Mt. San Antonio College – Computer Science
CSCI 145: Introduction to Java Programming
Sherdil Niyaz

Practice Final Exam **Solutions**, Fall 2023

This test has **7** questions worth a total of **??** points. The exam is closed book, except that you are allowed to use **six** two-sided handwritten cheat sheets. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided.

Write the statement out below in the blank provided and sign. You may do this before the exam begins. Any plagiarism, no matter how minor, will result in an F.

"I have neither given nor received any assistance in the taking of this exam."

_____

_____

Signature: _____

Name: _____
SID ("A-Number"): _____
Name of person to left:_____
Name of person to right: _____

Tips:

- **DO NOT STOP WORKING UNTIL YOU REACH THE STOP SIGN.**
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. **Do not get overly captivated by interesting problems or complex corner cases you're not sure about.**
- ***DO NOT WRITE ON THE BACKS OF PAGES.*** **Any work you put there WILL NOT be graded.**
- There are several "reserved" pages at the end of this packet that are intended as extra scratch paper. **Make clear what problem** each scratch page is for (if you chose to use them).
- Write the last four digits of your SID on each page in case pages get shuffled during scanning.

Problem	1	2	3	4	5	6	7
Points	??	??	??	??	??	??	??

Optional. Mark along the line to show your feelings on the spectrum between :(and ☺.

Before exam: [:(_____ ☺].
After exam: [:(_____ ☺].

1.Potpourri [?? pts]

For the following questions, please respond with **True** or **False.** You **must** write out the entire word you pick in each case. If you try to make any of your answers below look ambiguous, you will lose all points for that problem.

There is no justification required for each problem, and these questions will be graded **all or nothing.**

Question 1A

In Java, classes are allowed to inherit from multiple different parent classes.

False. Java explicitly prohibits multiple inheritance, as discussed in lecture and the slides.

Question 1B

Classes in Java are allowed to have several methods with the exact same signature, as long as they are differentiated by their return type.

False. The "signature" of a method in Java (used to determine overriding) **does not** include the return type. You can't have multiple methods with the same signature, so this scenario leads to a compile error.

Question 1C

A Linked List is an example of a "dynamically-sized" data structure.

True. A Linked List (unlike an Array) can grow as more elements are added to it. This is done by adding more Node objects to the end of the list.

Question 1D

In order for a child class to inherit a variable or method from its parent class, that variable/method needs to be declared as `public` in the parent.

False. The variable/method can also be declared `protected` (which is the same as `private`, but with a special "carve out" for inheritance).

Question 1E

A `long` is able to hold a larger range of values than an `int`.

True. This was shown in the slides from lecture (specifically in the table where we compared all the primitive types).

General Warning: The topic distribution of the first 7 questions on the Final Exam will follow the general distribution of this practice test. However, that still means that the Q1/Q2 (Potpourri) questions on the real exam can test **any topic** we learned this semester: including those not seen on Q1/Q2 of this practice test. Study up! :)

2. Potpourri Again (It Smells Fragrant In Here) [?? pts]

Each of the following questions can be answered with **at most** a single sentence. If you use more than that to describe your answer, you will receive zero points for that question.

These questions will be graded **all or nothing.**

Question 2A

In lecture, we learned about static type and dynamic type. Which of these matters at compile time? Which of these matters at runtime?

Static type matters at compile time: `javac` checks to make sure that every method/variable accessed on an object exists according to the static type (if not, we get a compilation error). We then run the program (assuming it compiles), at which point dynamic type becomes important (specifically in deciding which version of an overridden method is called).

Question 2B

In lecture, we discussed the two "halves" of a recursive function. What are these two halves?

These are the "base case" (no more recursion needed) and the "recursive case" (do "just a little bit more" work, then call yourself on a smaller input).

Question 2C

You are building a program to store student names. You don't know how many names will be input into the program (it reads input from the user in an infinite loop), and that number can grow during the life of the program. Which of the following would be a reasonable choice of data structure to store these names?

```
Array
ArrayList
Linked List
```

You should pick either an ArrayList or a Linked List. This is because these are both **dynamically-sized** Data Structures (aka they can grow over time instead of being fixed-size, like an array). That will be required by our program as it reads more and more names.

Question 2D

Applying a cast to a reference type variable changes either the static type of that variable or the dynamic type. Which is it?

It changes the static type (which, remember, only matters at compile-time). Note that this change is also only effective on the **specific line** where we applied the cast: references to that variable after the cast will have the original static type (unless, of course, we cast again).

