2 Raining Cats and Dogs

Assume that `Animal` and `Cat` are defined as above. What will be printed at each of the indicated lines?

```
1  public class TestAnimals {
2      public static void main(String[] args) {
3          Animal a = new Animal("Pluto", 10);
4          Cat c = new Cat("Garfield", 6);
5          Dog d = new Dog("Fido", 4);
6
7          a.greet();           // (A) _____________________
8          c.greet();           // (B) _____________________
9          d.greet();           // (C) _____________________
10
11         a = c;
12         a.greet();           // (D) _____________________
13         ((Cat) a).greet();   // (E) _____________________
14     }
15 }
16
17 public class Dog extends Animal {
18     public Dog(String name, int age) {
19         super(name, age);
20         noise = "Woof!";
21     }
22
23     @Override
24     public void greet() {
25         System.out.println("Dog " + name + " says: " + makeNoise());
26     }
27
28     public void playFetch() {
29         System.out.println("Fetch, " + name + "!");
30     }
31 }
```

(A) Animal Pluto says: Huh?
(B) Cat Garfield says: Meow!
(C) Dog Fido says: WOOF!
(D) Cat Garfield says: Meow!
(E) Cat Garfield says: Meow!

Consider what would happen we added the following to the bottom of `main`:

```
a = new Dog("Hieronymus", 10);
d = a;
```

Why would this code produce a compiler error? How could we fix this error?

This code produces a compiler error in the second line. The **static** type of d is Dog **while** the **static** type of a is Animal. Dog is a subclass of Animal, so **this** assignment will fail at compile time because not all Animals are Dogs.

We can fix that by using a cast:

```
d = (Dog) a;
```

This represents a promise to the compiler that at runtime, a will be bound to an object that is compatible with the Dog type.