CSSE 220 – Object-Oriented Software Development
Rose-Hulman Institute of Technology

Worksheet Polymorphic Practice

```
abstract class Pet {
    public void speak() {
        System.out.print("Pet ");
        sound();
    }
    public abstract void sound();
}
class Dog extends Pet {
    public void sound() {
        System.out.print("Woof ");
    }
}
class Cat extends Pet {
    public void sound() {
        System.out.print("Meow ");
    }
}
```

1. Pet a = new Dog();
   Pet b = new Cat();
   Dog c = new Dog();

   a.speak(); _____

   b.speak(); _____

   c.speak(); _____

   a.sound(); _____

   b.sound(); _____

2.
```
abstract class Animal {
    public void speak() {
        System.out.print("Animal ");
        makeSound();
    }
    public abstract void makeSound();
}
class Dog extends Animal {
    public void makeSound() {
        System.out.print("Woof ");
    }
    public void fetch() {
        System.out.print("Fetch ");
    }
}
class Cat extends Animal {
    public void makeSound() {
        System.out.print("Meow ");
    }
}
```

Animal a = new Dog();
Animal b = new Cat();
Dog c = new Dog();

a.speak(); _____

b.speak(); _____

a.makeSound(); _____

c.fetch(); _____

a.fetch(); _____

((Dog) a).fetch(); _____

((Dog) b).fetch(); _____

Animal d = c;

Dog e = a;

a.speak(); Pet Woof
b.speak(); Pet Meow
c.speak(); Pet Woof
a.sound(); Woof
b.sound(); Meow

a.speak(); Animal Woof
b.speak(); _____
a.makeSound(); _____
c.fetch(); _____
a.fetch(); _____
((Dog) a).fetch(); _____
((Dog) b).fetch(); _____
Animal d = c;
Dog e = a;

Animal Woof → speak() from Animal, makeSound() from Dog

Animal Meow → same logic, runtime = Cat

Woof → dynamic dispatch still applies

Fetch → c is a Dog, method exists

compile error → Animal does not define fetch()

Fetch → explicit downcast, runtime object is Dog

runtime error → compiles, but Cat is not Dog → ClassCastException

OK (no output) → implicit upcasting is always safe

compile error → implicit downcasting is NOT allowed → Java refuses without a cast