

**Mt. San Antonio College – Computer Science**

CSCI 145: Introduction to Java Programming

Sherdil Niyaz

**Practice** Midterm, 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:** \_\_\_\_\_

**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.**
- **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.

### Question 1B

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

### Question 1C

The following assignment in Java is valid:

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

### 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.

### Question 1E

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

## 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;
```

### Question 2B

What is the value of the result variable here?

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

### Question 2C

What is the value of the result variable here?

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

### Question 2D

Why does this program not compile?

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

### 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!");
        }
    }
}
```

Last 4 digits of SID: \_\_\_\_\_

CSCI 145 PRACTICE MIDTERM, FALL 2023

**Problem 1**

**Problem 2**

**Problem 3**

**Problem 4**

**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.

## 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.

## 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 attach as much scratch paper as you like, but only your final box-and-pointer will be graded.

**Question 6B**

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

## 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);
    }
}
```

## 8. Write Me Some Code [?? pts]

Consider the following Point class (which you should 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 _____  
{  
    _____  
    _____  
    _____  
}  
_____
```

**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 its `x` and `y` members multiplied by `scaleFactor`.

```
public _____  
{  
    _____  
    _____  
}  
_____
```

## 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._____  

  

        int powerCount = _____  

  

        while (_____)
        {
            _____  

  

            _____  

  

        }
    }
}
```

**You have reached the end.**

**There is no more.**

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

