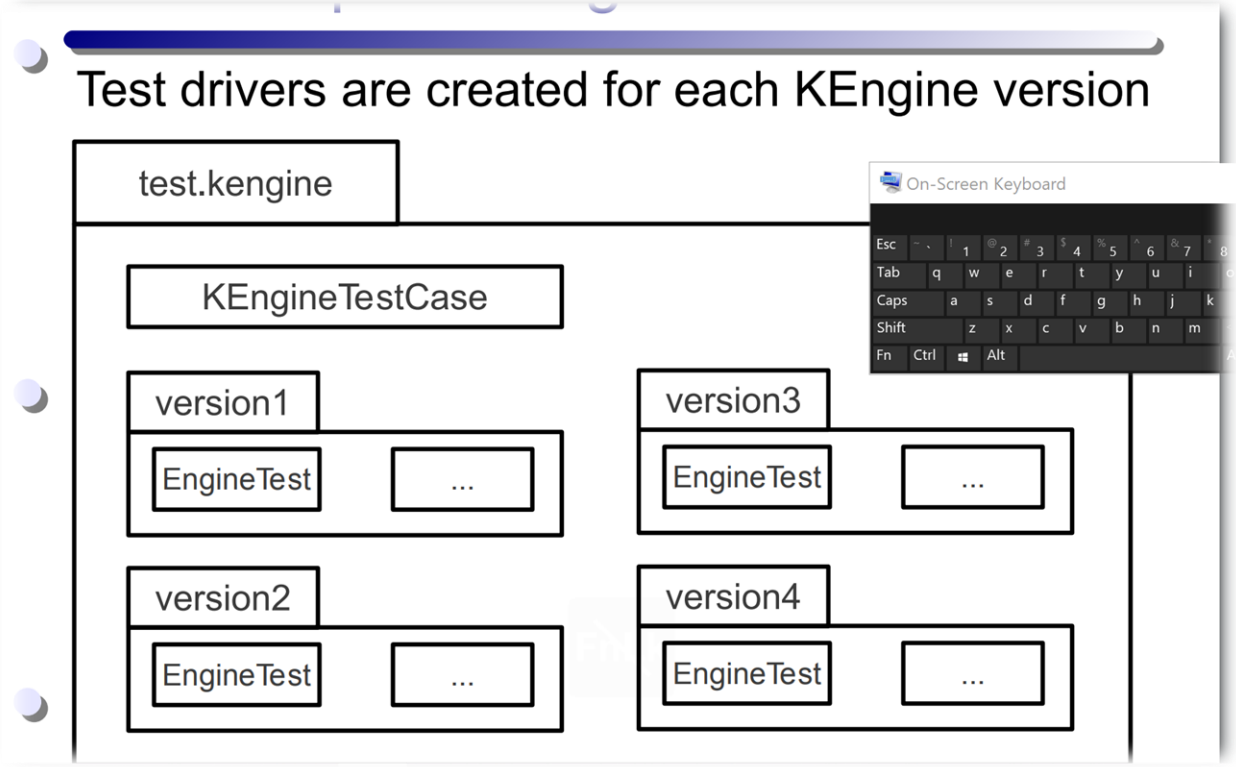
Test Driver design

organise -> type hierarchies → Reuse



each *test driver*
→ a *unit test*

Engine
test drivers created → each KEngine version

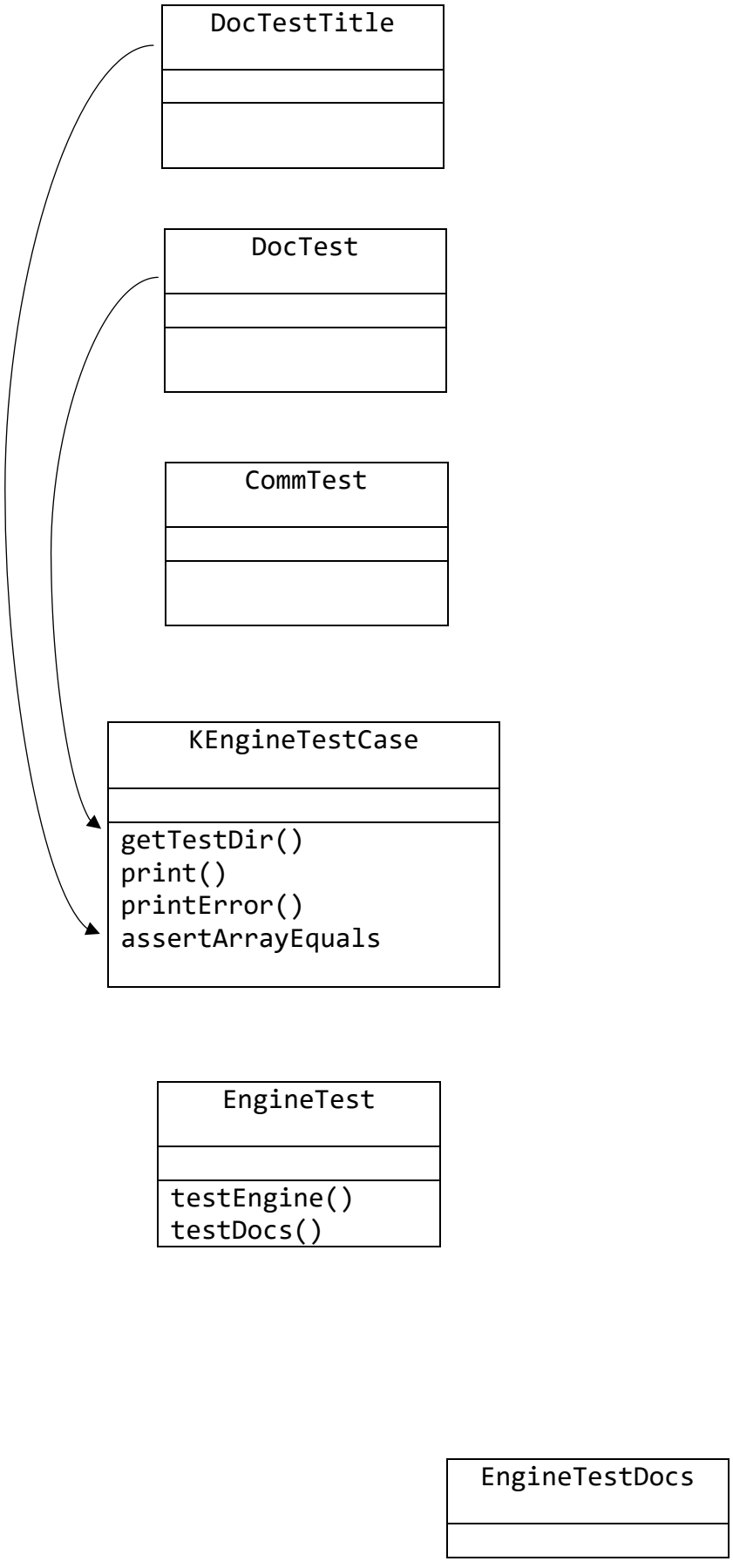


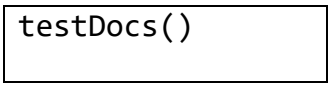
Test Iteration 1

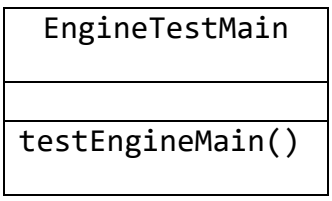
Core class: Doc Comm Engine

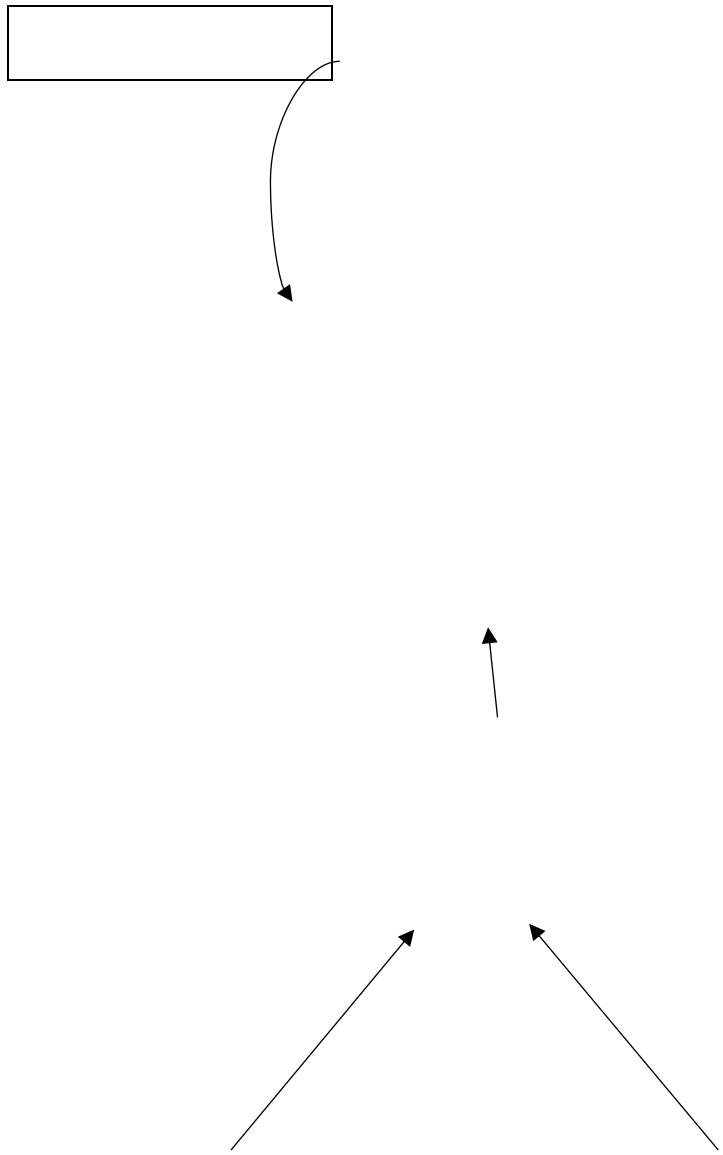
Stub class: Helpers Query TitleTable WordTable

Test driver hierarchy

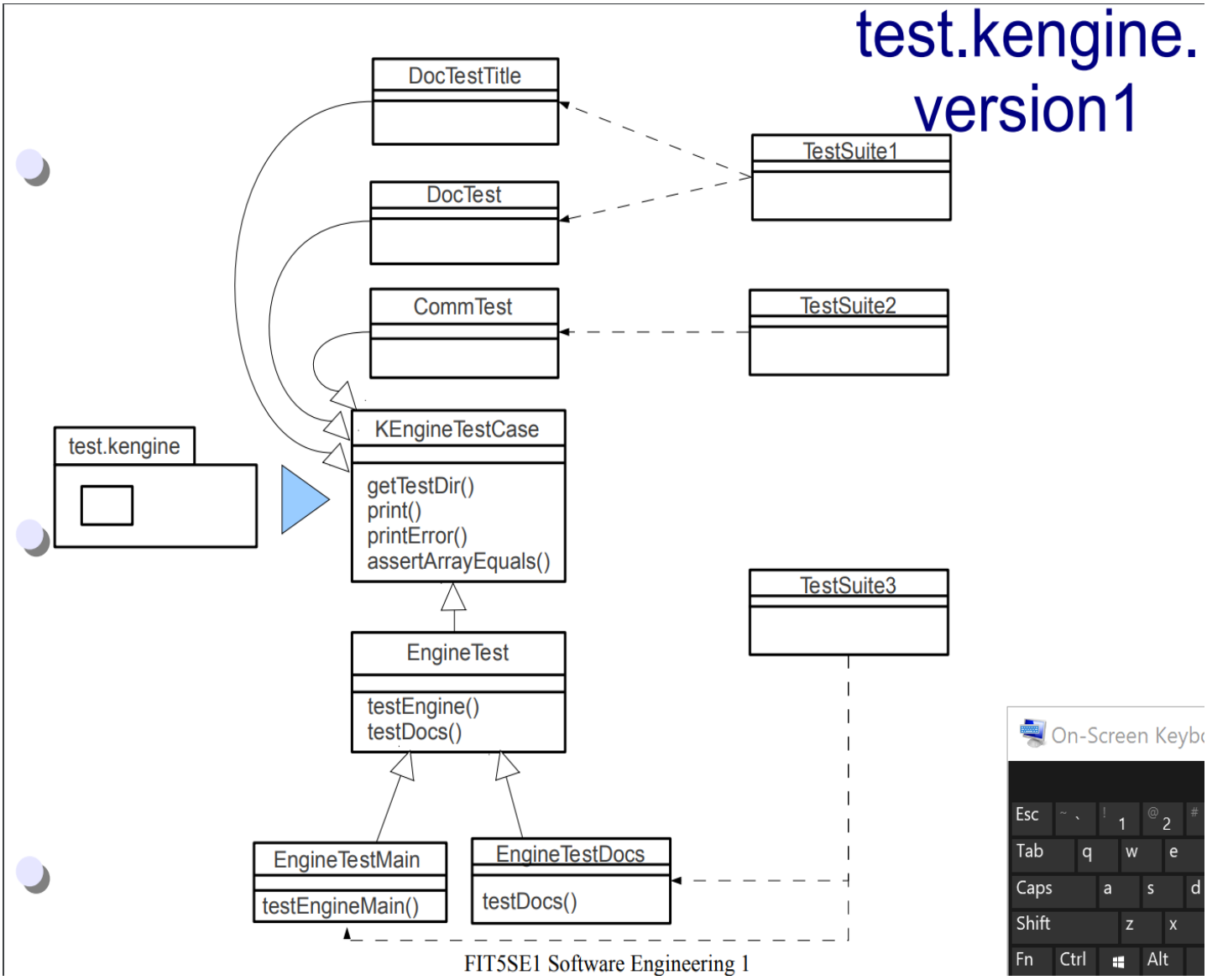



testDocs()


EngineTestMain
testEngineMain()



test.kengine.
version1



On-Screen Keyb
Esc
Tab
Caps
Shift
Fn
Ctrl
Alt

Engine Doc Comm

Engine

test method: Engine()

considers others: stubs

test driver: EngineTestMain

Doc

Doc
Doc(String) title(): String body(): String words(): Iterator

TCs: Doc(String) Dooc.title()

Doc(String) Doc.body() Doc.words()

Gr1: TCs : docs, titles: one of following cases

- null: ~~<title></title>~~ (tags)
- empty (“”): <title></title>

- single char: <title>a</title>
- a proper string:
  <title>” welcome to my page “</title>

Gr2: TCs: docs, body text: one of following cases

- null: ~~<body></body>~~ tags
- empty (“”): <body></body>
- only tags: <body><p></body>
- no tags: <body>some text</body> ~~<p>~~
- simple tags: <body><p>some text</body>
- complex tags:
  <body><p> <span …>some text</body>

Test Drivers:

Gr1: DocTestTitle.java

Gr2: DocTest.java

Test suite: TestSuite1

Comm

input String: valid folder path URL

/home/duclm/data/sites/hanu

TCs: U(valid) u(invalid)

- null
- empty String“”
- incorrect path

  u1 = /home/duclm/data/sites/void
- correct path

  U1 = /home/duclm/data/sites/hanu

Test Driver: CommTest

Test suite: TestSuite2

Engine
Engine() queryFirst(String): Query queryMore(String): Query findDoc(String): Doc **addDocs(String): Query**

