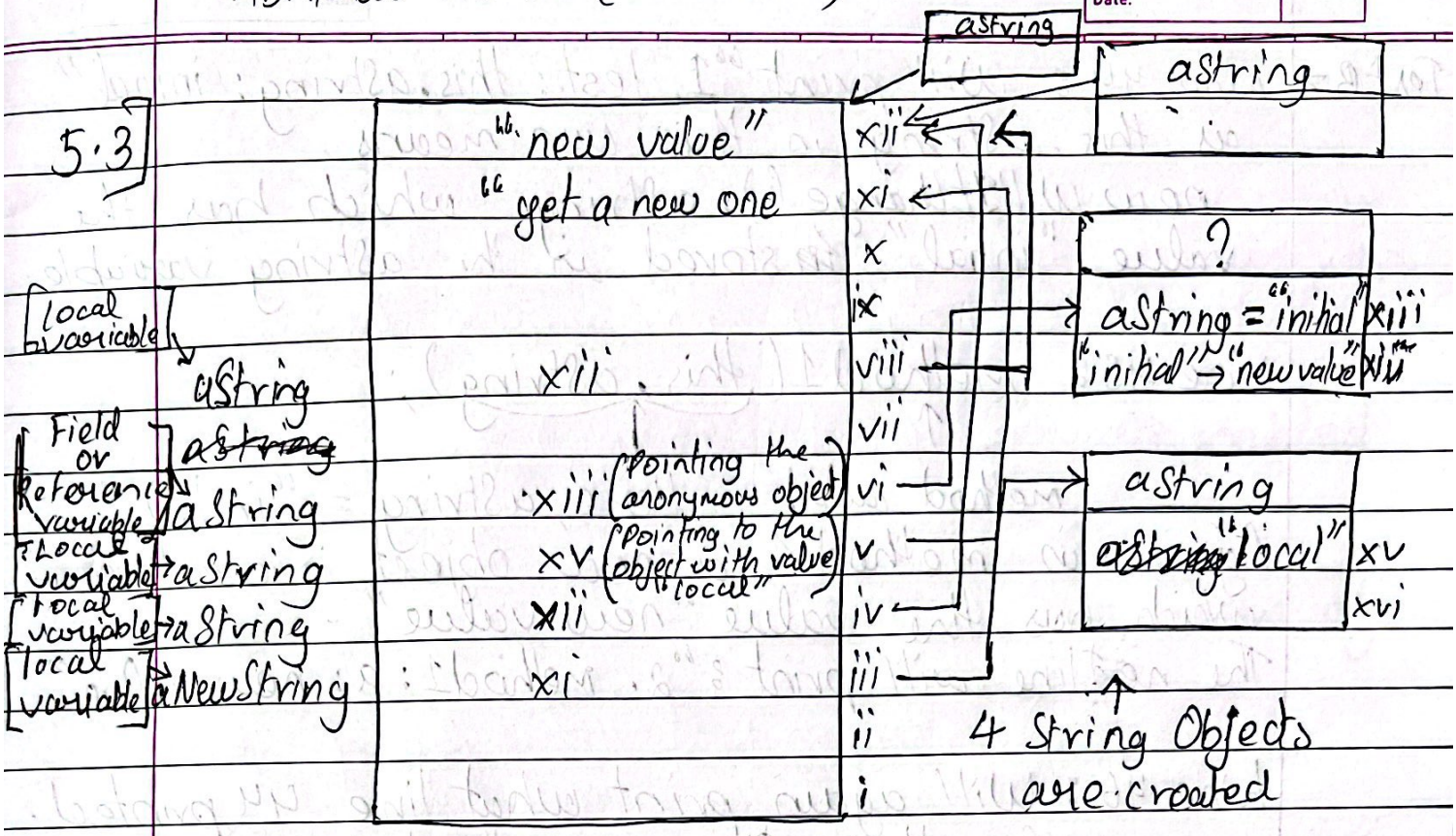
Srivenkatesh Nair (sn6711)
Abhinav Menon (am6176)

5.3



Line 81: new WillItChange().test();
This line will create an anonymous object and call the test method.

Line 36: String aString = new String("local");
will create a new object aString with the value "local"
Test A -
Line 39: Will print "1. test: aString: local"

Line 40: method1(aString);
↑ method called ↑ Here, aString = "local"
In method 1, aString = new String("new value");
will create a new aString object pointing to the value "new value".
The next line will print:
"2. method1: a: new value"
Line 41: Will print what line 39 printed for aString because strings are immutable objects.

Page No.:
Date:
YOUVA

Test B → Line 44: Will print "1. test: this.aString: initial"
as this.aString in this case means
new will change(), aString which has the
value "initial" stored in the aString variable.

Line 45: method1(this.aString);
↑ ↑
method called Here, aString = "initial"
Again in method 1 a new object is created
which has the value "new value".
The next line will print: "2. method1: a: new value"

Line 46: Will again print what line 44 printed.
for this.aString. i.e. "initial"

Test C → Line 44: Will print "1. test: aString: local"
Here aString will pointing to the value of object aStr
Line 50: method 2(aString);
↑ ↑
method called Here, aString = "local"
within the method 2,
this.aString = "new value"
So it changes the value of aString from
~~new value~~ "initial" to "new value".
↑
aString value of the anonymous object
The next line will print: "2: method 2: a: new value"

Line 51: Will again print what line 44 printed.
for aString. i.e. "local"

Test D → Line 54: Will print "1. test: this.aString: "new value"
It will print "new value" because previously
in method 2 we changed its value from
"initial" to "new value"

Line 55: method 2 (this.aString);

method called Here, aString = "new value"

within method 2, it will reinitialize aString to "new value" then print "2. method 2: a: new value".

Line 56: will again print what Line 54 printed for this.aString i.e. "new value"

TestE→ Line 59: Will print "1: test: aString: local".

Here, aString will be pointing to the value of object aString.

Line 60: method 3 (aString);

method called Here, aString = "local"

within method 3, aString is initialized to a NewString which is equal to "get a new one" and prints "2. method 3: a: get a new one".

Line 61: Will print ~~who~~ again what Line 54 printed for aString i.e. "local"

TestF→ Line 64: Will print "1. test: this.aString: new value"

Here, this.aString points to "new value".

Line 65: ~~method 3 (aString)~~

method 3 (this.aString);

method called Here, aString = "new value"

Within method 3, aString is initialized to a NewString which has the ~~val~~ value a string "get a new one". So, aString is equal to that and prints: "2. method 3: a: get a new one"

Line 66: Will again print what Line 64 printed for this.aString i.e. "new value".

Test G → Line 69: Will print "1. test: aString: local"
Here, aString is pointing to the value "local"

Line 70: method4(aString);
↑ method called ↑ Here, aString = "local"

within method 3:
aString = this.aString;
aString is initialized to the value of this.aString which is "new value" and will print:
"2. method 4: a: new value"

Line 71: Will again print what line 64 printed for aString i.e "local"

Test H → Line 74: Will print "1. test: this.aString: new value"
Here, this.aString is pointing to "new value"

Line 75: method4(this.aString);
↑ method called ↑ Here, aString = "new value"

Within method 4:
aString = this.aString;
The value aString is reinitialized to "new value" (which means it's value was already "new value"
Therefore it will print: "2. method 4: a: new value"

Line 76: Will again print what Line 74 printed for this.aString i.e "new value"