CSCI2600 Exam 2

Aaron Taylor

TOTAL POINTS

85 / 100

QUESTION 1

Question 1 40 pts

1.1 a 10 / 10

- 0 **Correct**
- 2 requires: 0<=i<size()
- 2 modifies: nothing
- 2 effects: none
- 2 return: String located at position i of this
- 2 throws: nothing

1.2 b 0 / 6

- 0 Correct
- 2 Correct but lacking correct/brief description
- 3 Correct but wrong/no description
- 3 Partially correct
- 6 **Incorrect**

> The answer is No. All of the instance variables are private and the only data that is shared by the list and client code are references to immutable String objects, which the client cannot change.

1.3 c 9 / 12

- 0 Correct
- 3 Test 1 incorrect
- 3 Test 2 incorrect
- 3 Test 3 incorrect
- 6 Incomplete test cases
- 3 **Didn't cover how to verify the results**
- 3 Incorrect answer for size == capacity
- 12 Incorrect/No answer

> For full credit, each of the three tests needed to specifically describe the test setup, observed results and how to verify the results.

1.4 d 12 / 12

- 0 **Correct**
- 3 The code must check for nulls
- 3 The code must check for sorted array
- 3 The code must check for exceptions

QUESTION 2

2 Question 2 14 / 16

- 0 Correct
- 2 a is false
- 2 b is false
- 2 **c is false**
- 2 d is false
- 2 e is false
- 2 f is false
- 2 g is false
- 2 h is true

QUESTION 3

3 Question 3 10 / 10

- 0 **Correct**
- 1 a is false
- 1 b is false
- 1 c is true
- 1 d is false
- 1 e is true
- 1 f is false
- 1 g is false
- 1 h is true
- 1 i is false
- 1 j is false

QUESTION 4

4 Question 4 10 / 10

- 0 **Correct**
- 5 4 -1: Incorrect/Missing Invariant
- 5 4 -2: Incorrect/Missing Invariant

QUESTION 5

5 Question 5 4 / 8

- 0 Correct
- 4 **first call is A: X m(Z z)**
- 4 second call is C: Y m(W w)
- 8 Wrong

QUESTION 6

6 Question 6 10 / 10

- 0 **Correct**
- 2 a wrong
- 3 b wrong
- 2 c wrong
- 2 d wrong
- 2 ewrong

QUESTION 7

7 Question 7 6 / 6

- 0 **Correct**
- 0 a wrong
- 0 b wrong

NAME Josh Barthelmess

RCS ID barthj2 @RPI.edu

Your RCS ID is the first part of your RPI e-mail address

Exam 2
CSCI 2600 Principles of Software
March 30, 2017

- **DO NOT OPEN THIS EXAM UNTIL TOLD TO DO SO!**
- **READ THROUGH THE ENTIRE EXAM BEFORE STARTING TO WORK.**
- **YOU ARE ALLOWED ONLY A SINGLE "CHEAT" PAGE. NO OTHER MATERIAL IS ALLOWED.**

This exam is worth 100 points.

Make sure you have 10 pages counting this one. If you need more room for an answer than is provided, please use the back of the page and indicate that you have done so. If you re-do a question, please make clear what is your final answer. Be clear and brief in your explanations—rambling and lengthy answers will be penalized. All questions have short answers.

The following is for the use of graders

1. _____/40
2. _____/16
3. _____/10
4. _____/10
5. _____/8
6. _____/10
7. _____/6

TOTAL:_____/100

Name: Josh Barthelmess RCS ID barthj2

Question 1. (40 points total)

Consider the following class:

```
public class FiniteSortedStringList {
    // Rep: items[0..size-1] contains Strings sorted in order
    // given by compareTo(i.e., items[0]<=items[1]....
    // Entries are not null. There may be multiple copies
    // of the same string in the FiniteSortedStringList (FSSL).

    private String[] items;
    private int size;

    /** Construct new FSSL with given capacity. */
    public FiniteSortedStringList(int capacity) {
        this.items = new String[capacity];
        this.size= 0;
    }

    /** Return capacity of this FSSL */
    public int capacity() { return items.length; }

    /** Return current size of this FSSL*/
    public int size() { return size; }

    /** Return item at position i of this FSSL */
    public String get(int i) {
        return items[i];
    }

    /** Return whether s is located in this FSSL *
    public boolean contains(String s) {
        if (size == 0) return false;
        return Arrays.binarySearch(items, 0, size, s) >= 0
    }
    // additional methods to be added or discussed below…
}
```

Name: Josh Barthelmess RCS ID barthj2

a) (10 points) The get() method specification is incomplete. Write an appropriate specification in PoS style (@requires, @modifies etc.) If some part of the specification would be empty or "none", say so explicitly. Do not alter the method.