Engine.addDocs: Doc Comm

Comm
getDocs(): Iterator

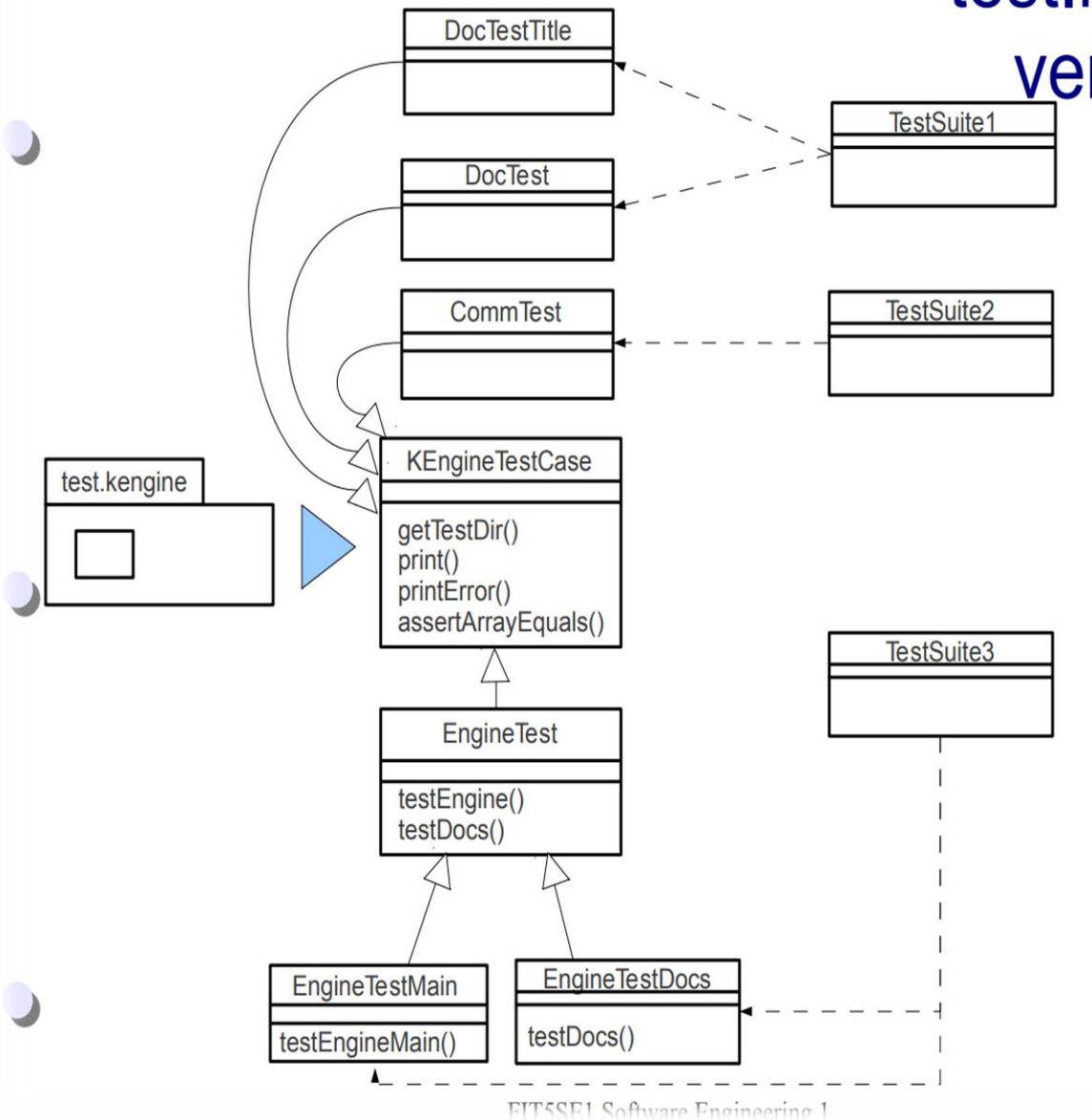
Doc
Doc(String) title(): String body(): String words(): Iterator

TCs: addDocs = Comm.getDocs() + {U1} (duplicate URL check)

Test driver: EngineTestDocs

Test suite: TestSuit3

test.kengine. version1



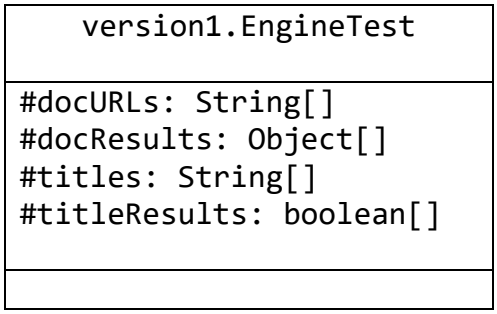
Test iteration 2

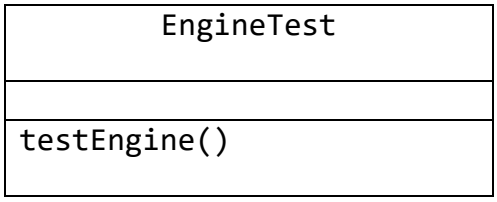
Helpers: common procedures → other classes

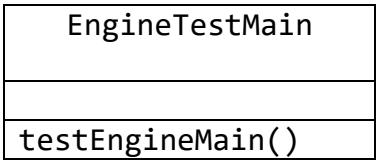
canon()

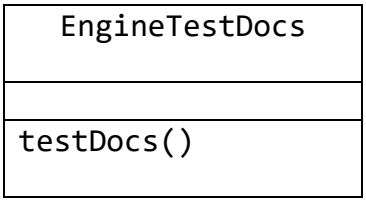
TitleTable: keep track of documents + titles

