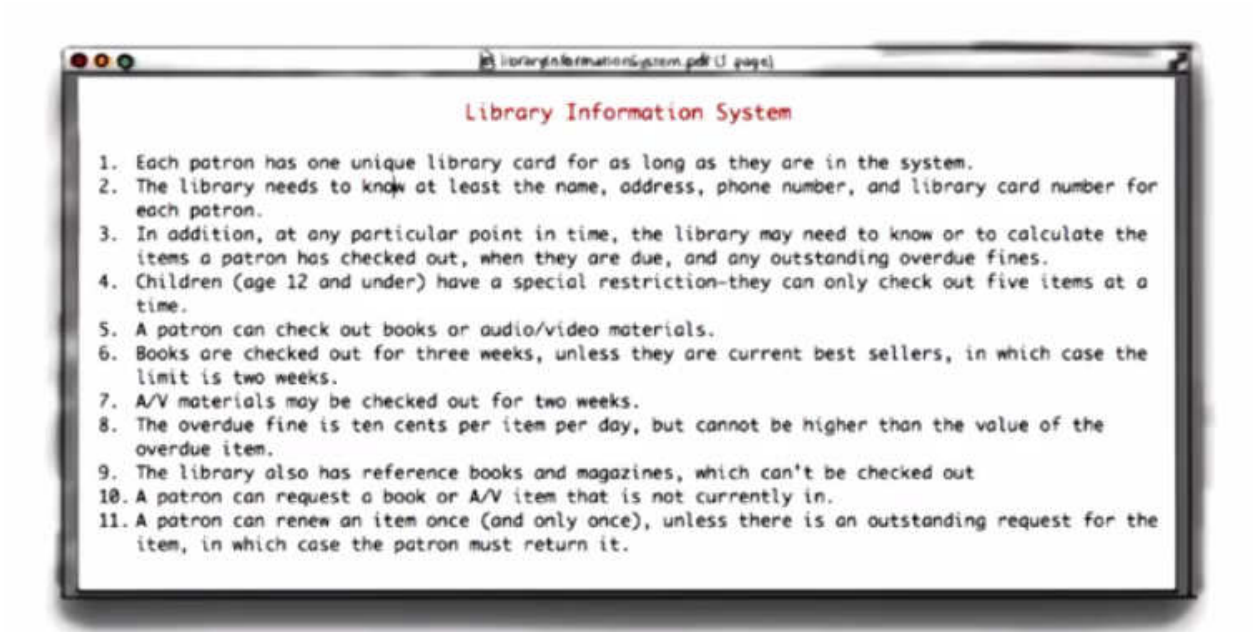
P3L2 A Tale of Analysis and Design

09/22/2014 Notes

Welcome to a tale of analysis and design, featuring Spencer Rugaber, as the librarian, and Alex Orso, as the software engineer.

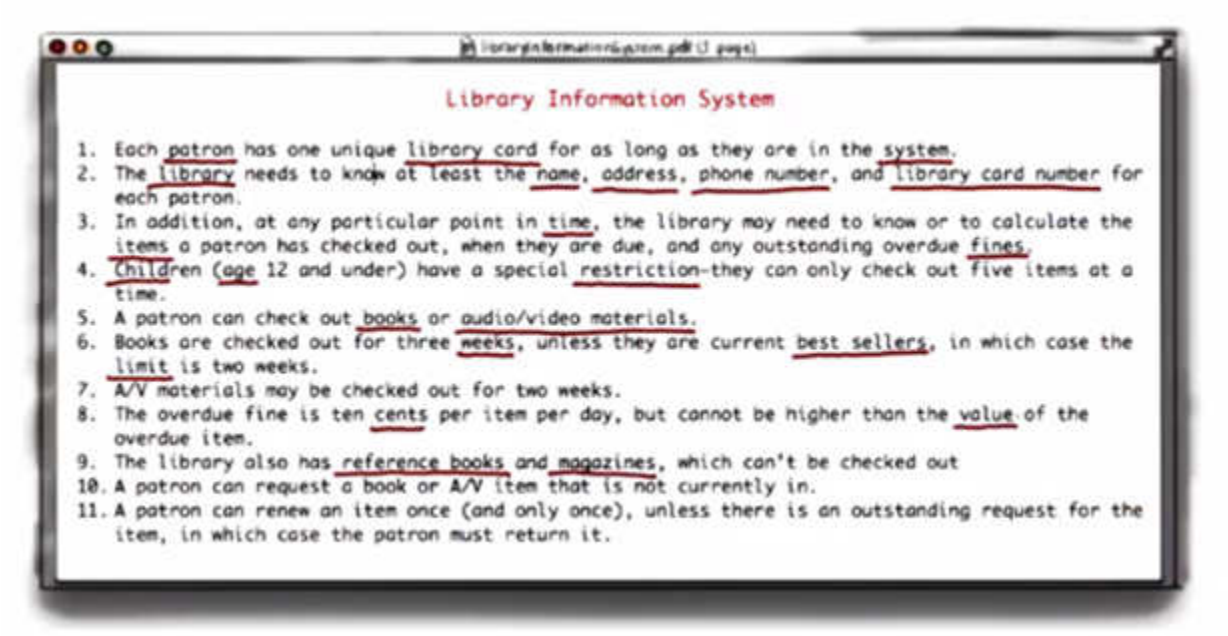


Library Information System

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction–they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can't be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

Introduction

- Hi! I'm here waiting for Spencer, my librarian friend. He needs some help developing an information system for a library. So I asked him to write down the requirements for the library. [KNOCK] that must be him.
- Hello Alex.
- Hey Spencer. How's it going?
- Good. Did you get those requirements I emailed you?
- Oh, you emailed them. Now let me check. And, by the way, got some coffee for you here.
- Thank you very much.
- Oh yeah. They're right here. Let me see. Oh, good. Oh, yeah, good. We have what we need. So the way I like to do this is **I like to start by looking at the requirements and identifying the nouns in the requirements, because those tell us the kind of the relevant elements in the requirements.** So if you don't mind we can start looking at those and you can tell me whether the ones that I identifying make sense or not.
- Sounds good.
- All right.



