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

Practice Midterm **Solutions**, Fall 2023

This test has **9** questions worth a total of **??** points. The exam is closed book, except that you are allowed to use three 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: *Spud*

Name: Spud Niyaz
SID ("A-Number"): 0
Name of person to left: Tuan Vo
Name of person to right: Jimmy Tamayo

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.**
- **You can use the back of each page for scratch paper.** If you need more, **staple those sheets** to your exam.
- Each page of scratch paper you use (either on the back of pages or attached to this exam) should contain work for a **single** problem. **Make clear what problem** each scratch page is for.
- 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	8	9
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

Every Java program that compiles correctly produces the intended output of the programmer.

False

Explanation (you wouldn't need to write this down):
You can still have logical errors in your program.

Question 1B

Once a variable has been created, it is possible to change the type of that variable in Java.

False

Explanation:
Java is a statically-typed language: the types of all variables are pre-determined and can't be changed.

Question 1C

The following assignment in Java is valid:

```
double foo = 4.5;
float bar = foo;
```

False

Explanation:
This is an example of a "narrowing conversion", since double variables have a bigger range (they can "hold more information") compared to floats. This is not possible without a cast in Java.

Question 1D

There is a fixed set of primitive types in Java: **every** other variable type outside of that fixed set is an object/reference type.

True

Explanation:
This was stated nearly-verbatim in lecture along with the "set" of primitive types.

Question 1E

Objects that have no reference pointing to them in Java are deleted automatically over the lifetime of the program.

True

Explanation:
This was mentioned in lecture as well: the formal name of this process is "Garbage Collection", and it's a big difference between Java and C++.

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

What are the values of the x and y variables (respectively) in the code below?

```
Scanner x;
int y;
```

Null (for x) and 0 (for y).

Explanation (you wouldn't need to write this down):
x is a reference/object type variable (see Question 1D on this exam). When initialized, those variables have a default value of null.

y *is* on the short list of primitive types (refer back to the lecture slides). So it has a default value of zero.

Question 2B

What is the value of the `result` variable here?

```
int foo = 10;
double bar = 2.5;
int result = foo / (int) bar;
```

The value of `result` is 5.

Explanation:
The key here is that the *precedence* of the casting operator is higher than everything *except* parenthesis. So the first thing that happens here is that the double bar is casted to an int *before* the division happens.

That casted value is 2 (remember that casting truncates). We then divide an int 10 by and int 2, and get and int 5. Tricky!

Question 2C

What is the value of the `result` variable here?

```
int foo = 10;
double bar = 2.5;
int result = (int) (foo / bar);
```

The value of `result` is 4.

Explanation:
The *precedence* of the casting operator is higher than everything *except* parenthesis. So here (unlike Q2B) we actually perform the division first. We divide 10 (and int) by 2.5 (a double).

During this division, *promotion* occurs, and the operand 10 gets promoted to double as well. So the result of that division is 4.0 (the "correct answer").

Casting that double 4.0 to an int changes the type, but it doesn't actually change the value.

Question 2D

Why does this program not compile?

```
int foo = 10;
double bar = 2.5;
int result = (int) foo / bar;
```

This is actually a narrowing conversion.

Explanation:
Again: the *precedence* of the casting operator is higher than everything *except* parenthesis.

So the first thing we do here is cast foo to an int: but foo is already an int. The cast doesn't actually do anything.

Then we divide 10 (an int) by 2.5 (a double). During this division, *promotion* occurs, and the operand 10 gets promoted to double as well. So the result of that division is 4.0 as a double.

However, assigning 4.0 (a double) to an int is a narrowing conversion (and a no-no in Java).

3. Now We Got Problems (And I Don't Think We Can Solve 'Em) [?? pts]

The following program has **four issues** that are so severe that they prevent the file `FixIt.java` from compiling. Describe each of these four problems in 1-2 sentences **per-problem**. You **do not need to provide a solution:** just pointing out the problem is enough.