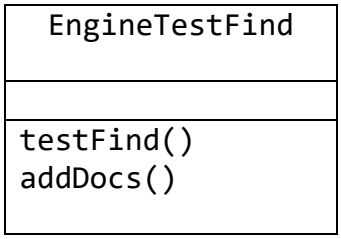
Test Driver hierarchy

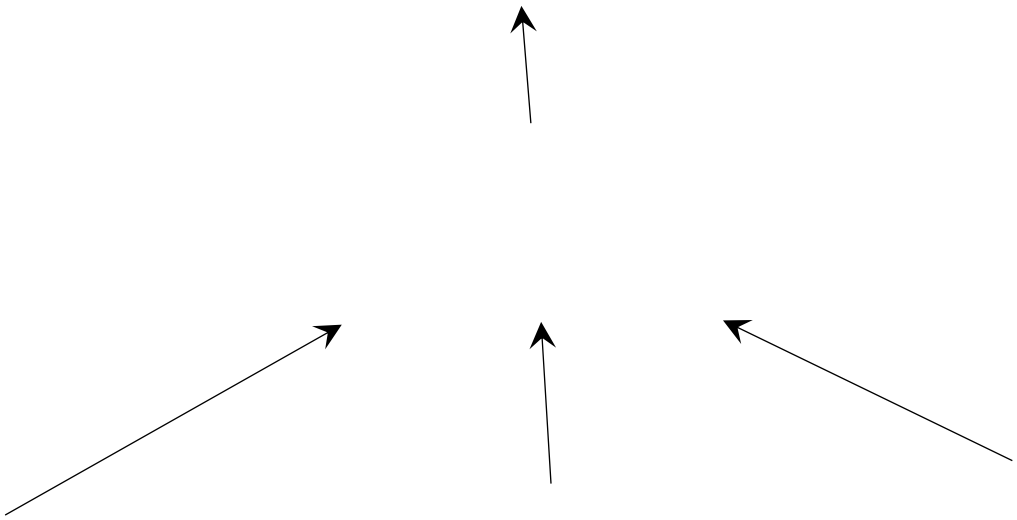




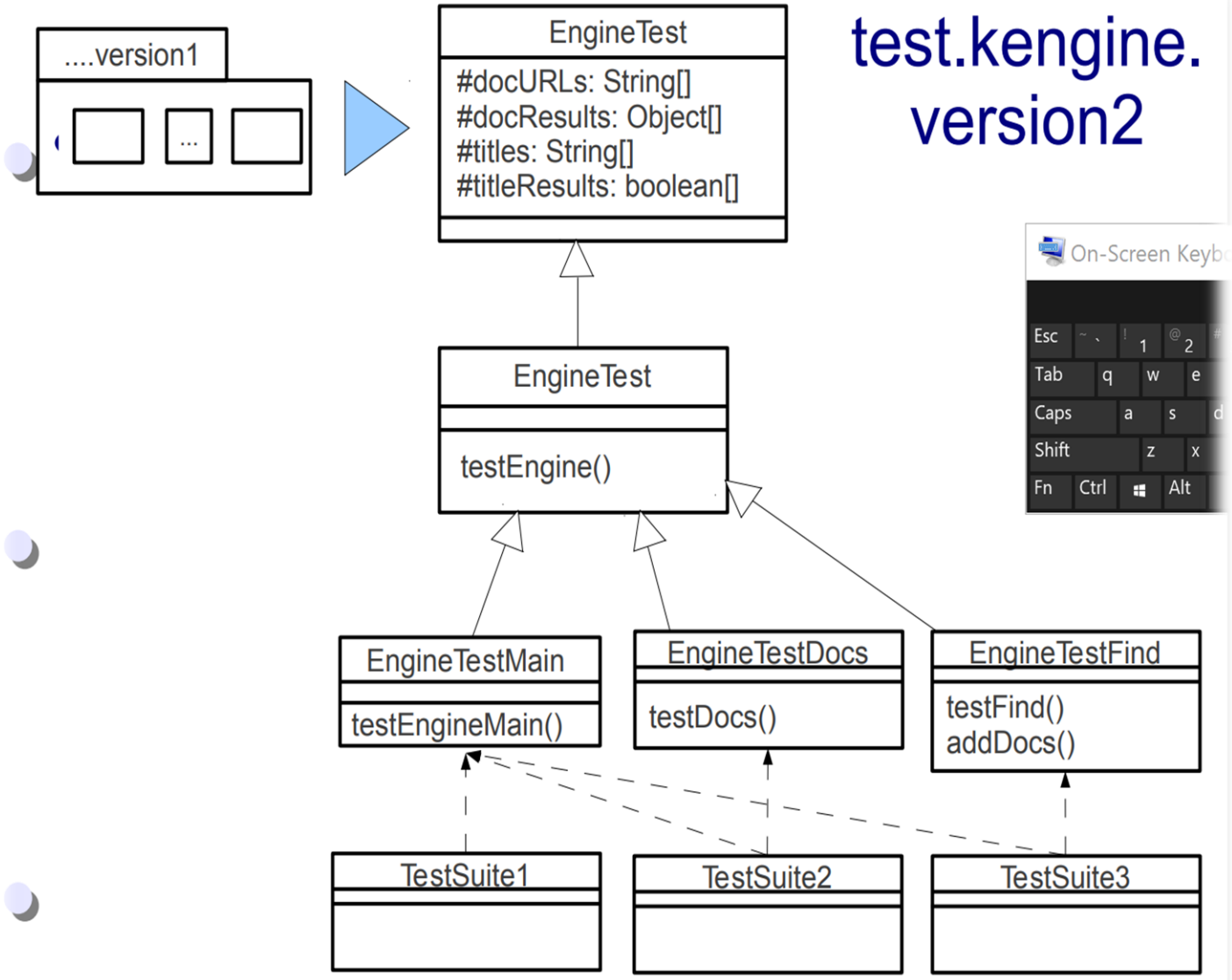








test.kengine.version2



Engine Helpers

Test Driver: EngineTestMain

tests version2.Engine

Test suite: TestSuite1

Helpers

Helpers
canon(String): String

BBT (black box test cases):

- null
- empty String (“”)
- single-char String: “a”
- normal Strings: “some text”

GBT (white box test cases):

- null
- not null: same as those for String.toLowerCase()

➔ skipped

TCs fall into one of GBT

Engine
Engine() queryFirst(String): Query queryMore(String): Query **findDoc(String): Doc** **addDocs(String): Query**

Engine.addDocs → TitleTable.addDoc + Helpers.canon

Engine.findDoc → TitleTable.lookUp + Helpers.canon

TitleTable
Hashtable docs
TitleTable() **addDoc(Doc)** **lookUp(String): Doc**

Helpers
canon(String): String

TitleTable TCs

Gr1: TitleTable() → a new empty table
addDoc() → a new <t, doc> entry

Engine.addDocs → TitleTable.addDoc

TCs: T(valid title) t(invalid title)

duplicate check: {T1, T1}
valid URL U1: 3 documents {d1, d2, d3}
d2.title = d3.title

Test Driver: EngineTestDocs
Test suite: TestSuite2

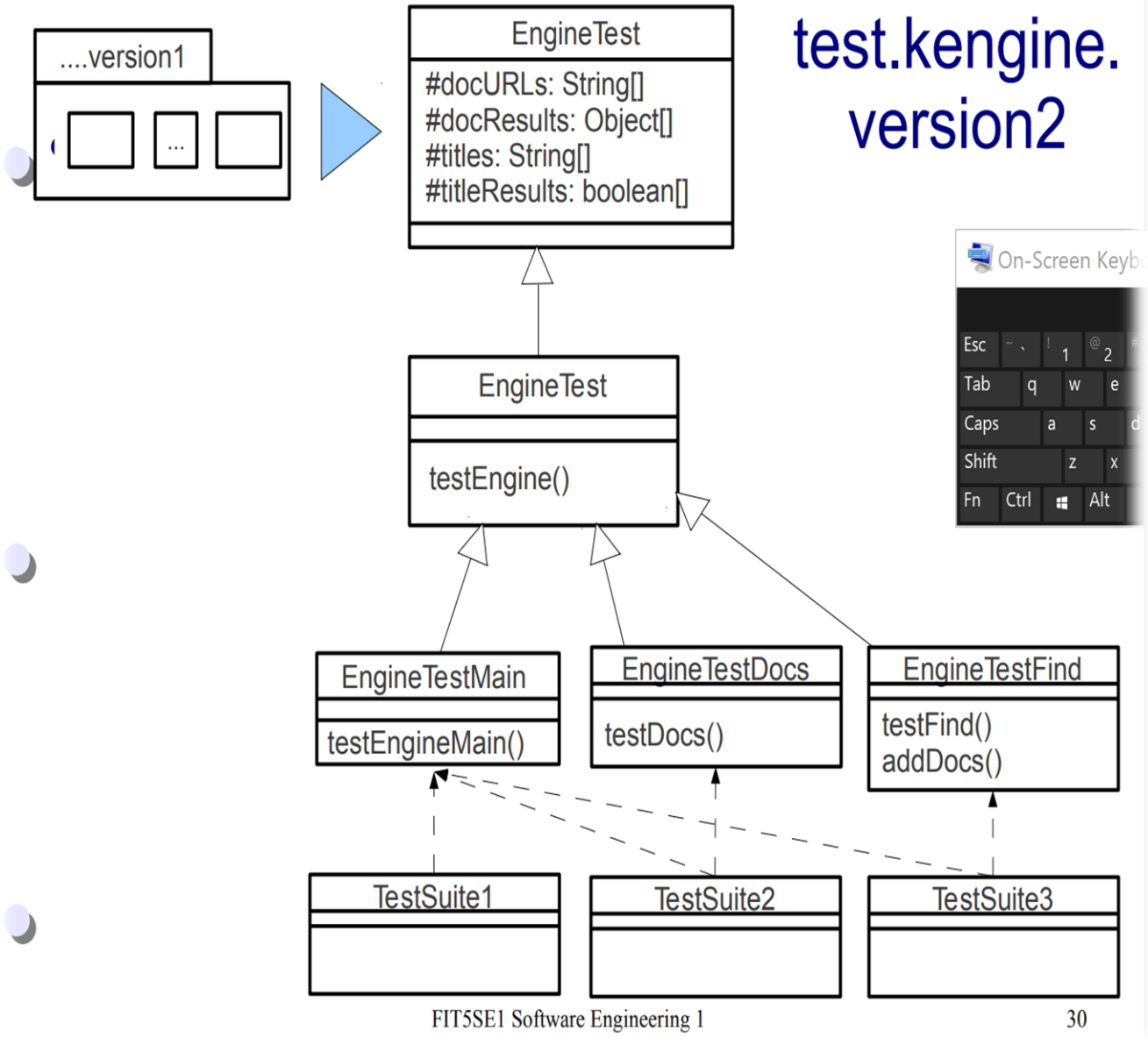
Gr2: *Gr1* → new entries
lookUp() → verifies inserted entries