Library Information System

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction-they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can't be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

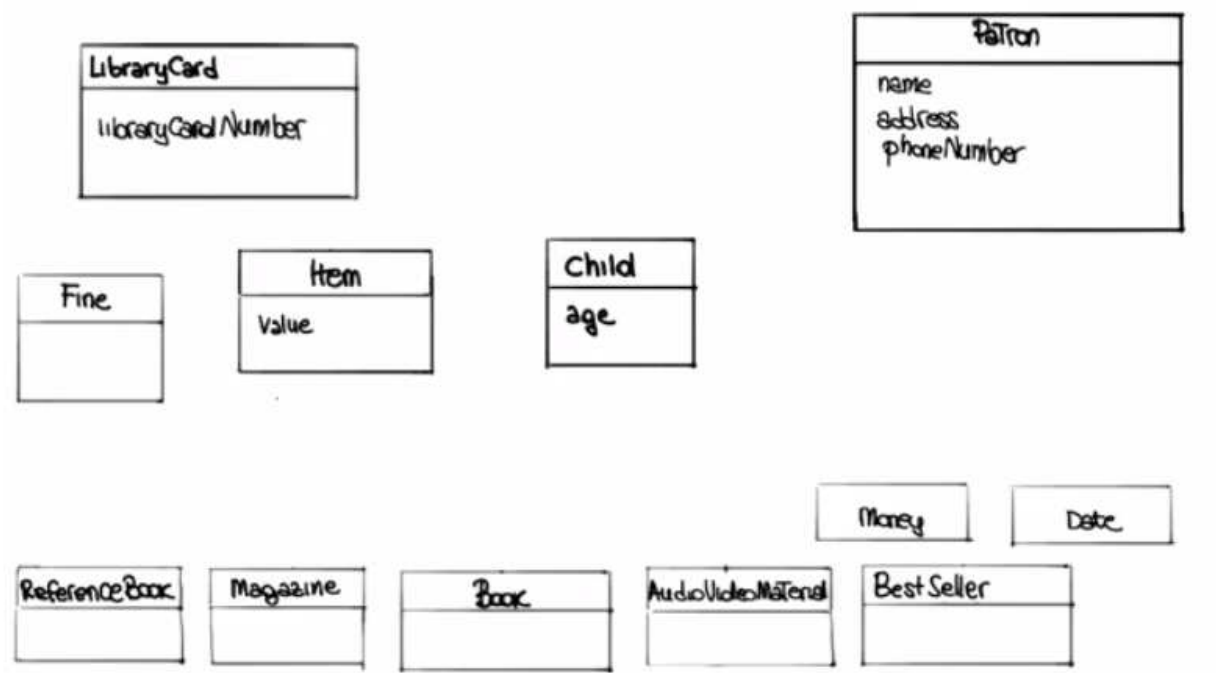
Analyzing Requirements

- Okay so let me start underline these nouns, and I'll start identifying the ones that are relevant, and I'll ask you some questions or you can ask me questions if you see something that doesn't make sense to you. Good enough?
- Okay, let's see, patron. It seems to me that patron is definitely an important entity.
- That's what it's all about.
- Okay, all right, so actually the way I'm going to do this, is I'm going to take all these relevant entities and I'm going to start putting them into what I call a **class diagram. So you don't really need to know what that is exactly, but imagine this being a diagram in which I represent all development items as rectangles with a given name and, and then later on some attributes.**
- Okay.
- Okay, and I'm just going to put them there. So I'm going to start with patron. I'm going to create one class for the patron. I'm going to give it the name patron. And by the way, assuming that you'd probably figure out, **it's important that we represent, that we use the right names so that it's clear when we're looking at the class diagram what we're referring to. So I'll just use the, the nouns themselves as, as names.** Okay, library card seems to be also a relevant element.
- Every patron has a library card.
- All right, perfect, so we'll just create a library card here. And let's see. As, as long as they're in the system. And I saw that there's a system here, this concept of system, this concept of library. And based on my experience, normally, those are kind of overarching themes. So this is really what we are modeling. So the only thing that will make a

difference is if there were more than one library or more than one system. Is that the case?

- We just want one system for our one library
- Okay so, in this case I won't even represent those because basically what I'm representing is the system and the library.
- I understand.
- It's kind of a new basic concept. Okay and then, oh name, address and phone number are interesting because these are important entities, but this seems like, you know, they're not entities in themselves, so they, they're more attributes of something else. I would imagine that this is the way you identify, or these are elements that are important for the patron?
- That's what we take down when we issue the cards.
- Okay. Perfect. So, I'm going to **take those and make those attributes of the patron, which means that I'm going to take the class that I created before, and I'm just going to write them down here so that they're represented** and, and we know that these are kind of what characterizes the patron.
- Gotcha.
- Okay? And then, I guess similar consideration for the library card number. So this is to be associated with the library card?
- It's printed right on it.
- All right. So we'll put this as an attribute of the library card, then. And in addition, "at any particular point in time". Okay, so time seems to be a relevant entity right, because time seems to occur several times in this descriptio?. For example, I think you guys keep track of how long a book has been loaned, right?
- Right.
- And there's some time associated also here.
- And a child's age.
- Oh yeah. The children's age here that I didn't see before. Yeah. So, what I'm going to do, I'm going to represent this in a sort of generic way, as a **date**.
- Okay.
- **These are kind of classes, utility classes we call them, that are normally in every system.**
- Okay.
- So I'm just going to put it down here as \ as a utility class that will be used by different elements in the diagram. Okay, so I want to calculate the items. So the items from what I know about libraries seem to be pretty relevant elements, right?
- This is what we check out, this is what we're for.
- Okay, so then items definitely will become a class. Oh there's also this concept of fines. I guess that seems to be important, right? You gives fines to people who are late.
- Right.
- Right, collect fines and so on. So we create a fine class down here and the children. So children are special customers, right? It's their age that makes a difference? Is that the way it works?
- Right. They, they can only check out a few books.

- Okay. So I, I'll create them a special kind of case, a special kind of customer so I just create here a class for children. And I can see that they're categorized by their age.
- Right.
- So I'll just put the age here as an attribute of the child. Okay, so the next one is restriction. And restriction is kind of tricky because it seems to be sort of a general concept. I mean, in a sense, all of those are restrictions, right?
- Right, this is just another one of these requirements.
- Oh, okay, so, so we don't need to represent it explicitly, right?
- Right.
- It's just telling us how the child, yeah, okay, right; this is just another requirement, so I just won't consider that for now. And oh, I see that these books and audio video materials, I guess these are things that the patrons can check out, right?
- Those are some of the items, right.
- There are two more down here, right? Reference books and magazines?
- But, they can't be checked out, but they're definitely in the library.
- Okay, so then I'm going to represent all of those now. So, I'm going to have books, I'm going to have audio video material, reference books, and magazines. And I'm just going to have those as classes. Then, okay here we have week, and, we already represented this general concept of time, so week will be represented by the date class as well. And oh, I see best sellers. So best sellers are also, I guess, items that can be checked out, right?
- Right.
- Okay, so I'll just represent those as a class as well and an additional item that is relevant for the library. And the limit, this is also a time limit, right?
- Right.
- So it can also be represented with a class. Oh, here we have cents, and for cents, we have the same consideration that we made for time. This is a kind of money, is a general concept that currency is in many IT systems. So, I'm going to just have a money class here, which is another utility class.
- Okay.
- Okay, and here I have value, so value is a property. Let me look again at the requirement. Oh, it's the value of the item. So value I'm going to put in the item as an attribute. Okay?
- Okay. That's how much it cost us.
- Okay. Perfect.
- Seems like we got them all. Right? Anything I forgot?
- That looks like it.
- Okay, so this one, what I'd like to do. We have a kind of a first take, first cut at the class diagram. I'd kind of like to move to that and go through the different classes with you. And I'll ask you some questions again. And you can tell me whether there is something that comes, jumps at you that's not right. And then we're going to try to refine that. Okay?
- Okay.
- Sounds good.