@requires: $0 \leq i \leq size-1$

@modifies: None

@returns: reference to string at ith position in the Array

@throws: None

@effects: None

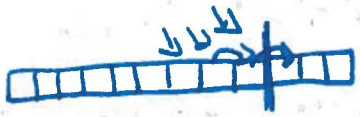
b) (7 points) Are there any potential problems from possible representation exposure with the FiniteSortedStringList class? Why or why not? (Be brief!)

The get() method exposes the representation. If the client were to save the returned and then set that to equal something else it would change the representation outside of the representation.

Name: Josh Barthelmess RCS ID barthj2

c) (12 points) Consider the following method to be added to the FiniteSortedStringList class. compareTo() returns a negative integer, zero, or a positive integer if *this* object is less than, equal to, or greater than the specified object.

```
public void add(String s) {
    int k = size;
    while (k > 0 && items[k-1].compareTo(s) > 0) {
        items[k] = items[k-1];
        k = k - 1;
    }
    items[k] = s;
    size = size + 1;
}
```



size=9

As part of the implementation of the add method, you should also create tests for it. Below, describe three good black box tests for this method. Each test should cover a different revealing subdomain. You do not need to give JUnit or other code, but you should describe a specific test setup and the expected results for each. Be brief and to the point

First, test the add function when FSSL is empty. It should work, size should equal 1 and the added string should be the first item in the list.

Next, test adding an item bigger than any other item in the FSSL. Again, it should work, and the string should be the last item in the list.

Finally, test adding a string that is the same as a string already in the list. If the string was the k^{th} item it should now also be the $k+1$ element.

Describe what happens in the add method if size is equal to capacity.

As described above, if size is equal to capacity, trying to add another item will produce an error, IndexOutOfBounds Exception.

Name: Josh Barthelmess RCS ID barthj2

d) (12 points) Alyssa P. Hacker, a diligent CSCI-2600 student, points out that the FiniteSortedStringList code does not have a checkRep() method. Write a checkRep() method for this class. The method should check the representation as described in the comments at the start of the class definition.

```
private void checkRep()
{
if(size > capacity)
    throw new RuntimeError("size is bigger than
                            capacity");

for(int i=0; i<size; i++){
{
    if(item[i] == null)
        throw new NullPointerException("Item is Null");
    if(item[i] > item[i+1])
        throw new RuntimeError("Items not in Sorted Order");

}
}
```

Name: Josh Barthelmess RCS ID barthj2

Question 2. (16 pts, 2pts each) TRUE/FALSE

a) (TRUE/FALSE) It is always possible to cover all def-use pairs in a function.
b) (TRUE/FALSE) When considering a def-use path, a variable is used when it appears on the left hand side of an assignment statement.
c) (TRUE/FALSE) A program containing a method M passes an exhaustive set of tests. You replace M by a method with a stronger spec. The program will still pass all tests.
d) (TRUE/FALSE) White-box tests are specification tests.
e) (TRUE/FALSE) A test suite that has 100% statement coverage covers all def-use paths.
f) (TRUE/FALSE) A well-written test suite for a piece of software can guarantee that there are no bugs in the software.
g) (TRUE/FALSE) In unit testing, it is best to test all of a class's or module's functionality in one test method
h) (TRUE/FALSE) In Java, it is possible to use try..catch to recover from exceptions and continue program execution.

Question 3. (10 pts)

Assume the following code compiles:

```
A a = new A();
B b = new C(a);
D d = b;
```

C is a subtype of B

Mark all of the following that must be true:

a) A is a supertype of B.
b) A is a subtype of B.
c) B is a supertype of C.
d) ~~B is a subtype of C.~~
e) B is a subtype of D.
f) B is a supertype of D.
g) C is a supertype of D.
h) C is a subtype of D.
i) A is a supertype of D.
j) A is a subtype of D.

Name: Josh Barthelmess RCS ID barthj2

Question 4 (10 points)

The following partial class definition is for an implementation of a polynomial with integer coefficients.

```
class Poly {
    private int [] coeffs;  // the integer coefficients
    private int degree;   // the degree of the polynomial

    // the rest of the class definition follows….
}
```

Write a suitable rep invariant for this class.

degree = coeffs.length - 1

coeffs[degree] != 0

Next, write an abstraction function.