Note: Do not describe problems related to the logic or functionality of the program: this program is completely useless anyway. You should describe issues that break the fundamental rules of Java that we discussed in the lectures.

Also: Don't "double count" errors as well: if a variable is created/modified incorrectly, don't just point out every use of that variable afterwards. The issues are all distinct and not related to each other.

```
public class FixIt {
    public char first;
    public char second;

    public static void fuzzy() {
        this.first = 'a';

        System.out.println("Mmm, ice cream so good");

        return 6;
    }

    public static void main(String[] args) {
        long foo = 56;
        int bar = foo;
        System.out.println("Gang Gang");

        count = 0;
        for (int i = 0; i < 6; i++) {
            System.out.println("I hate midterm exams :(");
            count++;
        }

        if (count == 5) {
            System.out.println("Meow!");
        } else if (count == 6) {
            System.out.println("Woof!");
        } else {
            System.out.println("Tweet!");
        }
    }
}
```

Problem 1

The method `fuzzy()` is `static` (which means it is "free-floating" and not attached to any particular object). However, we're using the `this` keyword in that method, which makes no sense.

Problem 2

The method `fuzzy()` has a `void` return type, but we're returning an int in that method.

Problem 3

The second line of `main()` is committing a "narrowing conversion": the line assigns a long type to an int variable. However, long ca hold "more information" (it has a larger range) than int.

Problem 4

The fourth line of `main()` is actually reassigning a variable named `count` to a new value of 0 (not creating a new variable). The problem is that a variable named `count` *doesn't exist* to be re-assigned, so this line is a compiler error.

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

Assume that we have these two files `Point.java` and `Mystery2.java` next to each other in the same directory. What does the main method in `Mystery2` print when we run it?

```
public class Point {
    public int x;
    public int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public String toString() {
        return "(" + this.x + ", " + this.y + ")";
    }
}

public class Mystery2 {
    public static void main(String[] args) {
        Point p = new Point(11, 22);
        System.out.println(p);

        int n = 5;
        mystery(p, n);
        System.out.println(p);

        p.x = p.y;
        mystery(p, n);
        System.out.println(p);

        Point p2 = new Point(100, 200);
        p = p2;
        mystery(p2, n);
        System.out.println(p + " :: " + p2);
    }

    public static void mystery(Point p, int n) {
        n = 0;
        p.x = p.x + 33;
        System.out.println(p.x + ", " + p.y + " " + n);
    }
}
```

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. Ask for scratch paper if you need it, and ATTACH it to your exam at the end. Use one piece of scratch paper per problem.

(11, 22)
44, 22 0
(44, 22)
55, 22 0
(55, 22)
133, 200 0
(133, 200) :: (133, 200)

For a detailed step-by-step:

https://docs.google.com/presentation/d/1ieXhGxwxTmiWXv7LvBCDgo18PTGI1TgUmKgvp55viMs/edit?usp=sharing

Problem Credit:
CSE 373 Spring 2021
University of Washington
Section 1, Question 1.1B

5.WWJD2 [?? pts]

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

Note: We didn't discuss **`toUpperCase()`** and **`toLowerCase()`** in lecture, but these methods are very straight-forward. They each are instance methods of the string object that return a *new* string object identical to the string the method was called on: except the returned string is completely upper/lower case (depending on which of the two you call).

```
public class Cat {
    public String name;
    public static String noise;

    public Cat(String name, String noise) {
        this.name = name;
        this.noise = noise;
    }

    public void play() {
        System.out.println(noise + " I'm " + name + " the cat!");
    }

    public void anger() {
        this.noise = this.noise.toUpperCase();
    }

    public void calm() {
        this.noise = this.noise.toLowerCase();
    }
}

public class CatProgram {
    public static void main(String[] args) {
        Cat a = new Cat("Cream", "Meow!");
        Cat b = new Cat("Tubbs", "Nyan!");
        a.play();
        b.play();
        b.anger();
        a.calm();
        a.play();
        b.play();
    }
}
```

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. Ask for scratch paper if you need it, and ATTACH it to your exam at the end. Use one piece of scratch paper per problem.

