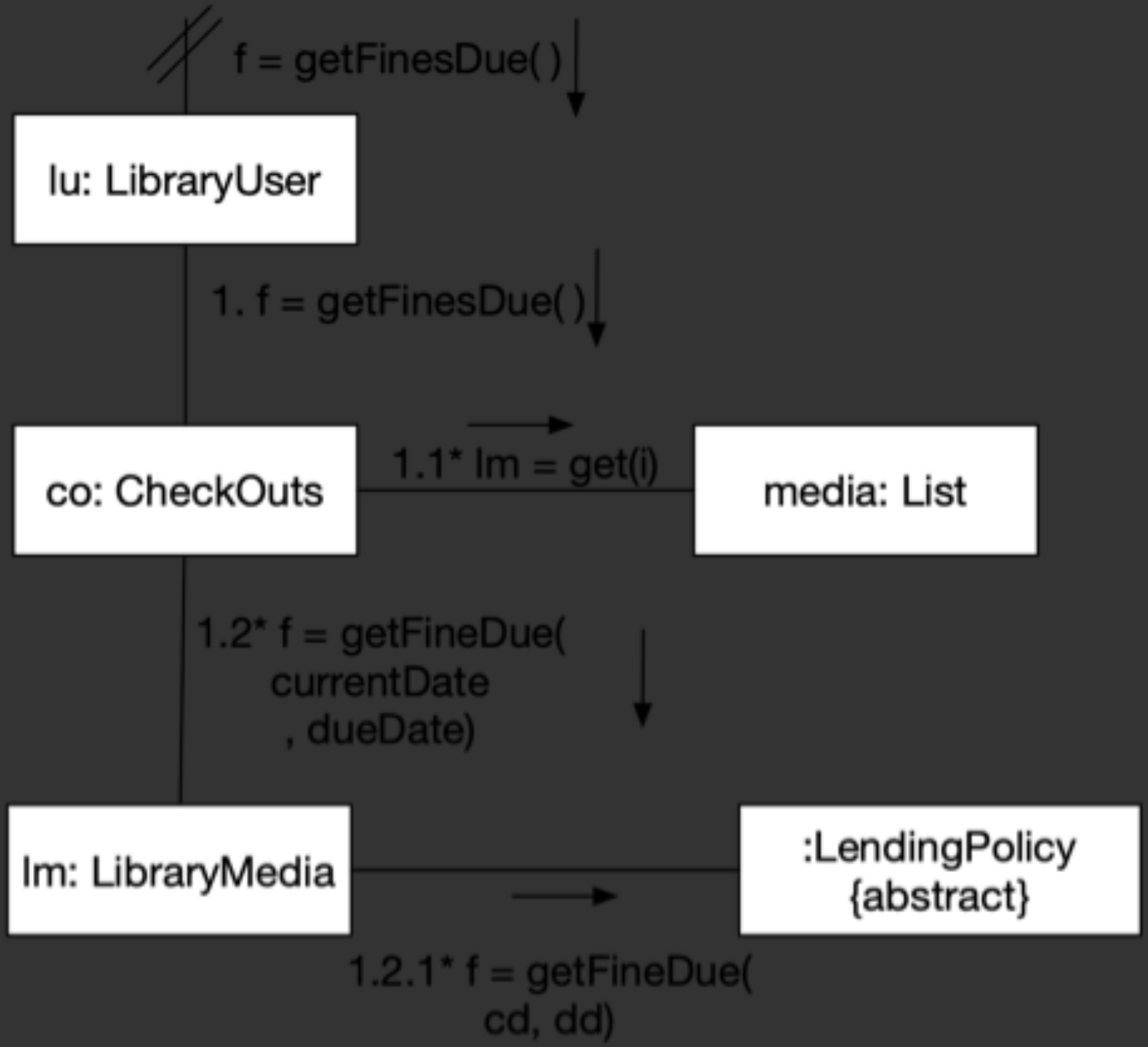
Communication Diagram

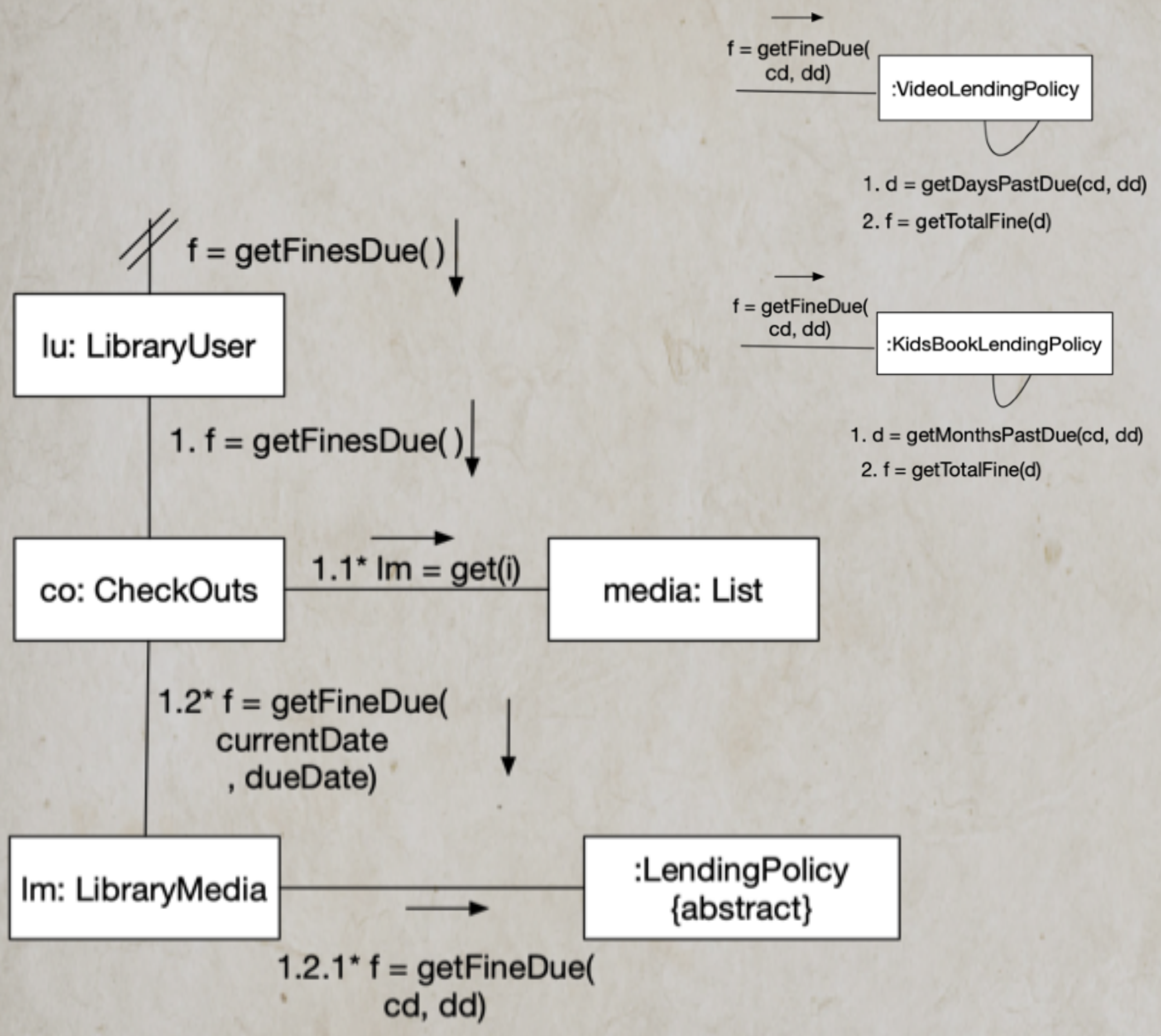


```
     class LibraryUser {
        CheckOuts co;
      public double getFinesDue() {
1.      return co.getFinesDue();
      }
     }

     public class Checkouts{
      List<LibraryMedia> media;
      Date currentDate; Date dueDate;
      public double getFinesDue() {
        double f = 0;
        for each lm in media
1.1       f+= lm.getFineDue(currentDate, dueDate);
1.2     return f;
     }

     public class LibraryMedia{
      LendingPolicy lp;
      public double getFineDue(date, date) {
        return lp.getFineDue();
1.2.1 }
     }
```

Communication Diagram



Notation Tips

- Each box represents an instance of something (usually a class)
- Each line represents a communication path between two instances. Only one path ever exists, even if multiple calls are made between the two. A looping line represents self-calls (normally private helper-methods)
- Each call is shown, in order, using a nested numbering scheme. An arrow shows which direction the call is made. Variables can be used to show return values and inputs.
- An asterisk can be used to denote looping / iterating multiple times.
- To illustrate polymorphic or abstract calls, use {abstract}, and then another diagram to show specific implementations.
- Use two parallel lines to denote calls coming through an API