Engine.findDoc → Title.lookUp

TCs: {null, “”, t1, T1}

Test Driver: EngineTestFind
Test suite: TestSuite3

test.kengine.version2



Test Iteration 3

DocCnt<Doc, count>

count: word frequency of some words in Doc

WordTable<String, Vector<DocCnt>>

keyword – DocCnt objects vector (contain)

Extra methods:

Engine(String): custom properties file

→ initialises a new Engine

getWords: check all words

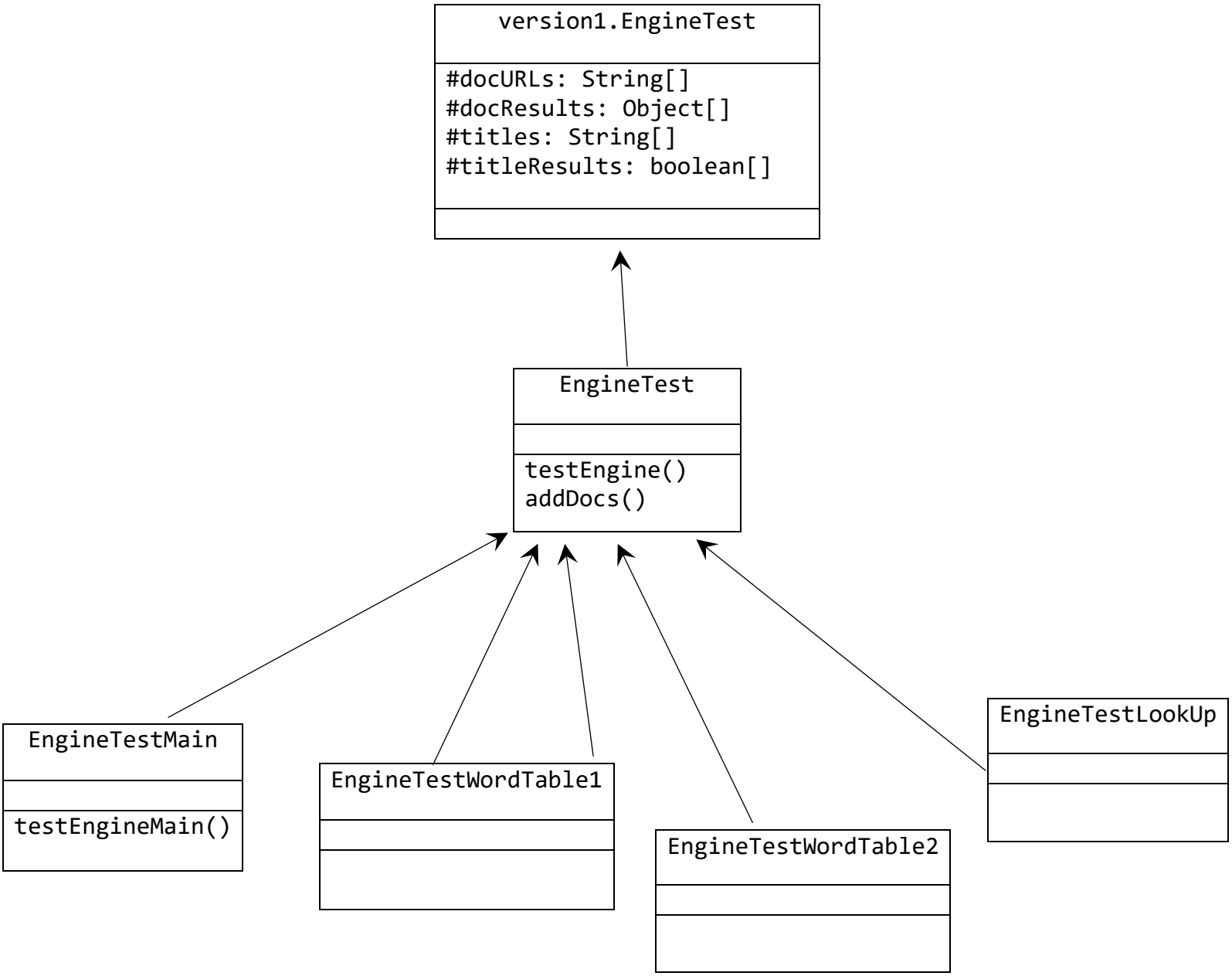
getNonKeys: check all non keys

Engine
Engine() **Engine(String)** queryFirst(String): Query queryMore(String): Query findDoc(String): Doc addDocs(String): Query **getWords(): String[]** **getNonKeys(): String[]**

