Lab 6.5 (Oakland): Java Questions

Please do not read these questions until directed to do so in lab.

Work through these questions with your labmates. When you disagree on an answer, discuss why, and consult a lab leader if you can't reach agreement.

When done, share your answers with a lab leader and answer their questions about your reasoning.

Section 1: Static vs. Instance

Q1

```
public class Counter {
    static int count = 0;
    int id;

    public Counter() {
        count++;
        id = count;
     public static void main(String[] args) {
        Counter c1 = new Counter();
        Counter c2 = new Counter();
        Counter c3 = new Counter();
        System.out.println(c1.id + " " + c2.id + " " + c3.id);
```

What does this print?

Q2

```
public class Tracker {
    static int total = 0;
    int value = 0;

    public void increment() {
        total++;
        value++;
     public static void main(String[] args) {
        Tracker t1 = new Tracker();
        Tracker t2 = new Tracker();
        t1.increment();
        t1.increment();
        t2.increment();
        System.out.println(t1.value + " " + t2.value + " " + Tracker.total);
```

What does this print?

Q3

```
public class Widget {
    static int numWidgets = 0;

    public Widget() {
        numWidgets++;
     public static void main(String[] args) {
        System.out.println(Widget.numWidgets);
```

Does this code run to completion? If so, what is its output? If not, does it have a compile-time or runtime error?

Q4

```
public class Example {
    int x = 10;

    public static void printX() {
        System.out.println(x);
     public static void main(String[] args) {
        printX();
```

Does this code run to completion? If so, what is its output? If not, does it have a compile-time or runtime error?

Section 2: Overloading vs. Overriding

Q5

```
public class Logger {
    public static void log(String msg) {
        System.out.println("TEXT: " + msg);
    }
    public static void log(Integer code) {
        System.out.println("CODE: " + code);
    }
    public static void log(Object obj) {
        System.out.println("OTHER: " + obj);
    }
    public static void main(String[] args) {
        Object a = "server down";
        Object b = 404;
        log(a);
        log(b);
```

What does this print?

Q6

```
class Vehicle {
    public String describe() { return "vehicle"; }
 class Car extends Vehicle {
    public String describe() { return "car"; }
 class Tesla extends Car {
    public String describe() { return "tesla"; }
}

Vehicle v = new Tesla();
System.out.println(v.describe());
```

What does this print?

Q7

```
class Shape {
    public double area() { return 0; }
 class Circle extends Shape {
    double radius;
    public Circle(double r) { this.radius = r; }
    public double area() { return 3.14 * radius * radius; }
 class Square extends Shape {
    double side;
    public Square(double s) { this.side = s; }
    public double area() { return side * side; }
}

Shape[] shapes = { new Circle(1), new Square(2) };
for (Shape s : shapes) {
    System.out.println(s.area());
}
```

What does this print?

Q8

Overloading is resolved at ______ based on the ______ type of the arguments.

What terms should fill in the blanks?

Q9

Overriding is resolved at ______ based on the ______ type of the object.

What terms should fill in the blanks?

Q10

```
class Instrument {
    public String play() { return "..."; }
 class Guitar extends Instrument {
    public String play() { return "strum"; }
 class BassGuitar extends Guitar {
    // no play() method
}

Instrument i = new BassGuitar();
System.out.println(i.play());
```

Does this code run to completion? If so, what is its output? If not, does it have a compile-time or runtime error?

Section 3: Constructors and `super`

Q11

```
class Engine {
    int horsepower;
    public Engine(int hp) {
        this.horsepower = hp;
 class TurboEngine extends Engine {
    double boostPressure;
    public TurboEngine(int hp, double boost) {
        this.boostPressure = boost;
```

Does this code run to completion? If so, what is its output? If not, does it have a compile-time or runtime error?

Q12

```
class Appliance { }

class Toaster extends Appliance {
    int slots;
    public Toaster(int slots) {
        this.slots = slots;
```

Does `Toaster`'s constructor need an explicit `super()` call?

A. Yes — without it, the code won't compile
B. No — Java automatically calls `super()` since `Appliance` has a no-arg constructor
C. No — subclass constructors never need to call `super()`
D. Yes — `super()` is always required in every constructor

Q13

```
class Account {
    String owner;
    public Account(String owner) {
        this.owner = owner;
 class SavingsAccount extends Account {
    double interestRate;

    public SavingsAccount(String owner, double rate) {
        this.interestRate = rate;
        super(owner);
```

What happens?

A. Compiles and runs correctly
B. Compile-time error — `super()` must be the first statement in the constructor
C. Runtime error
D. Compiles fine — order doesn't matter in constructors\

Q14

```
class Phone {
    String brand;
    public Phone(String brand) { this.brand = brand; }
    public String ring() { return "ring ring"; }
 class SmartPhone extends Phone {
    public SmartPhone(String brand) {
        super(brand);
     public String ring() {
        return super.ring() + " (vibrate)";


System.out.println(new SmartPhone("Pixel").ring());
```

What does this print, or does this have an error?

A. `ring ring`
B. `(vibrate)`
C. `ring ring (vibrate)`
D. Compile-time error
E. Run-time error

Q15

```
class Base {
    public Base() {
        System.out.print('B');
 class Middle extends Base {
    public Middle() {
        System.out.print('M');
 class Top extends Middle {
    public Top() {
        System.out.print('T');


Top t = new Top();
```

What does this print?

A. `TMB`
B. `BMT`
C. `T`
D. `BT`

Section 4: Null and NullPointerException

Q16

```
public class NullDemo {
    public static void main(String[] args) {
        String s = null;
        System.out.println(s);
```

What happens?

A. Prints `null`
B. Prints nothing (blank line)
C. Compile-time error
D. NullPointerException

Q17

```
public class NullDemo {
    public static void main(String[] args) {
        String s = null;
        System.out.println(s.length());
```

What happens?

A. Prints `0`
B. Prints `null`
C. Compile-time error
D. NullPointerException

Q18

```
public class NullCheck {
    public static void describe(String s) {
        if (s.equals("hello")) {
            System.out.println("greeting");
        } else {
            System.out.println("other");
     public static void main(String[] args) {
        describe(null);
```

What happens?

A. Prints `other`
B. Prints `greeting`
C. Compile-time error
D. NullPointerException