

1 It's a Bird! It's a Plane! It's a CatBus!

On a research expedition studying air traffic, we discovered a new species: the Flying Interfacing CatBus, which acts like a vehicle and has the ability to honk (safety is important!).

- (a) Given the Vehicle and Honker interfaces, fill out the CatBus class so that CatBuses can rev their engines and honk at other CatBuses with a CatBus-specific honk.

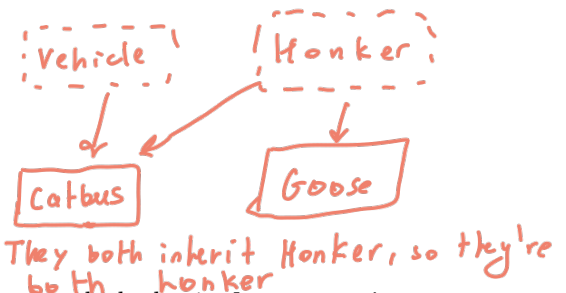
```
interface Vehicle {
    public void revEngine();
}
```

```
interface Honker {
    public void honk();
}
```

```
public class CatBus implements Vehicle, Honker {
    @Override
    public void Vehicle { /* CatBus revs engine, implementation hidden */ }

    @Override
    public void Honker { /* CatBus honks, implementation hidden */ }

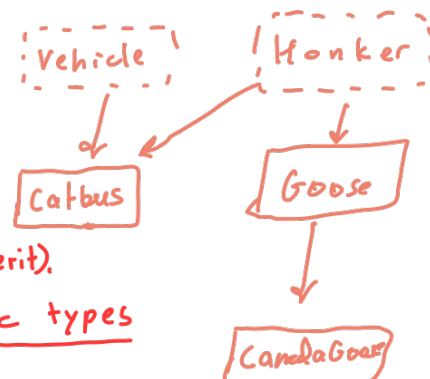
    /** Allows CatBus to honk at other CatBuses. */
    public void conversation(CatBus target) {
        this.honk();
        target.honk();
    }
}
```



- (b) It's a lovely morning in the skies and we've encountered a horrible Goose, which also implements Honker (it has a knife in its beak!). Modify the conversation method signature so that CatBuses can honk at both CatBus and Goose objects while only having one argument, target.

- (c) Assume that we have another class, CanadaGoose, which extends Goose. Which of the following lines compile?

static → Honker cb = new CatBus(); ✓
 Dynamic ↓ CatBus g = new Goose(); ✗
 Honker h = new Honker(); ✗
 CanadaGoose cg = new Goose(); ✓
 Honker hcg = new CanadaGoose(); ✓



Right hand side is guaranteed to be a left one (if they inherit).
 Interfaces can't be instantiated, they can't be dynamic types

2 Raining Cats and Dogs

- (a) What would Java do after executing the main method in the `TestAnimal` class? Fill in the table provided with the method saved at compile time, the method called at runtime, and overall output for lines 8-19 if applicable. If there is an error, write whether it is a runtime error or compile time error, and then proceed through the rest of the code as if the erroneous line were not there.

```
public class Animal {
    public String name, noise;

    public Animal(String name) {
        this.name = name;
        this.noise = "Huh?";
    }

    public void greet(Animal a) { System.out.println("Hi " + a.name + ", I'm " + name); }
    public void play() { System.out.println("I love to play! " + noise); }
    public static void sleep() { System.out.println("Naptime!"); }
}

public class Cat extends Animal {
    public Cat(String name) {
        super(name);
        this.noise = "Meow!";
    }

    public void greet(Animal a) { System.out.println("Cat " + name + " says: " + noise); }
    public void play() {
        System.out.println("Woo it is so much fun being a cat! " + noise);
    }
}

public class Dog extends Animal {
    public Dog(String name) {
        super(name);
        noise = "Woof!";
    }

    public void greet(Animal a) { System.out.println("Dog " + name + " says: " + noise); }
    public void play() {
        System.out.println("Woo it is so much fun being a dog! " + noise);
    }
    public static void sleep() { System.out.println("I love napping!"); }
}
```

```

1 public class TestAnimal {
2     public static void main(String[] args) {
3         Animal a = new Dog("Pluto");
4         Animal b = new Animal("Bear");
5         Cat c = new Cat("Garfield");
6         Dog d = new Dog("Lucky");
7
8         Cat e = new Animal("Kitty");
9         a.greet(c);
10        a.sleep();
11        c.play();
12        c.greet(d);
13        ((Animal) c).greet(d);
14        d.sleep();
15        a = c;
16        a.play(14);
17        ((Cat) b).play();
18        d = (Dog) a;
19        c = a;
20    }
21 }

```

line	Compile time (static)	Runtime (dynamic)	Output
8	cat	NA	CE (Animal isn't necessarily a cat)
9	Animal.greet	Dog.greet	Dog Pluto says woof
10	Animal.sleep	Animal.sleep	Naptime!
11	C.play	C.play	... Meow!
12	C.greet	C.greet	cat Garfield says meow
13	Animal.greet	C.greet	same as last time
14	D.sleep	D.sleep	I love napping
15	A	C	nothing
16	NA	NA	CE
17	C.play	NA	RE
18	D	NA	RE
19	NA	NA	CE

durig runtime we're making sth general (Animal), very specific (cat)

cat class is overriding

(b) Spoiler alert! There is an error on the last line, line 19. How could we fix this error?

$c = (\text{cat}) a;$