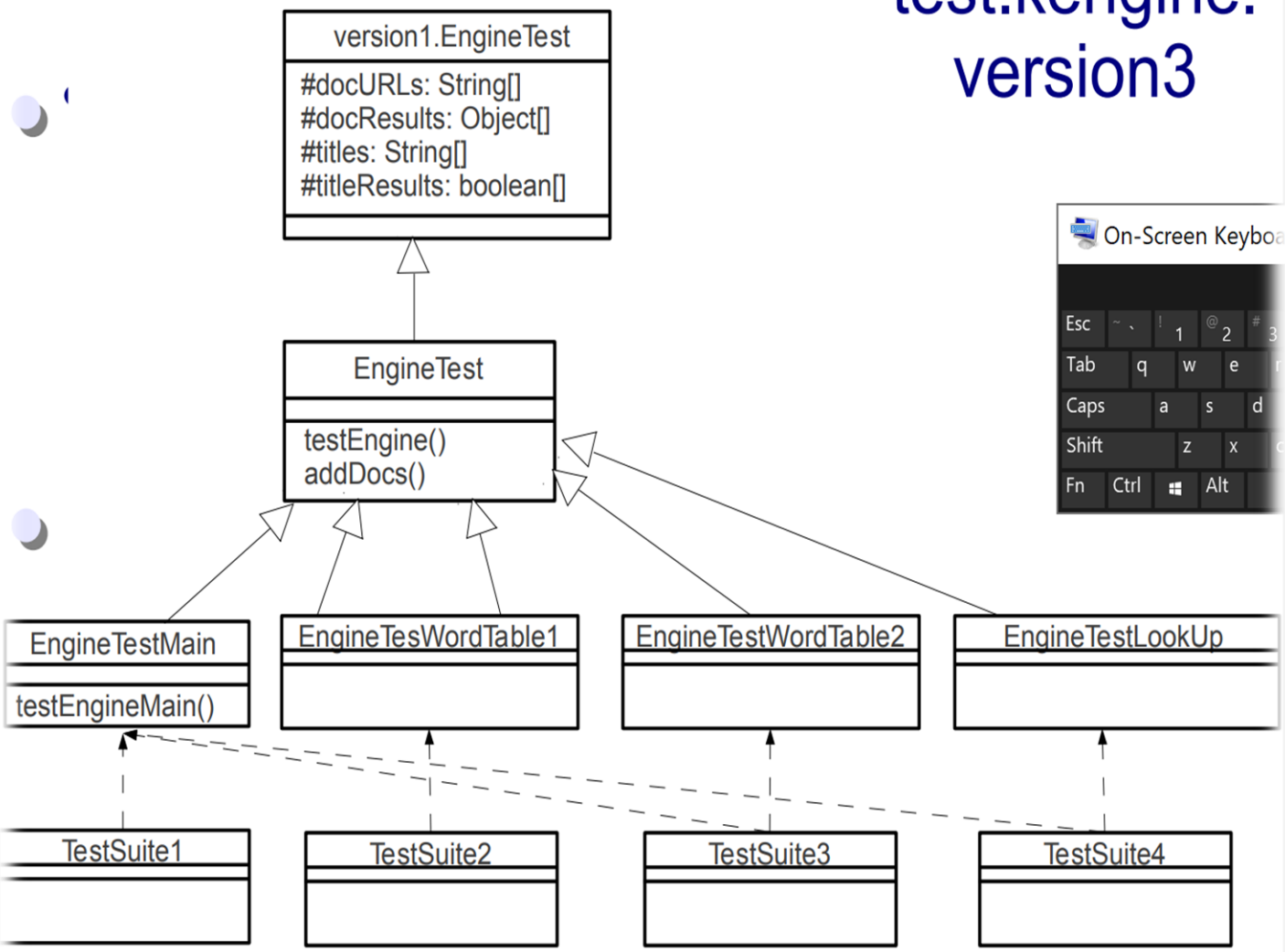
WordTable(String): custom nk.dat file

→ initialises a new WordTable

WordTable
Hashtable table
WordTable() **WordTable(String)** isInteresting(String): boolean lookUp(String): Vector addDoc(Doc): Hashtable


version1.EngineTest
#docURLs: String[]
#docResults: Object[]
#titles: String[]
#titleResults: boolean[]
EngineTest
testEngine()
addDocs()
EngineTestMain
testEngineMain()
EngineTestWordTable1
EngineTestWordTable2
EngineTestLookUp

test.kengine.version3



Engine DocCnt

Engine

EngineTestMain
testEngineMain()

Test method: Engine()

Test Driver: EngineTestMain

Test suite: TestSuite1

DocCnt

DocCnt
Doc d int cnt
DocCnt(Doc, int) getDoc(): Doc getCount: int toString(): String

rep + getters

TC + driver:

valid → Doc objects + counts

TestDriver: DocCnt objects ← test data

invoke getters → check results

WordTable
Hashtable table
WordTable() WordTable(String) isInteresting(String): boolean lookUp(String): Vector addDoc(Doc): Hashtable

~~test isInteresting()~~

local context

invoked only by addDoc()

Gr1: *WordTable(String)*

BBT:

- nk-file not exist
- nk-file exist: canonical test - words (different cases)

GBT: nk-file exist

- nk-file: empty
- nk-file: single word
- nk-file: 2 or more words

Test Driver: EngineTestWordTable1.java

Engine(String) Engine.getNonKeys()

Test suite: TestSuite2

Gr2: *Gr1* (WordTable(String)) + *addDoc*

BBT:

- d: (some) interesting words
- d: interestings in different cases

GBT:

- d: ~~words~~
- d: ~~interesting~~
- d: some distinct interestings
- d: repetitive interestings

Test Driver: EngineTestWordTable2.java

Engine.addDocs Engine.getWords()

Test suite: TestSuite3

Gr3: *Gr2* (WordTable(String), addDoc) + *LookUp*

BBT:

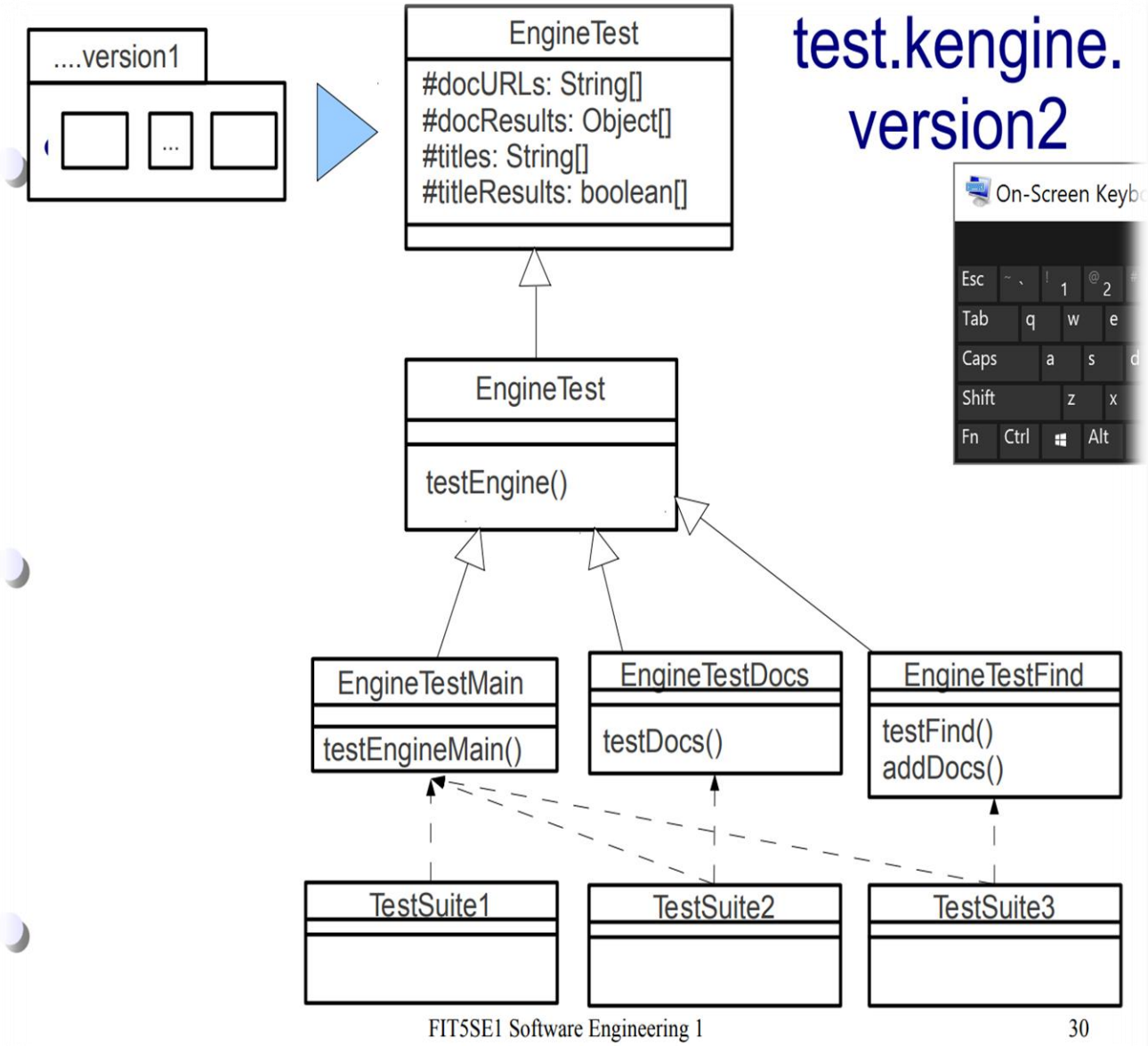
- k: an interesting word
- k: uninteresting
- k: non-word