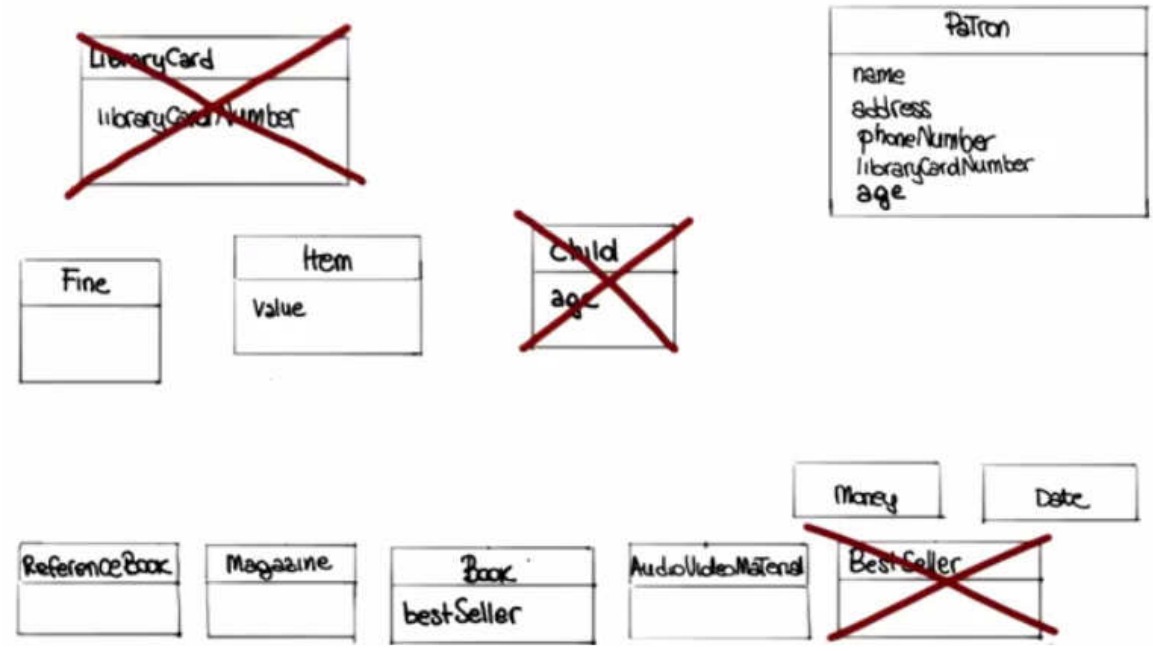


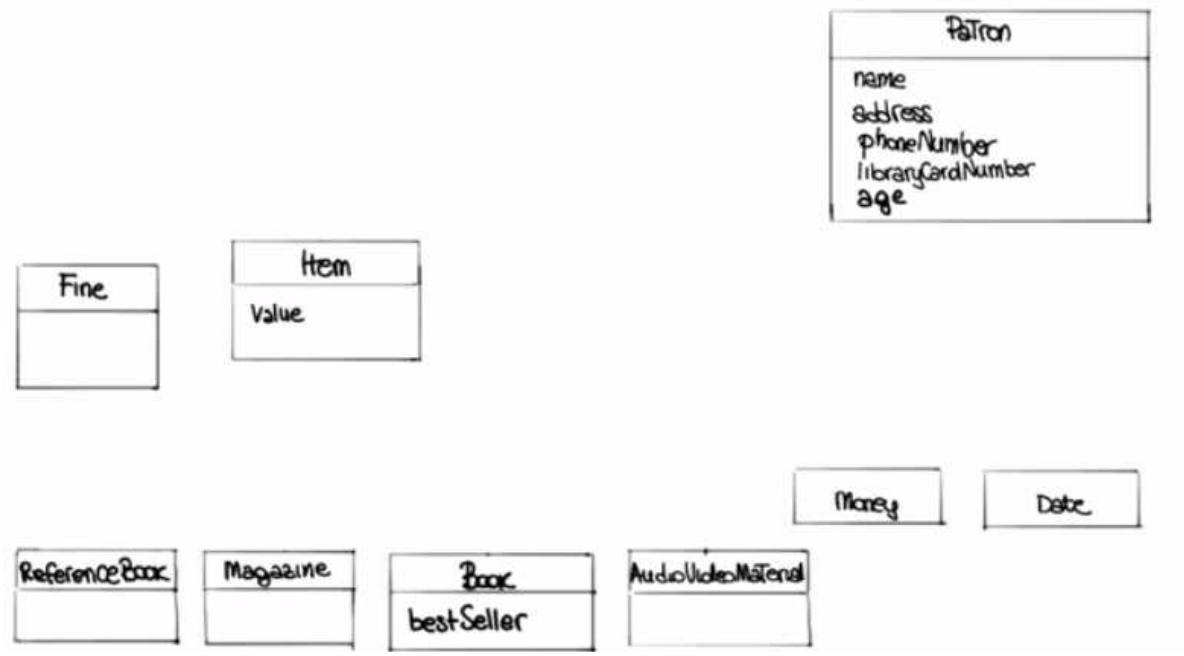
Refining Classes and Attributes

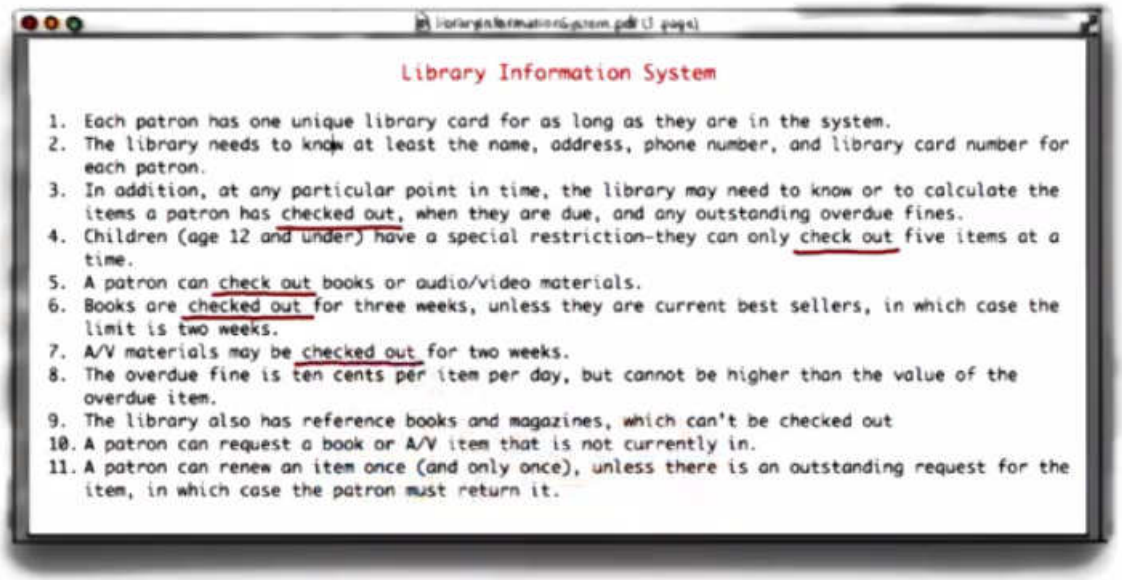
- Okay, so this is our first class diagram.
- So, let me ask you something about what we've done so far.
- Okay.
- I also sent you some stories about how the actual Library is used. You asked me to do that and are we going to use that here?
- Glad you asked actually. Yeah. **Those are, what we call use cases, or what we will use as scenarios, the kind of things that we will use to derive use cases. And they're also a very good way of extracting requirements. We're not going to look at them right now because now, we're more working on the static structure of the system.** But after we're done with the class diagram, you know, we will do it at a different time. But we're going to use those to see how the libraries actually use them, and see whether we can get more information that we can use to refine our requirements based on that.
- Okay.
- Okay, so, for now, we'll just focus in on the structure, but, just so you know, I'm glad you sent them because they are going to be very useful as well. Okay. So let's see. Well, first of all seems like that this is already pretty crowded, right? We have a number of, classes. So let's see if there's some class that may be superfluous and we can model in a different way. So, for example I was thinking the library card, it doesn't really contain much information, right? So is it basically just the number?
- The card has a number on it. We have a separate vendor that does that for us.
- Oh.

- It doesn't need to be part of this system. We just have to make sure that every patron has a, a library card.
- Okay, so basically for you, in a sense, the library card is just an ID that gets associated with a patron.
- That's right.
- So I think that the best way to represent this, I would like to take the library card number and add it to the pattern.
- Okay, makes sense.
- Okay, so I'll add it here. And as an additional attribute. Okay, and it will eliminate this class.
- Okay.
- Okay. Oh, and, wait a second, so I guess also the child needs a library card number, right?
- Child needs a library card number, but let me ask you about that. Is child a separate class, or is it just another kind of patron?
- Oh, I see, I see. Because, yeah, it is sort of a special patron, right? And, so maybe we should represent it as a kind of refinement of the patron. Hm, but then that made me think. So what is the only thing that characterizes children? Is it just the age?
- Well, that they can't check out more than five books.
- Okay. And and the only difference is the fact that they are less than, you know, twelve years old.
- Twelve or less, right.
- Twelve or less. So, I guess I would probably like to represent this by making the age explicit in the patron rather than to represent it as a class. And I'll tell you why. One of the issues that might happen is that there are patrons that are children and they're no longer children when they become 13 or older right.