3.WWJD (What Would Java Do?) [?? pts]

Assume that we have these two files `Rizzler.java` and `FunArrayProgram.java` next to each other in the same directory. What does the main method in `FunArrayProgram` print when we run it?

```
public class FunArrayProgram {
    public static void fanumTax(Rizzler[] theBoys)
    {
        theBoys[2] = theBoys[4];
        theBoys[0] = new Rizzler();
    }

    public static void skibbidy(Rizzler[] theBoys)
    {
        theBoys[2].rizzLevel = 420;
    }

    public static void cringe(Rizzler[] theBoys)
    {
        theBoys = new Rizzler[7];

        for (int i = 0; i < theBoys.length; i++)
        {
            theBoys[i] = new Rizzler();
            theBoys[i].rizzLevel = i;
        }
    }

    public static void main(String[] args)
    {
        Rizzler[] theBoys = new Rizzler[5];
        for (int i = 0; i < theBoys.length; i++)
        {
            theBoys[i] = new Rizzler();
            theBoys[i].rizzLevel = i * 10;
        }

        fanumTax(theBoys);
        cringe(theBoys);
        skibbidy(theBoys);

        for (int i = 0; i < theBoys.length; i++)
        {
            System.out.println(theBoys[i].rizzLevel);
        }
    }
}
```

```
public class Rizzler {

    public int rizzLevel;

}
```

Write your Answer Below.

HEADS UP: Make it VERY clear what your final answer (output) is. Ideally that means boxing or circling it. If I have to guess or you try to play games, I will just zero out this entire problem.

HEADS UP 2: Make it VERY clear whether certain characters are printed on the same line or on the next line.

HINT: You really should draw a box-and-pointer for this, instead of attempting to do it all in your head. Use the scratch paper provided at the end of this exam if you need it.

The program prints:

```
0
10
420
30
420
```

You can step through the box-and-pointer yourself by clicking **here**.

4.WWJD2 [?? pts]

Assume that we have these two files `Blorple.java` and `InstanceVariableProgram.java` next to each other in the same directory. What does the main method in `InstanceVariableProgram` print when we run it?

```
// I hope you remember how instance variables and instance
// methods work! :)
public class InstanceVariableProgram {

    public static void main(String[] args)
    {
        Blorple first = new Blorple(100, 200, 300);
        Blorple second = new Blorple(10, 20, 30);

        first.meow();
        second.tricky();

        first.woof();
        second.woof();

        System.out.println(first.x);
        System.out.println(first.y);
        System.out.println(first.z);

        System.out.println(second.x);
        System.out.println(second.y);
        System.out.println(second.z);
    }

}
```

```
public class Blorple {
    public int x;
    public int y;
    public int z;

    public Blorple(int givenX, int givenY, int givenZ)
    {
        this.x = givenX;
        this.y = givenY;
        this.z = givenZ;
    }

    public void meow()
    {
        x = 45;
        int x = 20;
        this.x = 30;
        x = 10;
    }

    public void tricky()
    {
        z = x * y;
        int x = 45;
        int y = 20;
        this.x = z + y;
    }

    public void woof()
    {
        if (this.x < 20)
        {
            System.out.println("Case 1");
        } else if (x < 40) {
            System.out.println("Case 2");
        } else {
            System.out.println("Case 3");
        }
    }
}
```

Write your Answer Below.

HEADS UP: Make it VERY clear what your final answer (output) is. Ideally that means boxing or circling it. If I have to guess or you try to play games, I will just zero out this entire problem.

HEADS UP 2: Make it VERY clear whether certain characters are printed on the same line or on the next line.

HINT: You really should draw a box-and-pointer for this, instead of attempting to do it all in your head. Use the scratch paper provided at the end of this exam if you need it.

The program prints:

```
Case 2
Case 3
30
200
300
220
20
200
```

Step through the box-and-pointer yourself by clicking **here.**

5. Dynamic Method Selection [?? pts]

Suppose we have the following classes below (all in the same directory on your computer). What will get output at each commented line of `TestAnimals.java` when we compile and run that program?

Hint: It might be worth drawing a box and pointer before each method call. What are the static and dynamic types of each variable? What are the instance variables of each object in the program?

```
public class Animal {
    protected String name;
    protected String noise;
    protected int age;

    public Animal(String name, int age) {
        this.name = name;
        this.age = age;
        this.noise = "Huh?";
    }

    public String makeNoise() {
        if (age < 5) {
            return noise.toUpperCase();
        } else {
            return noise;
        }
    }

    public void greet() {
        System.out.println("Animal " + name + " says: " + makeNoise());
    }
}


public class Cat extends Animal {
    public Cat(String name, int age) {
        super(name, age);
        this.noise = "Meow!";
    }

    public void greet() {
        System.out.println("Cat " + name + " says: " + makeNoise());
    }
}
```

```
public class Dog extends Animal {
    public Dog(String name, int age) {
        super(name, age);
        noise = "Woof!";
    }

    public void greet() {
        System.out.println("Dog " + name + " says: " + makeNoise());
    }

    public void playFetch() {
        System.out.println("Fetch, " + name + "!");
    }
}


public class TestAnimals {
    public static void main(String[] args) {
        Animal a = new Animal("Pluto", 10);
        Cat c = new Cat("Garfield", 6);
        Dog d = new Dog("Fido", 4);

        a.greet();          // (A) _____
        c.greet();          // (B) _____
        d.greet();          // (C) _____

        a = c;
        ((Cat) a).greet();  // (D) _____
        a.greet();          // (E) _____
    }
}
```

Answer Here:

View the solution slides by clicking **here**.

Problem Credit: CS61B Spring 2018, UC Berkeley

6. Polymorphic Party! [?? pts]

Assume that you have the following four Java files next to each other in the same directory of your computer.