Test Driver: EngineTestLookUp.java

Engine.addDocs Engine.queryFirst

Test suite: TestSuite4

test.kengine.version2



Test Iteration 4

Query: full query logic

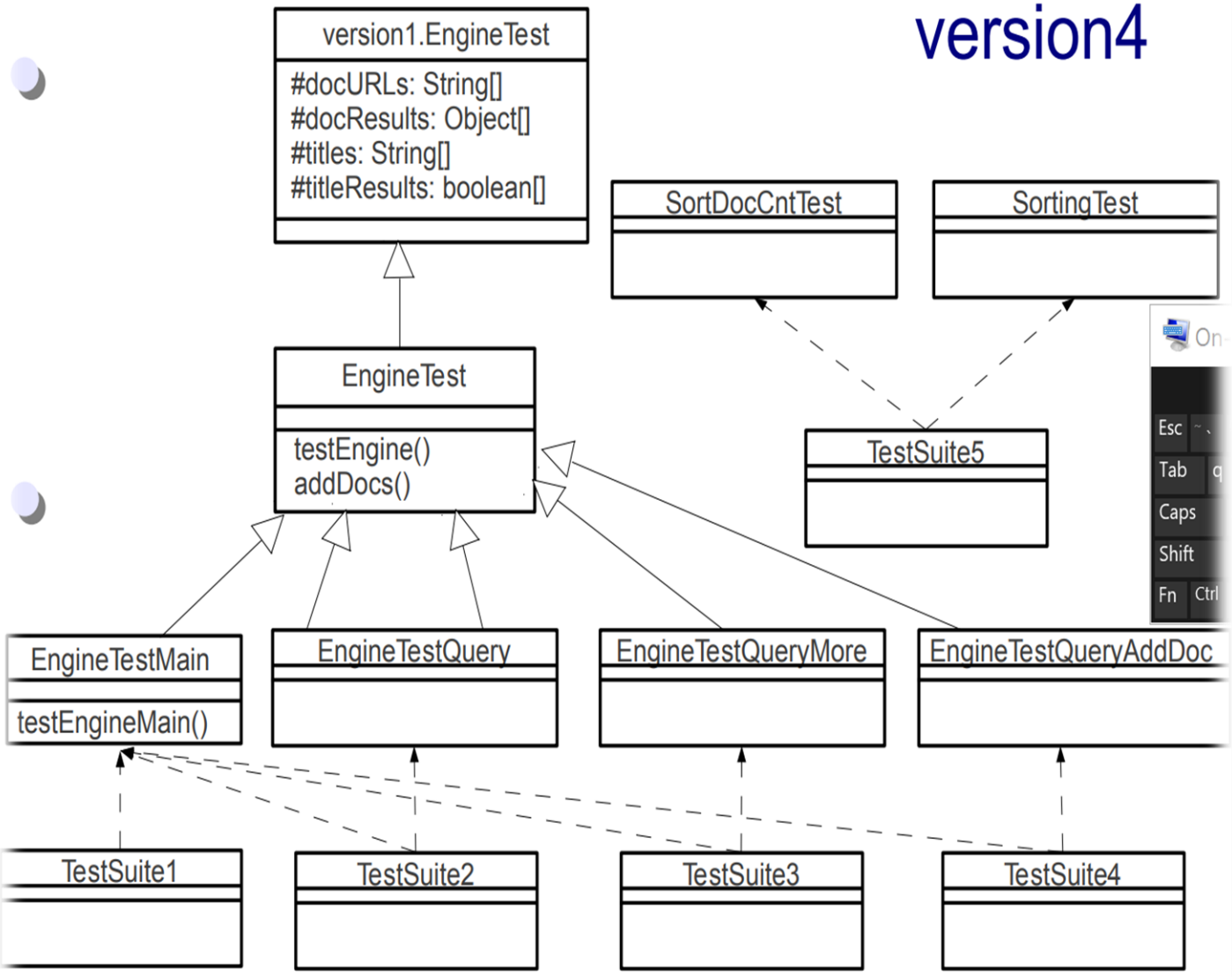
Engine: queryFirst queryMore addDocs

Sorting + DocCnt:

DocCnt —implement→ Comparable (interface)

➔ quickSort(Vector) —sort→ DocCnt objects Vector

test.kengine.version4



Engine Sorting

Engine

constructor Engine()

Test Driver: EngineTestMain

Test suite: TestSuite1

Sorting

Sorting
quicksort(Vector)

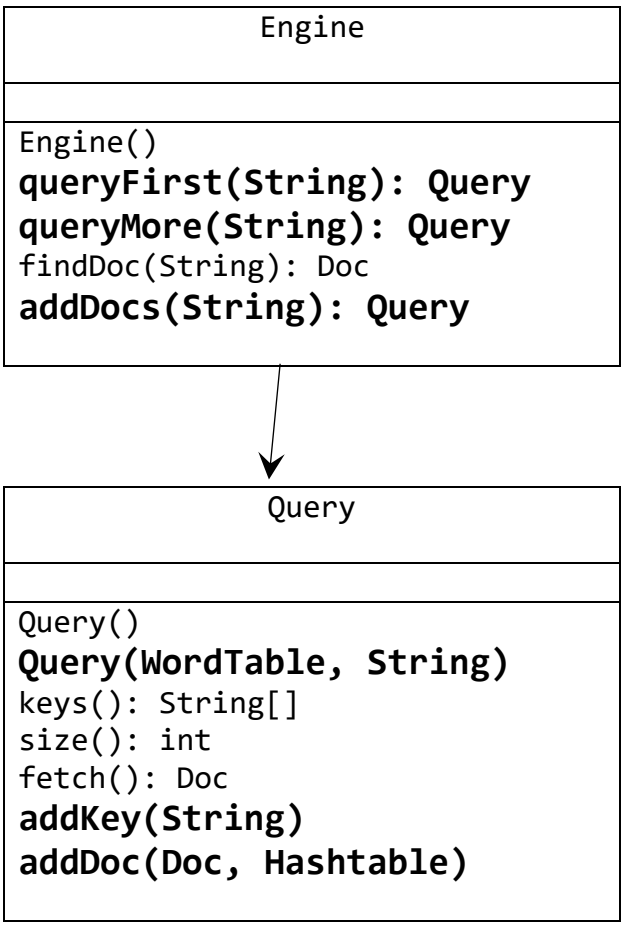
Test method: quicksort(Vector)

Test Driver: SortingTest

Query & Engine Sorting & DocCnt

Query & Engine

use same set of Test Drivers → Engine + Query



keys(): toString() → results

size() & fetch(): all groups → verify test results

Engine.queryFirst → Query(WordTable, String)

Engine.queryMore → Query.addKey()

Engine.addDocs → Query.addDoc()

Gr1: *Query(WordTable, String)*

TCs → Engine.queryFirst() + Query(WordTable, String)

- w: a keyword (table)
- w: invalid word (~~table~~)
- w: non-interesting

Test Driver: EngineTestQuery.java

← Engine.queryFirst

Test suite: TestSuite2

Gr2: *Gr1*(Query(WordTable, String) + *addKey*

addKey TCs:

- w: an existing keyword
- w: new keyword

Engine.queryMore additional TCs:

- w: an invalid word
- w: non-interesting

Test Driver: EngineTestQueryMore.java

← Engine.queryFirst + Engine.queryMore

Test suite: TestSuite3

Gr3: *Gr2*(Query(WordTable, String), addKey) + *addDoc*

addDoc TCs:

- d: ~~query keywords~~
- d: 1 of query keywords
- d: all query keywords

Test Driver: EngineTestQueryAddDoc.java

← Engine.queryFirst queryMore addDocs

Test suite: TestSuite4

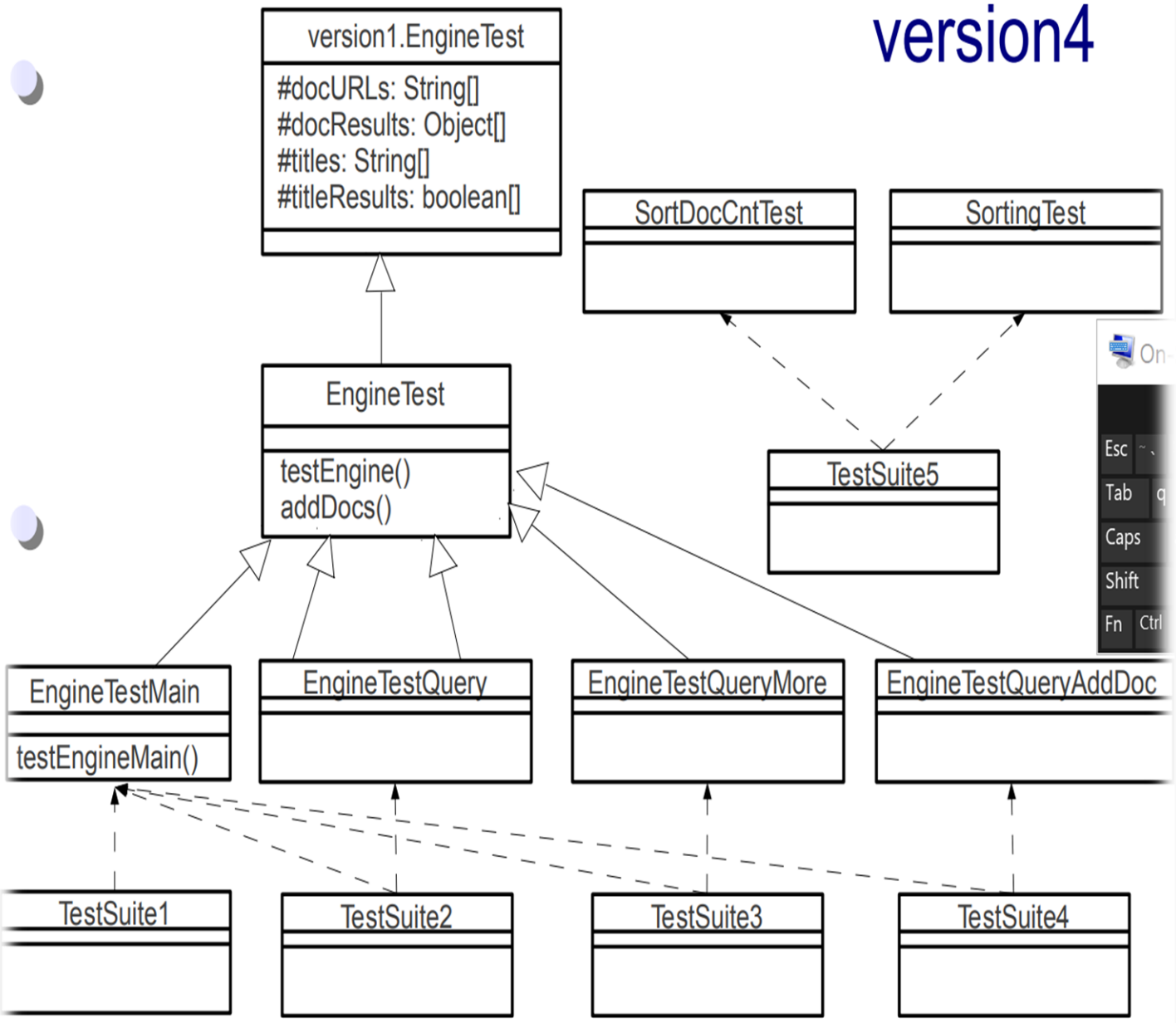
Sorting & DocCnt

TC: a Vector → (some) DocCnt objects

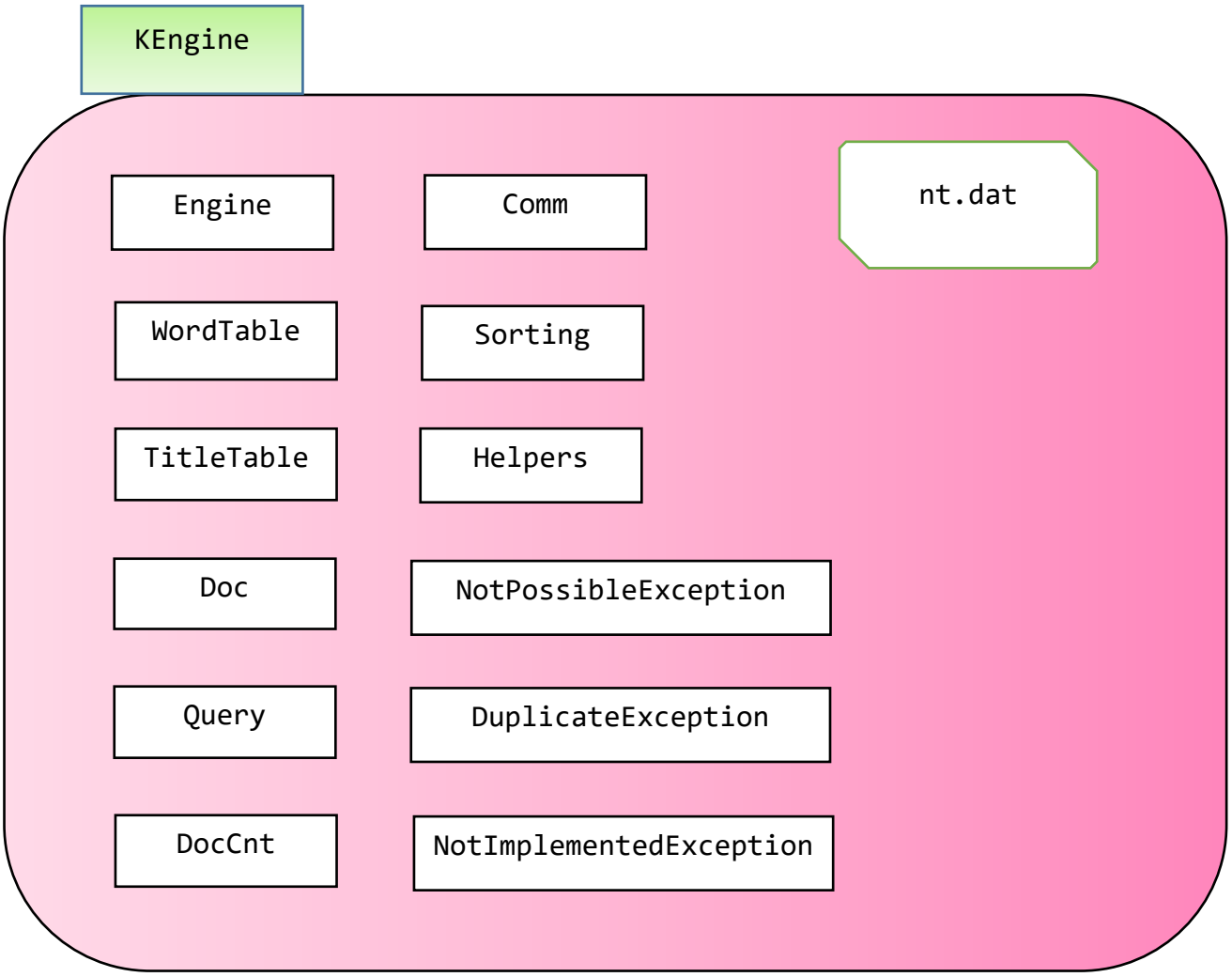
Test Driver: SortDocCntTest.java

Test suite: TestSuite5

test.kengine.version4



Tested Code


KEngine
Engine
Comm
nt.dat
WordTable
Sorting
TitleTable
Helpers
Doc
NotPossibleException
Query
DuplicateException
DocCnt
NotImplementedException