- Right.
- And if we represent them with a separate class in a sense, then we cannot really change the type of an instance of this classes. So we're left to kind of destroy the patron, create a new one. So that means we also have to transfer any history and we want to keep history and so on. So I think I kind of like better the idea that we represent the age exclusively in the patron, and then it'll behave differently, based on whether the patron is 12 years old, or younger, or 13 or older.
- It makes things a little simpler.
- Okay, and it allows us also to eliminate one class here. So I'm going to proceed this way. I'm going to eliminate the children class, and I'm going to put the age in the patron. Okay, and let me see. In this spirit, actually, something else that jumps at me is this idea of the bestseller, because I kind of feel like, we might have the same problem. So, what is the story? What is a bestseller?
- Well it's an item that we want to restrict how long people can keep, because there is such demand for it.
- I see, and so basically a book that's a bestseller, like the New York Times bestseller, is a bestseller forever?
- No, no, no. It's hot for awhile, and then it becomes just a regular item.

- I see. Hm. Then I guess it's a similar situation to the one I was mentioning before, right?
- Okay.
- That if we have a book, it will kind of have to change its type if it becomes a best seller. Then we have to change its type again, if it's no longer a best seller.
- Right.
- So it seems to me that a better way to represent this, is just to eliminate this BestSeller class and instead, I'm going to put the best seller attribute, which would just be a Boolean in the book.
- Okay, what do you mean by Boolean?
- The Boolean is basically just a number. It can have two values, right? True or false.
- Okay.
- So we normally use it in this case. Imagine one, zero, right? Then it's just kind of the basic.
- Okay.
- You know, the bits, right?
- Okay.
- So, this is just telling us, it's like a flag that is telling this book is a best seller, or not.
- Okay.
- It's very easy to change this value and make a book a best seller or not a best seller, than just creating and destroying instances of these classes.
- Okay, makes sense.
- Okay, so at this point, this already looks better, right? Because we have, less classes, and I think we did some serious cleanup. That's good. Okay, so now that we eliminated some of this, what I would like to do, as I said, we are going to both clean up, but also refine. I would like to go back to our, requirements and see whether we can identify additional attributes for this class that maybe are not as obvious as the one that we saw so far, okay?







Library Information System

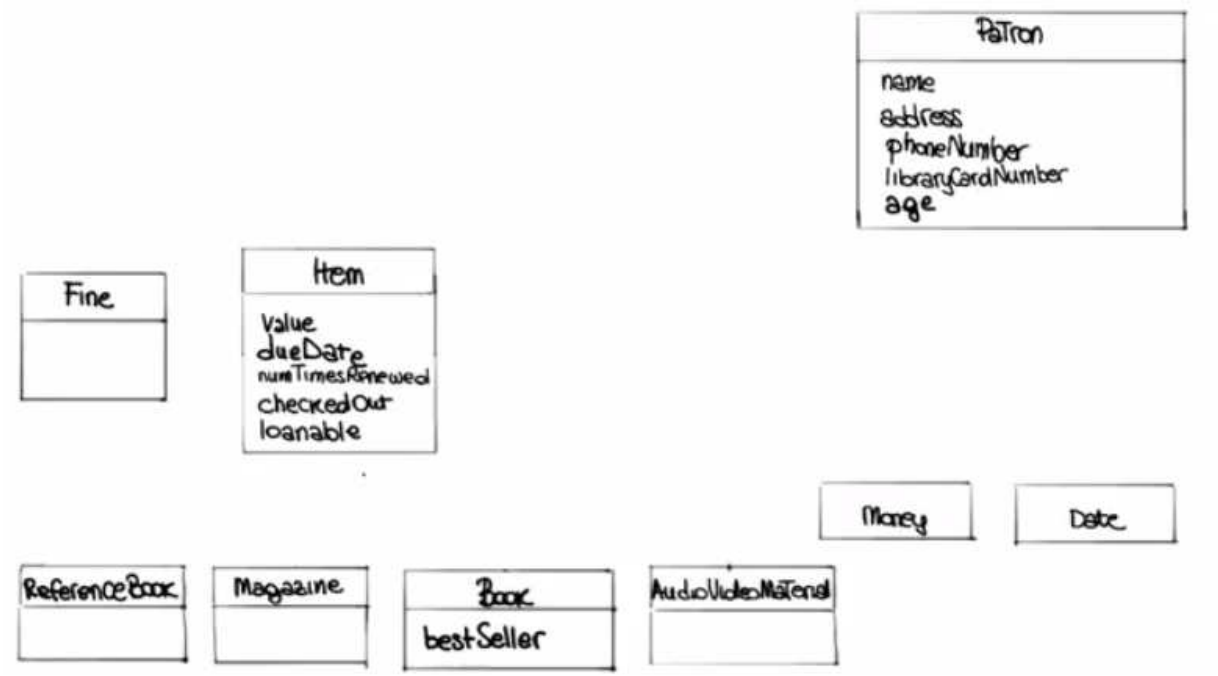
1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction-they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can't be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