Comment out (or otherwise strike out) any lines in `D.main()` that cause compile-time errors, runtime errors, or cascading errors (failures that occur because of an error that happened earlier in the program). For each line you comment out, explain **why** you had to comment that line out (you can do this next to each such line).

What does `D.main()` output after removing these lines?

```
public class A {
    public int x;

    public void m1() {System.out.println("Am1-> " + x);}
    public void m2() {System.out.println("Am2-> " + this.x);}
    public void update() {x = 99;}
}


public class B extends A {
    public void m2() {System.out.println("Bm2-> " + x);}
    public void m2(int y) {System.out.println("Bm2y-> " + y);}
    public void m3() {System.out.println("Bm3-> " + "called");}
}

public class C extends B {

    public void m2() {System.out.println("Cm2-> " + this.x);}
}
```

```
public class D {
    public static void main (String[] args) {
        // B a0 = new A(); // COMPILE FAIL: Static types violate "right to left" rule.
        // a0.m1(); // CASCADE: a0 is invalid.
        // a0.m2(16); // CASCADE: a0 is invalid.
        A b0 = new B();
        System.out.println(b0.x); // PRINTS 0 (default value for int).
        b0.m1(); // PRINTS Am1-> 0
        b0.m2(); // PRINTS Bm2-> 0
        // b0.m2(61); // COMPILE FAIL: Static type (A) has no m2 method that takes int.
        B b1 = new B();
        b1.m2(61); // PRINTS Bm2y-> 61
        b1.m3(); // PRINTS Bm3-> called
        A c0 = new C();
        c0.m2(); // PRINTS Cm2-> 0
        // C c1 = (A) new C(); // COMPILE FAIL: Static types violate "right to left" rule.
        A a1 = (A) c0;
        C c2 = (C) a1;
        c2.m3(); // PRINTS Bm3-> called
        ((C) c0).m3(); // PRINTS Bm3-> called
        // (C) c0.m3(); // COMPILE FAIL: Attempts to cast return value, which is nothing.
        b0.update();
        b0.m1(); // PRINTS Am1-> 99
    }
}
```

Write Your Program Output Below (After Removing Bad Lines):

After being fixed, the program prints:

```
0
Am1-> 0
Bm2-> 0
Bm2y-> 61
Bm3-> called
Cm2-> 0
Bm3-> called
Bm3-> called
Am1-> 99
```

Problem Credit: CS61B Spring 2018, UC Berkeley

7. Writing Recursive Code [?? pts]

Question 7A

Complete the outline of the `searchLinkedList` method on the following page. This method should take as parameters a node that is the **front** of a linked list, as well as a `target` integer. It should return true if `target` is contained in that linked list, and false otherwise.

We're using the same Node class as in lecture. In case you've forgotten it:

```
public class Node {
   public int info;

   public Node next;
}
```

```
public class LinkedListProgram {
    // NOTE: Other (omitted) methods here.

    // TODO: Fill in this method.
    public static boolean searchLinkedList(Node curNode, int target)
    {
        if (curNode == null)
        {
            return false;
        }

        if (curNode.info == target)
        {
            return true;
        }

        return searchLinkedList(curNode.next, target);
    }

    public static void main(String[] args)
    {
        // NOTE: Returns the first Node in a Linked List that looks
        // like this:
        // 1 -> 2 -> 3 -> null
        Node frontOfFirstList = getFirstList();

        // NOTE: Returns the first Node in a Linked List that looks
        // like this:
        // 4 -> 5 -> 6 -> null
        Node frontOfSecondList = getSecondList();

        // This should print true.
        System.out.println(searchLinkedList(frontOfFirstList, 2));

        // This should print false.
        System.out.println(searchLinkedList(frontOfSecondList, 2));

    }
}
```

Question 7B

Fill in the following blanks in the `printStarsRecursive` method so that the method prints the passed number of stars on single line of the terminal. It should then print every *lower* integer count of stars on subsequent lines. For example, a call to this method with an argument of five should print five stars on the first line of the terminal, four on the next line, three on the line after that, etc.

```
public class StarProgram {

    public static void printStarsRecursive(int numStars)
    {
        if (numStars == 0)
        {
            return;
        }

        String curString = "";
        for (int i = 0; i < numStars; i++)
        {
            curString += "*";
        }
        System.out.println(curString);

        printStarsRecursive(numStars - 1);
    }

    public static void main(String[] args)
    {
        printStarsRecursive(5);

        // NOTE: This should end up printing:
        // *****
        // ****
        // ***
        // **
        // *
    }
}
```

Scratch Paper 1

Question this page corresponds to: _____
*(Feel free to detach this page if you want, just make sure to **re-staple** it at the end).*

Scratch Paper 2

Question this page corresponds to: _____
*(Feel free to detach this page if you want, just make sure to **re-staple** it at the end).*

Scratch Paper 3

Question this page corresponds to: _____
*(Feel free to detach this page if you want, just make sure to **re-staple** it at the end).*

You have reached the end.

There is no more.

Go outside after this and take a study break. You deserve it :)