Report: Stage 1 Experimentation and research

In this section, we evaluate the performance of a grammar based test suite by comparing it against a baseline: a manually constrcuted suite. We evaluate the performance of each test suite using the following indicators :

1. Time taken to construct the test suite
2. Code Coverage

Code coverage

Instruction Coverage: this metric measures the % of java bytecode instruction executed.

Branch Coverage: also known as decision coverage, this metric measured the % of branch (typically if/switch statments) exercised by the test suite. Exceptions are not included.

Experimental Setup

The test suites were constructed using a blackbox approach
For the baseline, the test were constructed by simply looking at the api
For the grammar suite, The context free grammars were written up then used to construct the test suites

Test Data

For the purpose of this investigation , libraries of various sizes and complexities were chosen

Library	Number of Lines	Number of Methods	Number of Instructions (java bytecode)
Strman	358	148	2049
TrieSET (princeton)	77	16	416
StringWriter	32	13	119

Grammars

Class: StringWriter
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/io/StringWriter.html

test :- object,testrule+ .
object :- identifier,'=',constructor, ';' .
testrule :- identifier,'.',rules, ';' .
identifier :- 'x' .
rule :- StringWriter | void | StringBuffer | String

constructor :- 'new','StringWriter','(',[int],')' .
StringWriter :- append .
void :- close | flush | write .
StringBuffer :- getBuffer .
String :- toString

append :- 'append', '(', (char | charsequence | charsequence, ',', int, ',', int), ')' .
close :- 'close', '(', ')' .
flush :- 'flush','(', ')' .
write :- 'write', '(', (int | String, [',', int+] | char[], ',', int+) , ')' .
getBuffer :- 'getBuffer','(',')' .
toString :- 'toString','(',')' .

Class:TrieSet
https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/TrieSET.html
test :- object,testrule+ .
object :- identifier,'=',constructor, ';' .
testrule :- identifier,'.',rule, ';' .
identifier :- 'x' .
rules :- void | boolean | Iterator<String> | Iterable<String> | String | int

constructor :- 'new', 'TrieSET', '(', ')' .
void :- add | delete .
boolean :- contains | isEmpty .
Iterator<String> :- iterator .
Iterable<String> :- keysThatMatch | keysWithPrefix .
String :- longestPrefixOf .
int :- size

add :- 'add', '(', String, ')' .
delete :- 'delete', '(', String, ')' .
contains :- 'contains', '(', String, ')' .
isEmpty :- 'isEmpty', '(', ')' .
iterator :- 'iterator', '(', ')' .
keysThatMatch :- 'keysThatMatch', '(', String, ')' .
keysWithprefix :- 'keysWithPrefix', '(', String, ')' .
longestPrefixOf :- 'longestPrefixOf', '(', String, ')' .
size :- 'size', '(', ')'

Class: Strman
http://shekhargulati.github.io/strman-java/
test :-testrule+ .
testrule :- identifier,'.',rule, ';' .
identifier :- 'Strman' .

rule :- String | String[] | List<String> | Map<Character, Long> | boolean | long | int | Optional<String> .

String :-
append
|appendArray

|collapseWhiteSpace
|ensureLeft
|base64Decode
|base64Encode
|binDecode
|binEncode
|decDecode
|decEncode
|ensureRight
|format
|hexDecode
|hexEncode
|insert
|last
|leftPad
|leftTrim
|prepend
|prependArray
|removeLeft
|removeNonWords
|removeRight
|removeSpaces
|repeat
|replace
|reverse
|rightPad
|rightTrim
|safeTruncate
|truncate
|htmlDecode
|htmlEncode
|shuffle
|slice
|slugify
|transliterate
|surround
|toCamelCase
|toStudlyCase
|toDecamelize
|toKebabCase
|toSnakeCase
|decode

|encode
|join
|lowerFirst
|upperFirst
|capitelize
|swapCase
|humanize
|dasherize .

String[] :- between | chars | removeEmptyStrings | split | words | chop | lines | underscored .
List<String> :- zip .
Map<Character, Long> :- charsCount .

boolean :- contains | containsAll | containsAny | endsWith | inequal | isEnclosedBetween | isLowerCase | isString | isUpperCase | unequal | isBlank .

long :- countSubstr
int :- indexOf | lastIndexOf | length .

Optional<String> :- at | first | head | tail | trimsEnd | trimStart .