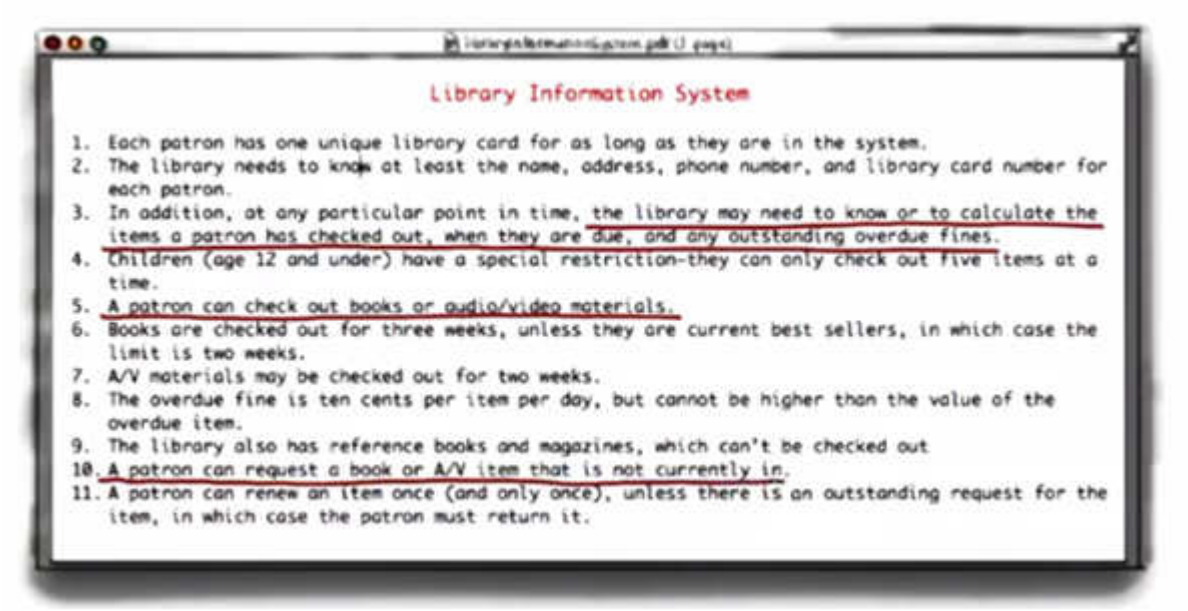
Adding Attributes

- Okay, so let me look at the requirements and it's something that I can see here that we didn't point out before is that there seems to be clearly some concept of a due date. And I'm telling you why I'm saying that because here, for example, I notice that it says when items are due. We mention overdue several times, so is this something we need to keep track of?
- Yeah remember when we used to stamp them on the books? In the stamp pad?
- Oh yeah! Oh course!

- Right? Yeah we definitely keep track of it. The system has to keep track of when books are due.
- Okay. So it seems to me that one good way of doing that is by basically adding an attribute to the item.
- Okay.
- And I'll just call it due date.
- Okay.
- So basically for each item in case it's loaned, there will be this attribute that will contain the value of when the item is due. And then, something else that I noticed here is it seems like the requirements are saying that an item can be renewed only once. So, I guess, that's something we need to keep track of, right?
- Yeah.
- The system needs to know?
- We have to know whether they've renewed it or not.
- Okay so, I'll do a similar thing here. I think I want to go and add an attribute that we'd call number of times renewed, and add it to the item class.
- Okay.
- And this is kind of more generic than what you need, because here it says only once, but let's say that in the future you want to allow it to renew twice, then you'll be able to use these attributes again because we can just count how many times it was renewed. Okay?
- Makes sense.
- Alright. One last thing I want to point out. And this seems obvious but I'm going to check with you anyways. And seems like there is basically the need to keep track of whether an item is checked out or not. If you look at the text here, the requirements here, I can see that check out and checked out are mentioned five times. So, I'm assuming that that's something also that we want to know about an item, whether it's checked out or not.
- We have to keep track of whether they're checked out.
- Okay, so I'll add an additional attribute there. So I'm going to again go back to the diagram and I'm just going to write here also the checked out attribute. And, I think that's it as far as I'm concerned. Is there anything that you think is missing?
- Well, I do have a question. Wit checked out, it better not be the case that someone can check out a reference book.
- Oh, I see.
- Okay. I mean, it's only the books and the audio visual material that can be checked out.
- Right. Okay, so I guess the way I will fix that is I'll probably put yet another attribute in the item class, and I'll call it loanable. And basically, this attribute is just telling us whether an item is loanable or not. So, when it's not true, when loanable is not on, basically that item cannot be checked out.
- Okay. And, the system would know this?
- The system will know that.
- And ban it from happening.
- And ban it from happening. Oay?

- Alright.
- Perfect. So, we're going to do that and any other objections?
- No, that was my question.
- Okay, perfect, so what I'm going to do next, I mean, I haven't mentioned that yet, but you know classes right now **we just looked at the attributes that give you sort of the state of the class.** And there's something else, **there's a second part of the class, kind ofan orthogonal aspect, which is what the class can do**. And we call those operations. So normally these kinds also have operations. And one way, **one very natural way to identify operations is to look at the requirements and look for verbs. Because verbs associated with an item will tell you basically what the item can do.**
- Okay.
- So I, I'd like to go back to the requirements and start, the same way in which we underlined, nouns, we're going to underline verbs and we're going to see **which ones of those verbs actually represent actions that we want to represent explicitly, model explicitly in our class diagram.**
- Okay.
- Okay.





