CS 430 Virtual Method Tables Lab

PURPOSE

This lab will exercise and assess your understanding of how class instance records and virtual method tables are used to implement classes and objects in object-oriented programming languages, and how these structures enable polymorphic behavior via dynamic dispatch.

You will demonstrate your understanding by examining a code snippet, drawing a comprehensive memory diagram, and writing a short description of how a particular method call is implemented.

INTRODUCTION

Consider the following snippet in a Java-like language, where all non-primitive variables are passed by reference and method invocation uses dynamic dispatch on the receiver.

Additional note: classes marked "`abstract`" may not be directly instantiated and any methods inside an abstract class that are individually marked "`abstract`" must be implemented in any direct subclasses.

CODE LISTING

```
// original code written by Dr. John Bowers

abstract class Animal {
    boolean sleeping = false;
    void sleepToggle() { sleeping = !sleeping; }
    abstract void speak();
}

class Dog extends Animal {
    void speak() { println(“Bark! Bark!”);
}

class Cat extends Animal {
    int lol_power = 100000;
    void speak() { println(“I CAN HAZ CHEEZBURGER”); }
}

class Evil extends Cat {
```

```
    void pillage() { println("I DESTROYZ UR THATCHEDROOF COTTAGES!!1!");
    void sleepToggle() { sleeping = false; }
}

void poke(Animal a) {
    a.speak();  // CALL SITE
}

void main() {
    Animal bonnie = new Dog();
    Animal garfield = new Cat();
    Cat trogdor = new Evil();
    poke(trogdor);
}
```

INSTRUCTIONS

1. Make a copy of the Jamboard ("Lab18-Vtables-ID") for you or your group.

2. Draw a diagram of the layout of the stack, heap, and static memory sections of the program when it reaches the line of code with the comment "CALL SITE". You should include the following components, each clearly labeled and placed in the appropriate section of memory with arrows depicting pointers between components:

    a. an activation record for each active method invocation,
    b. a class invocation record for each class instance,
    c. a virtual method table for each class, and
    d. an entry for every method implementation, each placed in the appropriate section of memory.

3. Explain how the compiler implements the call to `a.speak()` at the line with the comment "CALL SITE" and how this allows for polymorphic behavior via dynamic dispatch at runtime.

4. Export the Jamboard as a PDF or PNG and submit it to the appropriate assignment on Canvas by the due date specified there.