coeffs[i] -> $c_i x^i$

coeffs [c_0 | c_1 | c_2 | c_3] -> $c_0 + c_1 x + c_2 x^2 + c_3 x^3$

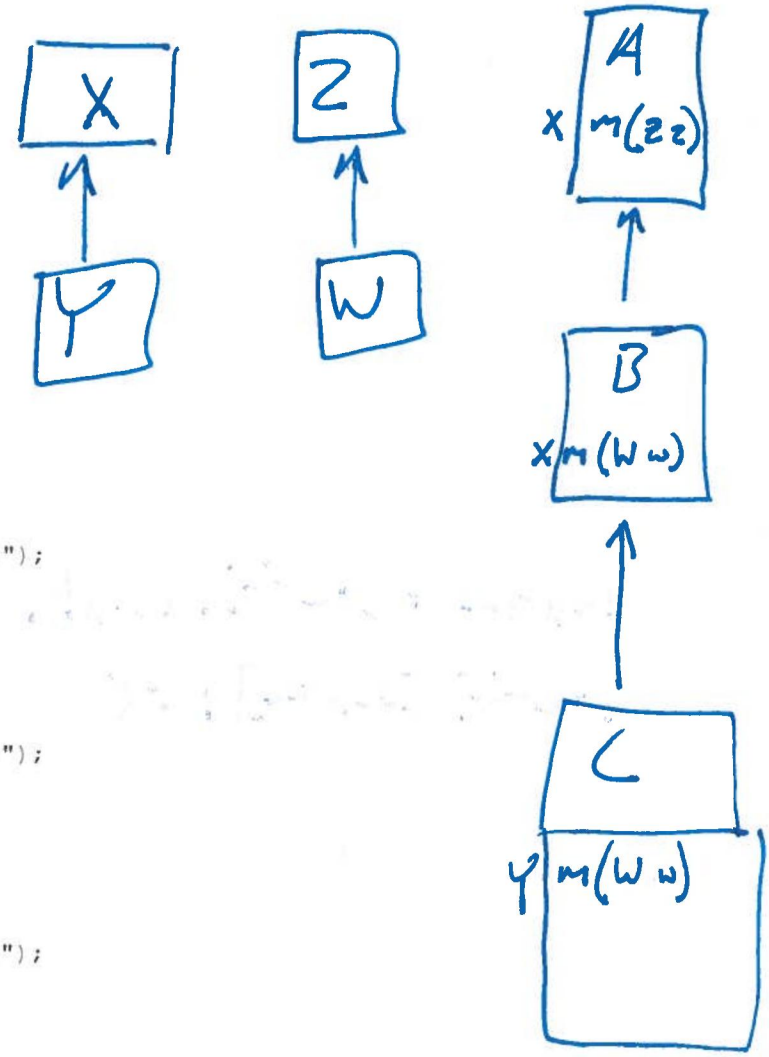
~~[illegible]~~ Each number in coeffs corresponds to the x^i term in the polynomial.

Name: Josh Barthelmess RCS ID barthj2

Question 5 (8 points)

```
class X {
    X() { }
}
class Y extends X {
    Y() { }
}
class Z {
    Z() { }
}
class W extends Z {
    W() { }
}
class A {
    A() {}
    X m(Z z) {
        System.out.println("A: X m(Z z)");
        return new X();
    }
}
class B extends A {
    B() { }
    X m(W w) {
        System.out.println("B: X m(W w)");
        return new X();
    }
}
class C extends B {
    C() { }.
    Y m(W w) {
        System.out.println("C: Y m(W w)");
        return new Y();
    }
}
public class SubclassDemo3 {
    public static void main(String[] args) {
        A a = new C();
        W w = new W();
        // Which m is called?
        X x = a.m(w);

        C c = new C();
        W w2 = new W();
        // Which m is called?
        Y x3 = c.m(w2);
    }
}
```



When the code above is executed, which m's are called?

In both instances, the m defined in the subclass C is called.

Name: Josh Barthelmess RCS ID barthj2

Question 6 (10 points)

For each of the following design patterns, give an example that shows one benefit of using that particular pattern. Explain why using this pattern is worthwhile.

a) Adapter

- Useful when you want to use the same functionality of a class, but different interface.
i.e. evaluating sin(x) in degrees instead of radians.

b) Factory

- Useful for handling object creation with needing to write lots of different constructors
i.e. Many instances needed [redacted]

c) Proxy

- Useful when you want to restrict Access to a class
i.e. Only allow usage if you know a password

d) Singleton

- Useful When All instances of the class are the same
i.e. if the class contained solely a library of functions and some constants

e) Visitor

- Useful for connecting various pieces of composite classes

Used when the function "accepts" the visitor and the visitor class does the work for the function

Name: Josh Barthelmess RCS ID barthj2

Question 7 (6 points)

The implementation of equals in class Object returns true if two objects are exactly the same, i.e., a.equals(b) returns the result of the comparison a==b.

a) Show that this definition defines proper equivalence relationship. That is, show that it is reflexive, symmetric, and transitive.

a.equals(a) returns a==a always true for any object

a.equals(b) -> b.equals(a) a==b->b==a

~~the~~ the .equals() function is ~~a~~ symmetric because == is symmetric

a.equals(b) ∧ b.equals(c)

↘ a.equals(c)

again .equals() is transitive because == is transitive.

a==b ∧ b==c -> a==c

b) The implementation of hashCode in Object returns the memory address of the object. Explain why this implementation of hashCode provides an effective hash function.

This is an effective hash function because two ~~same~~ different objects cannot have the same memory address, so each hashcode ~~is~~ will be unique to the object it refers to.