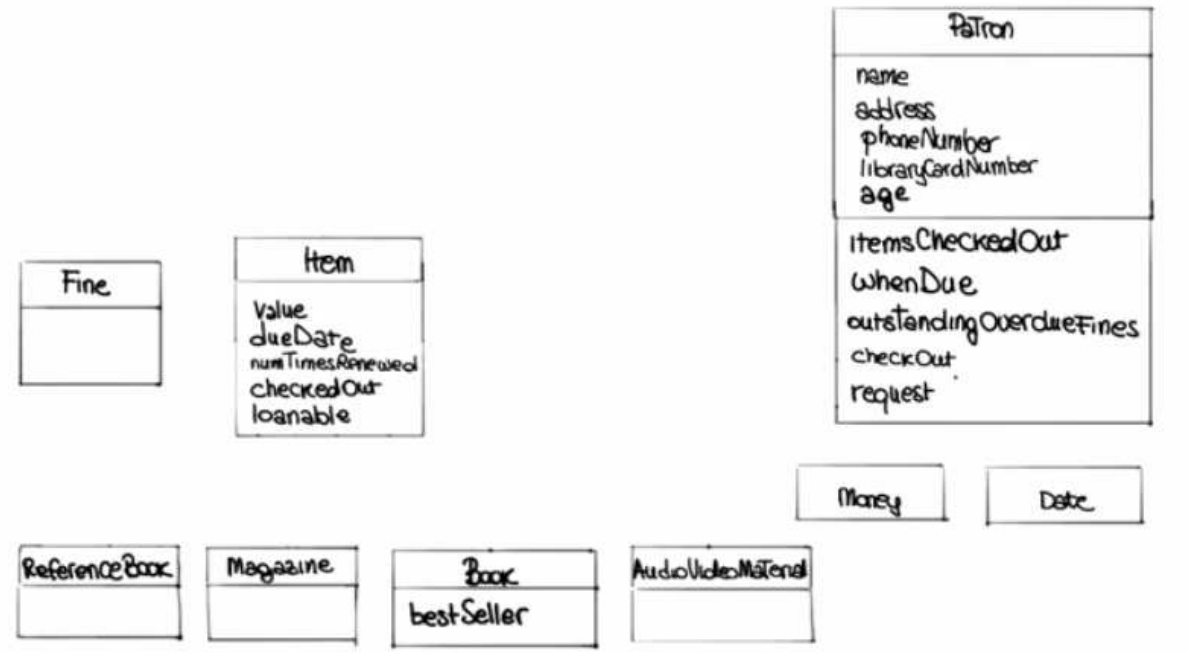
Library Information System

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction-they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can't be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

Identifying Operations

- And before we get started actually, I'd like to mention that there's different kinds of verbs because **what I'm looking for is really action verbs.** So verb, verbs that clearly express an action that can tell me for example, what an item could do, okay?
  Not the verbs that represent, for example, relationships, okay?
- Okay.
- So, the ones that I've identified and, underlined here actually, I underlined complete sentences so that you can look at the verbs in context. And the first one is this sentence that says that the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines. So I will imagine that this is representing a situation in which you bring up a patron's record and you start looking up this information.
- The, the patron often wants to know what they have currently checked out.
- Oh, alright.
- Or when are they due or how much they're owed.
- Oh, now that you mentioned it, I think one of the scenarios you sent me had to do with the patron coming in and asking for this information. So yeah, and it makes a lot of sense. So what I'm going to do, I'm going to model this by adding this three operations to the patron method. The first one, I'm going to call, itemsCheckedOut and, basically, it's an operation, but you don't need to, you know, understand the implementation details, but when you call this operation, it will give you back exactly this information, so the items that are checked out by that patron. The second one, I'm going to call it whenDue. That will tell you basically when an item is due. And the third one is going to be called the outstandingOverdueFines and as the name says, it's going to tell you what are the outstanding overdue fines for that patron.
- Okay.

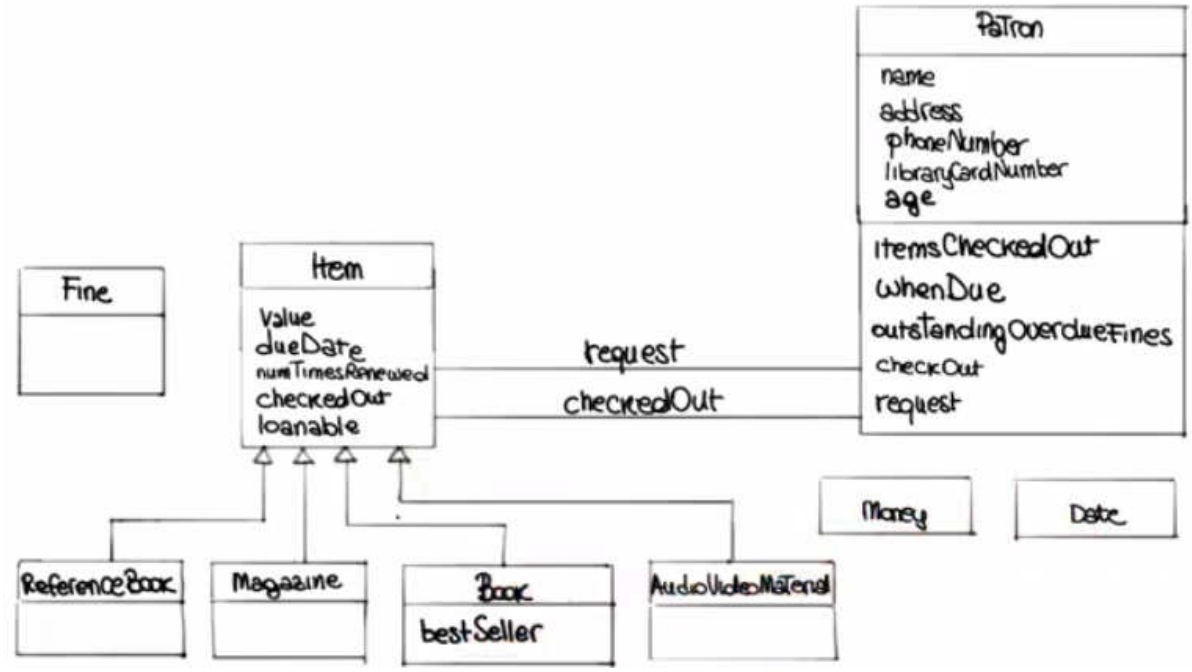
- And as you might notice **I'm going to separate the, the attributes from the operations by having a separate kind of sub-rectangle. In this way, it's clear what is an attribute and what's an operation.**
- Gotcha.
- And let me see then. Okay, for the second one you can see that that patron can check out books and audio visual materials. So I guess, similarly you build kind of the record for a patron. The patron will give you an item and you will record the fact that the patron is checking it out.
- Right. And is that operation related to the checked out attribute that we did a minute ago?
- It is actually because what will happen then, if we jump ahead a little bit, would be that every time you invoke this operation (I'm going to represent this as a checkOut operation for the patron), you will also have to say something about the item and so we will also flip that build information in the item.
- Okay.
- Okay? And, finally, here I can see that a patron can request a book or an audio video item that is not currently in. So I guess this is referring to items that are already checked out but for which there is interest. Is that it?
- Right. So, particularly, the popular items the patrons want to get on the list so that they get notified when it comes back in and check it out.
- I see. Okay. Then I'm going to do the same thing here. I'm going to add this method, which I'm going to call request and I'm going to put it here in the list of the methods in the list of operations for the patron, okay?
- Okay.



