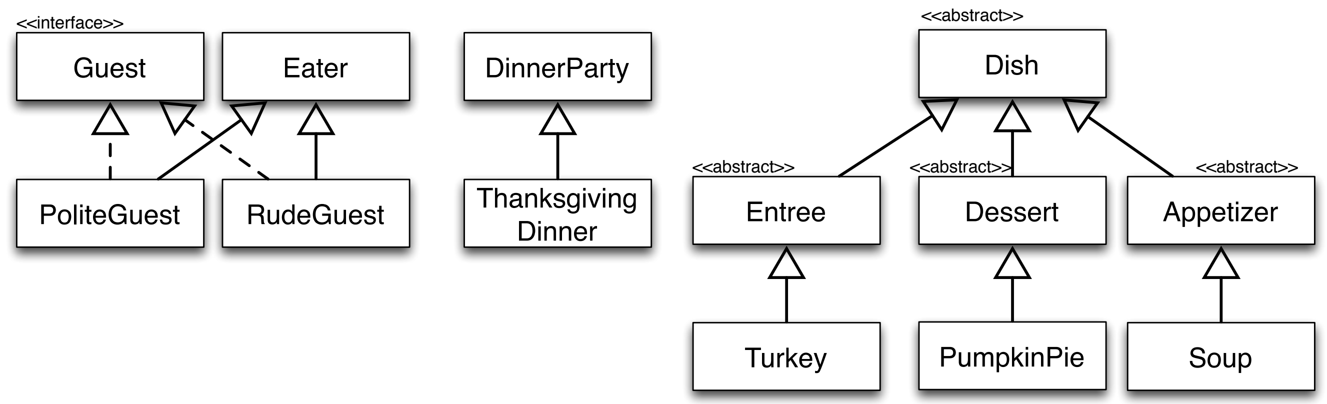
1.



2. Would compile: a) iii, vi b)i, ii

Won't compile:

a)i — Guest is an interface

a)ii — Guest is not a subtype of PoliteGuest

a)iv — Turkey is not a subtype of Soup

a)v — Appetizer is an abstract class

3.

- Set up a new Eater. (Eater e = new Eater())
- Determine initial hunger level of the Eater. (int hunger = e.getHunger())
- Make a new dish (Dish d = new Soup())
- Determine food value of that dish. (int foodValue = dish.getFoodValue())
- Call eat on that new Eater. (e.eat(d))
- Use an assertion statement to check that the Eater's hunger has decreased by the food value. (assertTrue(e.getHunger == hunger - foodValue)

4. Requires: Nothing.

Modifies: This or this.guests

Effects: Each dish is served to each guest.

5a.

```
Making soup!
Serving appetizer
                        Soup!
Thank you. This looks lovely!
Slurp Slurp
Done eating
```

5b (one option):

```
private void serveCourse(Dish d, Guest g){
        //anything local to this block must be in the parameter list
        g.serve(d);
        d.waitForEatingTime();
        g.clear(d);
        //return value must be returned if there is one
}
```

5c (the call for the above option) (return value and parameters must match)

```
this.serveCourse(dish, guest)
```