```java
public class Puzzle13 {

    public static void main(String[] args){
        A b1 = new B();
        A c1 = new C();

        A b2 = b1;
        A c2 = c1;

        // what will get printed?
        b2.print(c2);
    }

    public static class A extends Object {
        public void print(A object) {
            System.out.println("An instance of " + object.getClass().getSimpleName()
                    + " was passed to A's print(A object)");
        }
    }

    public static class B extends A {
        public void print(A object) {
            System.out.println("An instance of " + object.getClass().getSimpleName()
                    + " was passed to B's print(A object)");
        }
    }

    public static class C extends B {
        public void print(A object) {
            System.out.println("An instance of " + object.getClass().getSimpleName()
                    + " was passed to C's print(A object)");
        }
    }

    public static class D extends A {
        public void print(A object) {
            System.out.println("An instance of " + object.getClass().getSimpleName()
                    + " was passed to D's print(A object)");
        }
    }

}
```

Review

- Static vs. Dynamic Dispatching

Call Graph Construction

- Reachability Analysis (RA)
  - Simply matches callsite signatures to corresponding function signatures
  - Create a call relationship from each callsite to every matching function
  - Very imprecise, but very cheap and easy to implement