Nyan! I'm Cream the cat!
Nyan! I'm Tubbs the cat!
nyan! I'm Cream the cat!
nyan! I'm Tubbs the cat!

Problem Credit:
CS 61B Spring 2016
UC Berkeley
Section 2, Question 2

6. Box-And-Pointer [?? pts]

Assume that we have the two files `Node.java` and `BoxPointer.java` next to each other in the same directory.

```
public class Node {
    public String name;

    public int x;
}

public class BoxPointer {
    public static void troll(Node first, Node second)
    {
        int x = 0;
        int y = 1;
        first.name = second.name;

        trollAgain(x, y);
    }

    public static void trollAgain(int x, int y)
    {
        x = 9;
        System.out.println(y);
    }

    public static void main(String[] args)
    {
        Node firstNode = new Node();
        int x = 5;
        int y = 8;

        Node secondNode = new Node();
        secondNode.name = "Worship";
        secondNode.x = 7;

        Node thirdNode = new Node();
        thirdNode.name = "Java";
        thirdNode.x = 9;

        troll(secondNode, thirdNode);
    }
}
```

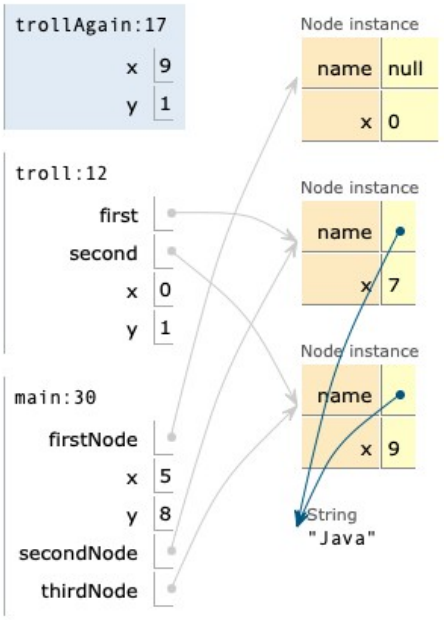
Question 6A

Draw the box-and-pointer diagram (using the **same style** shown in lecture and the slides) at the moment **right after** the underlined line of code in the `trollAgain` method has **finished executing.**

Make sure to label all of the "variable boxes" (aka stack frames) for your methods with the name of the method. Also make sure to label instance variables of an object with their variable name (just as we have in lecture).

You can use and attatch as much scratch paper as you like, but only your final box-and-pointer will be graded.

Voila:



Note: This box-and-pointer draws `null` a bit differently than I do in lecture: it just writes out the word "null". I personally like drawing an arrow pointing to an "x" to show that `null` is literally a reference/pointer that goes nowhere. I've also seen people draw `null` as an "x" or "-" inside the variable's box. In reality all of these are totally fine and make clear that you're talking about `null`. I would accept any of them on an exam.

BTW: This box-and-pointer was generated with the same online tool used in Lecture/Lab 10. That means you can ignore the numbers next to `trollAgain`, `troll`, and `main`: they're just the current line executing in the visualizer.

https://cscircles.cemc.uwaterloo.ca/java_visualize/

Question 6B

When run **until completion,** what does this program actually output to the terminal?

It prints 1.

This is because we print the `y` variable from `trollAgain`, so we need to look inside the "variable box" (or "stack-frame" if you're a pretentious computer scientist) for that method. This lets us "resolve" what value the `y` variable should have.

We see that the `y` variable inside the `trollAgain` variable box has a value of 1.

7. I Feel...Loopy [?? pts]

What does the following program print?