at :- 'at','(',String,',',int,')'.
first :- 'first','(',String,',',int,')'.
head :- 'head','(',String,')'.
tail :- 'tail','(',String,')'.
trimEnd :- 'trimEnd','(',String,{String},')'.
trimStart :- 'trimStart','(',String,{String},')'.
indexOf :- 'indexOf','('String,',',String,',',int,',',boolean,')'.
lastIndexOf :- 'lastIndexOf','(',String,',',String[,',',int][,',',Boolean],')'.
length : - 'length','(',String,')'.
countSubstr :- 'countSubstr','(',String,',',String[,',',boolean,',',boolean],')' .
contains :- 'contains','(',String,',',String[,',',Boolean],')'.
containsAll :- 'containsAll','(',String,',',String[][,',',Boolean],')'.
containsAny :- 'containsAny','(',String,',',String[][,',',Boolean],')'.
endsWith :- 'endsWith','(',String,',',String[,',',int][,',',Boolean],')'.
inequal :- 'inequal','(',String,',',String,')'.
isEnclosedBetween :- 'isEnclosedBetween','(',String,',',String[,',',String],')'.
isLowerCase :- 'isLowerCase','(',String,')'.
isString :- 'isString','(',String,')'.

```
isUpperCase :- 'isUpperCase','(',String,')'.
unequal :- 'unequal','(',String,',',String,')'.
isBlank :- 'isBlank','(',String,')'.
charsCount :- 'charsCount','(',String,')'.
zip :- 'zip','(',String[],')' .
between :- 'between','(',String,',',String,',',String,')'.
chars :- 'chars','(',String,')'.
removeEmptyStrings :- 'removeEmptyStrings','('String[],')'.
split :- 'split','(',String,',',String,')'.
words :- 'words','(',String,')'.
chop :- 'chop','(', String, ',',int,')' .
lines :- 'lines','(',String,')'.
underscored :- 'underscored','(',String,')'.
append :- 'append','(',String,{',',String},')' .
appendArray :-      'appendArray','(',String,','String[],')' .
collapseWhiteSpace: -     'collapseWhiteSpace','(',String,')' .
ensureLeft :-          'ensureLeft','(',String,',',String[,',',Boolean],')'.
base64Decode :-    'base64Decode','(',String,')'.
base64Encode :-    'base64Encode','(',String,')'.
binDecode:-'binDecode','(',String,')'.
binEncode :-          'binEncode','(',String,')'.
decDecode :-          'decDecode','(',String,')'.
decEncode:-'decEncode','(',String,')'.
ensureRight :-        'ensureRight','(',String,',',String[,',',Boolean],')'.
format :-      'format','(',String,{',',String},')'.
hexDecode :-          'hexDecode','(',String,')'.
hexEncode:-           'hexEncode','(',String,')'.
insert :-        'insert','(',String,',',String,',',int,')'.
last:-  'last','(',String,',',int,')'.
leftPad :-     'leftPad','(',String,',',String,',',int,')'.
leftTrim :-    'leftTrim','(',String,')'.
prepend :-    'prepend','(',String,{',',String},')'.
prependArray :-     'prependArray','(',String,','String[],')'.
removeLeft :-         'removeLeft','(',String,',',String[,',',Boolean],')'.
removeNonwords :-         'removeNonWords','(',String,')'.
removeRight :-      'removeRight','(',String,',',String[,',',Boolean],')'.
removeSpaces :-    'removeSpaces','(',String,')'         .
repeat :-       'repeat','(',String,',',int,')'.
replace :-     'replace','(',String,',',String,',',String,',',Boolean,')'.
reverse :-     'reverse','(',String,')'          .
rightPad :-   'rightPad','(',String,',',String,',',int,')'.
rightTrim :-  'rightTrim','(',String,')'.
```

safeTruncate :- 'safeTruncate','(',String,',',int,',',String,')' .
truncate :- 'truncate','(',String,',',int,',',String,')'.
htmlDecode :- 'htmlDecode','(',String,')'.
htmlEncode :- 'htmlEncode','(',String,')'.
shuffle :- 'shuffle','(',String,')'.
slice :- 'slice','(',String,',',int,',',int,')'.
slugify :- 'slugify','(',String,')'.
transliterate :- 'transliterate','(',String,')'.
surround :- 'surround','(',String,',',String,',',String,')'.
toCamelCase :- 'toCamelCase','(',String,')'.
toStudlyCase :- 'toStudlyCase','(',String,')'.
toDecamlize :- 'toDecamelize','(',String,')'.
toKebabCase :- 'toKebabCase','(',String,')'.
toSnakeCase :- 'toSnakeCase','(',String,')'.
decode :- 'decode','(',String,',',int,',',int,')'.
encode :- 'encode','(',String,',',int,',',int,')'.
join :- 'join','(',String[],',',String,')'.
lowerFirst :- 'lowerFirst','(',String,')'.
upperFirst :- 'upperFirst','(',String,') .
capitelize :- 'capitelize','(',String,')'.
swapCase :- 'swapCase','(', String,')'.
humanize :- 'humanize','(',String,')'.
dahserize :- 'dasherize','(',String,')'.

Results

Suite	Library	Time	Instruction Coverage/%	Branch Coverage/%
Baseline	Strman	10 hours	86%	67%
	TrieSET	1hour 20 min	96%	88%
	StringWriter	4 hours	95%	75%
Grammar	Strman	5 hours	91%	84%
	TrieSET	1 hour	98%	95%
	StringWriter	1 hour 30min	98%	75%

Commentary:

took less time to construct suites using grammar for all libraries; took roughly 15hours and 20 min for basline and 7hours and 30min (almost half the time of the baseline)

Instruction coverage increased in all cases for the grammar suite
Branch coverage increased in all cases for grammar suite, more significant for larger libraries (strman)

Handling void Libraries

Consider the following library which contains a set of instance methods that have a void return type:

PipedWriter()
PipedWriter(PipedReader snk)

void close()
void connect(PipedReader src)
void flush()
void write(char[] cbuf, int off, int len)
void write(int c)

To test such a library, we enumerate different combinations of method calls to simulate method chaining. In order to do this, we append to the grammar of the api a test grammar. This test grammar allows us to chain method calls on a single object of the class.

```
test :- object,rule+ .
object :- identifier,'=',pipedWriter,';' .
rule :- identifier,'.',void, ';' .
identifier :- 'x' .

pipedWriter :- 'new','PipedWriter','(','PipedReader',')' .
void :- close | connect | flush | write .
close :- 'close','(',')' .
connect :- 'connect','(','pipedReader',')' .
```

```
flush :- 'flush','(',')' .
write :- 'write','(',(char[], ',', int, ',', int | int),')' .
```

Test grammar, API grammar