Adding Relationships

- OK I like the way this class diagram is coming along. So at this point I think we have all the classes that we need. For each class we specified the attributes or the characteristics of the class. And we also specified the operations so we know what the class can do
- And, I like to kind of move forward on this, but I first want to see that you're fine with the class structure. So that's the way the class structure is going to be in terms of attributes and operations. So anything that bothers you?
- Well, one thing I didn't understand is how come you put check out over with the patron when it's really the item being checked out?
- Right. Okay. So that actually is you know, perfect segue way, for the next thing that we really wanted to model. What you're talking about is basically a relationship between two classes which is something we haven't touched on yet.
- So we haven't looked at individual classes. But now, it is typical, now we are looking more at requirements, we're starting to find more things about our system, and what you've pointed out right now is the fact that patron and item are somehow related.
- So this checkout operation is not really something that belongs only in the patron, because it needs to know about the item. And it doesn't belong only on the item because it needs to know about the patron. So, it's something that **associates** the patron and the item.
- Okay? And that's exactly the way we call it in UML which is the notation that we're using here for this kind of relationship. So, **we're going to represent that by drawing a line between these two classes that tells us there is an association.** And we're also going to give a name to this. Since this refers to the fact of checking out Items, we're just going to call it, checkout.
- Gotcha.
- And notice that this basically will end up kind of replacing this attribute. Because the existence of this association will tell us that this is checked out. We're not going to do it right now, but in the final cleanup of the diagram, this name will disappear. Okay?
- Okay.
- And so since we started talking about relationships and associations, is there any other kind of relationship that you see here?
- Well, what you just did with checked out is, it seems similar to the whole issue of requests.
- It is. So a request is something else that happens in both the patron and in the item. It involves both. And in fact in a request, I would definitely represent this as an additional association. So I will just draw an another line between these two that represent that specific kind of relationship and I will call it request. So that indicates that this association refers to a request that also connects the patron with an item.
- Okay. And, let's see. Anything else that jumps at you?
- Yeah, well, how about all these ones down at the bottom? I mean book and item's got to be related, right? A book is a kind of item, and audiovisual. Are there associations between them?
- Can you repeat that, what you said about the book, yeah?

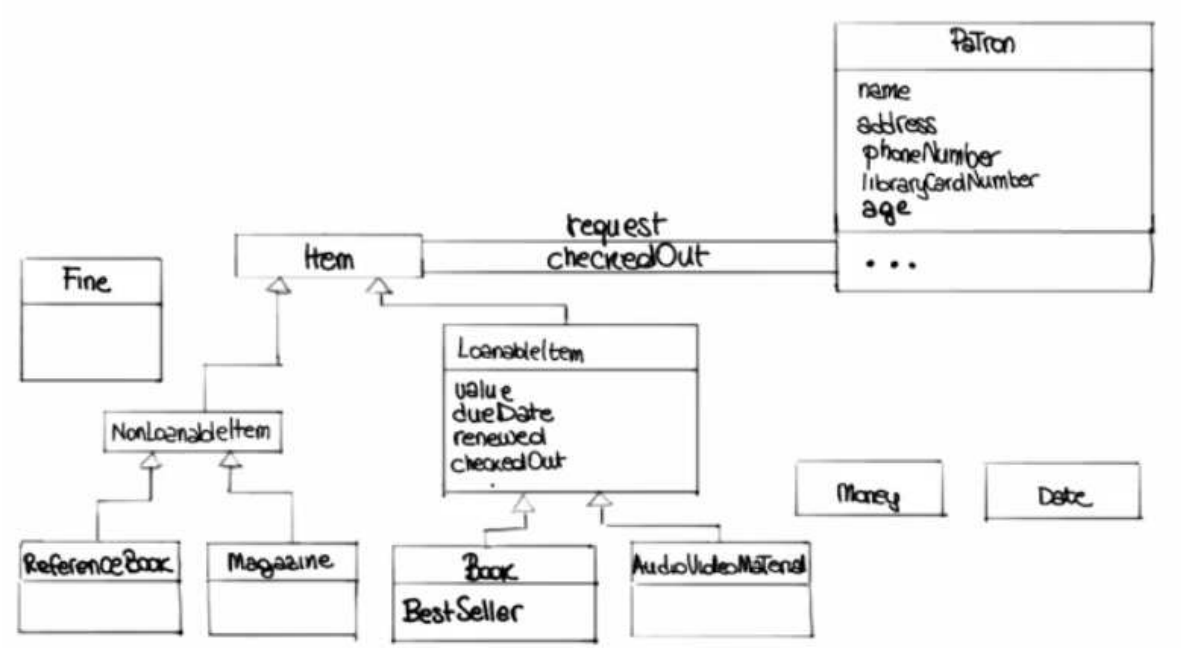
- It is a kind of item.
- Perfect, that's exactly what we're modeling next, which is what we call the Is-A relationship. So you said, a book is an item?
- A book is an item.
- And, we can model that in the diagram. So, we do that using another kind of relationship between the classes. So we're going to represent that as a specialization we call it. And, a specialization is indicated in this way. Okay? With this arrow at the end, so a solid line with this kind of arrow at the end. And we can do the same for book, magazine, reference book and audiovisual material.