Hint: How many times does the inner loop iterate? How many times does the outer loop iterate? What is the difference between the **`break`** and **`continue`** statements?

```
public class Loopy1 {
    public static void main(String[] args)
    {
        int count = 0;

        for (int i = 1; i <= 100; i++)
        {
            if (i == 1)
            {
                continue;
            }

            for (int j = 1; j <= 100; j++)
            {
                count++;

                if (j == 3)
                {
                    break;
                }
            }
        }

        System.out.println(count);
    }
}
```

It prints 297.

Notice that the inner loop increments count 3 times (after the third increment we hit break, which kills that inner loop immediately).

The outer loop reaches the inner loop 99 times: on the first iteration (out of 100) we hit the continue, but that only kills iteration 1. The other 99 successfully reach the inner loop.

Multiplying like we did in the Quick Check from lecture, we get 99* 3 = 297.

8. Write Me Some Code [?? pts]

Consider the following Point class (which you should be be **very** familiar with by now):

```
public class Point {
    public int x;
    public int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

READ THIS: For each of the following questions, you will be writing a new method that goes **inside** this Point class. You may use *fewer* lines than are provided to you, but you **should not** use *more* lines.

Question 8A

Complete the outline below for a **static** method `returnNewScaledPoint`. This method should take as arguments a `Point` object (use the argument name `inputPoint`) and an `int` (use the argument name `scaleFactor`).

The method should return a **new `Point`** object that has the same `x` and `y` members as `inputPoint`, but each multiplied by `scaleFactor`. The original `Point` object (`inputPoint`) **should not change**.

```
public static Point returnNewScaledPoint(Point inputPoint, int scaleFactor)
{
     Point outputPoint = new Point(inputPoint.x * scaleFactor, inputPoint.y * scaleFactor);

     return outputPoint;

    ____________________________________________________________________________
}
```

Note: Because we have a custom constructor defined for the Point class, that means the default zero-arg constructor goes away. We can't use it anymore.

Question 8B

Complete this outline for an **instance** method `scaleThisPoint` of the `Point` class. This method should take as arguments **only** an `int` (use the argument name `scaleFactor`).

The method should **modify** the Point object it is called on. That object should have both it's `x` and `y` members multiplied by `scaleFactor`.

```
public void scaleThisPoint(int scaleFactor)
{
    this.x = this.x * scaleFactor;_____

    this.y = this.y * scaleFactor;_____

    _____
}
```

9. Write Me Some *More* Code [?? pts]

In this question, you will be filling in the `main` method of the `WhatPower` class. This is the only method in this entire class.

Fill in the `main` method so that the program:

1. Reads an int as input from the user (assume they type the int out then hit enter).
2. Prints back what "power of 2" the user typed out.

Note: The program doesn't do this forever: it literally just reads a single int as input from the user, then prints what "power of 2" that int was. The program then exits after that single print. So **one read, one print.**

Note 2: To make your life easier, you can assume the user always types out something that's an "exact" power of 2 (i.e. 2 raised to some integer power). So the user will only type out 2, 4, 8....etc.

Here is some sample input/output of what your program should do:

```
sniyaz@sniyazs-MacBook-Air practice-midterm % java WhatPower
16 <--- What the user types.
4 <--- 16 is 2^4.
[Program Exits]

sniyaz@sniyazs-MacBook-Air practice-midterm % java WhatPower
8 <--- What the user types.
3 <--- 8 is 2^3
[Program Exits]

sniyaz@sniyazs-MacBook-Air practice-midterm % java WhatPower
1 <--- What the user types.
0 <--- 1 is 2^0.
[Program Exits]
```

Fill in this program:

```
import java.util.Scanner;

public class WhatPower
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);

        int readInt = scanner.nextInt();

        int powerCount = 0;

        while (readInt > 1)
        {
            readInt = readInt/2;_____

            powerCount++;_____
        }

        System.out.println(powerCount);
    }
}
```

Note: This is essentially Lab 8 Q2, but "in reverse". In Lab 8 you went from the exponent to the result, but here we're going from the result to the exponent.

You have reached the end.

There is no more.

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