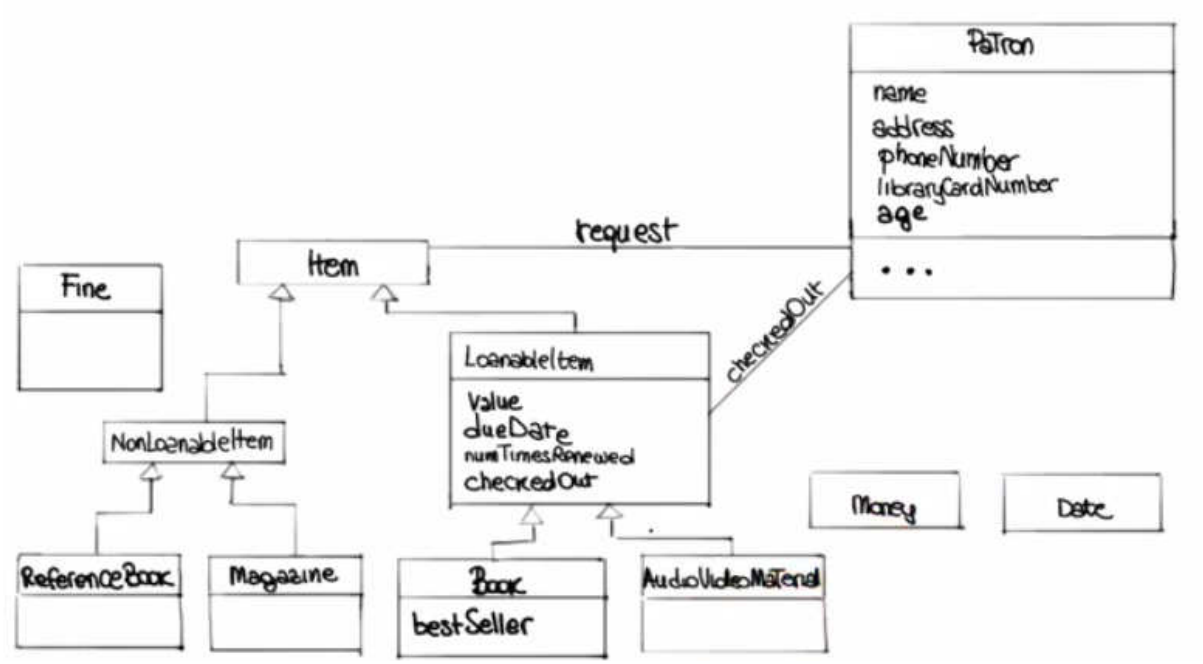
- So we're going to connect all of them to the item, using the same kind of connection. And now that we have connected all this with item and indicated them in subclasses, there's something else that we can do. So we can make this kind of cleaner. And I'll tell you what I mean by that. So now we have this loanable attribute that refers to item, but it seems to me from what you were saying before, that loanable is not really an attribute of an item. Right? It's more of a characteristic of the type of item.
- Right.
- Is that right?
- Right. Books, books and audio/visual are rentable but, but the others aren't.
- Okay, and so representing it here, it will work, but it's not really right. So from the style standpoint it's not the best way of modeling this. What we're going to do instead, we're going to use this specialization relationship to make that more explicit; to make it cleaner.
- Okay, so what I'm doing here is I'm going to take this hierarchy of classes, this is just on two levels now, and I'm going to kind of make it a little richer. So I'm going to add an

intermediate set of classes. And in particular I'm going to have these two classes that I'm going to call non loanable item and loanable item. So, they're both items but they tell me clearly that some items are loanable and some items are not. Okay?

- Okay.
- And then I'm simply going to put book and audio video material as subclasses of loanable items and reference book and magazine as subclasses of non-loanable item. So, if we look at this diagram now it's pretty clear what is loanable and what is not. And it actually is a much cleaner design.



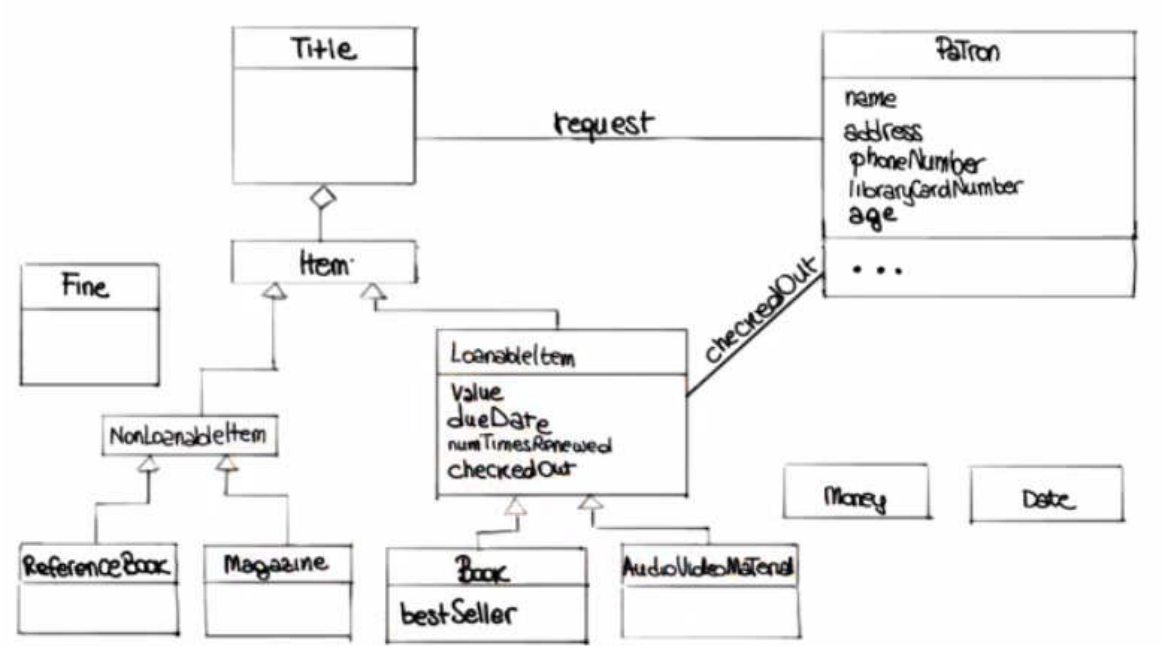
- And, and I see you've gotten rid of the loanable attribute, too.
- I did. Because at this point this is already represented by the fact of having these two classes. And actually, something else that I did is that I moved all these attributes, value, due date, renewed and checked out, that makes sense only for loanable item from item to loanable item. So at this point, this really is telling me that these characteristics are just meaningful for the loanable item, and not for the other ones.
- Well, speaking of that, the way that you got the lines going in the diagram here is you still have request and checked out going to item, even though you can't request non loanable Items.
- You can't check out non loanable Items. Oh, you were right actually. You got me on that one. You're exactly right. So this associations are between the two wrong classes. So, I guess, at this point, you can probably go and fix the diagram yourself.
- Well, can we just make the lines go from patron to loanable item instead of to item?
- That's exactly the way in which we are going to fix it. So, we're going to move these two associations down here. And at this point, this will represent the right relationships in the, in the diagram, and in the system.
- Makes sense to me.



Refining Relationships

- Spencer, I gotta tell you, I'm impressed. You're getting very good at this. So, why don't you go wild and continue. Is there anything else you think we can improve here?
- Well something was bothering me, that what happens if there's more than one book with the same title and somebody puts in a request?
- Oh, I see. That's a good point. So basically what you are telling me is there's kind of a difference between an item and the title, so the title is kind of a more general concept, in a sense. So if you can have multiple copies of a given title, is that right?
- Yeah, we have five copies of Tom Sawyer, and the person's, the patron's really putting in a request for any Tom Sawyer.
- They don't want like copy number three of Tom Sawyer, right? They want, they want to read Tom Sawyer. Okay and I can represent that. So, in which I suggest we do that, and you can tell me whether it makes sense to you is by introducing an additional class, which I call Title. And that represents exactly the concept that you're mentioning. So this is a title which represents some specific content that is not related to a specific physical element. It can be rated to multiple, physical elements. So basically I'm going to create this title.
- And then I'm going to create a relationship between the title and the item. And what the relationship is telling me is the association between these two in this case is an association that we call aggregation. So it's a special kind of association, that basically indicates that an item of this type, so a title, can consist of multiple elements of this type of multiple items. So it's telling me that one title can consist of multiple items, and I'm going to indicate it with this annotation, which is a this diamond at the top of the association.

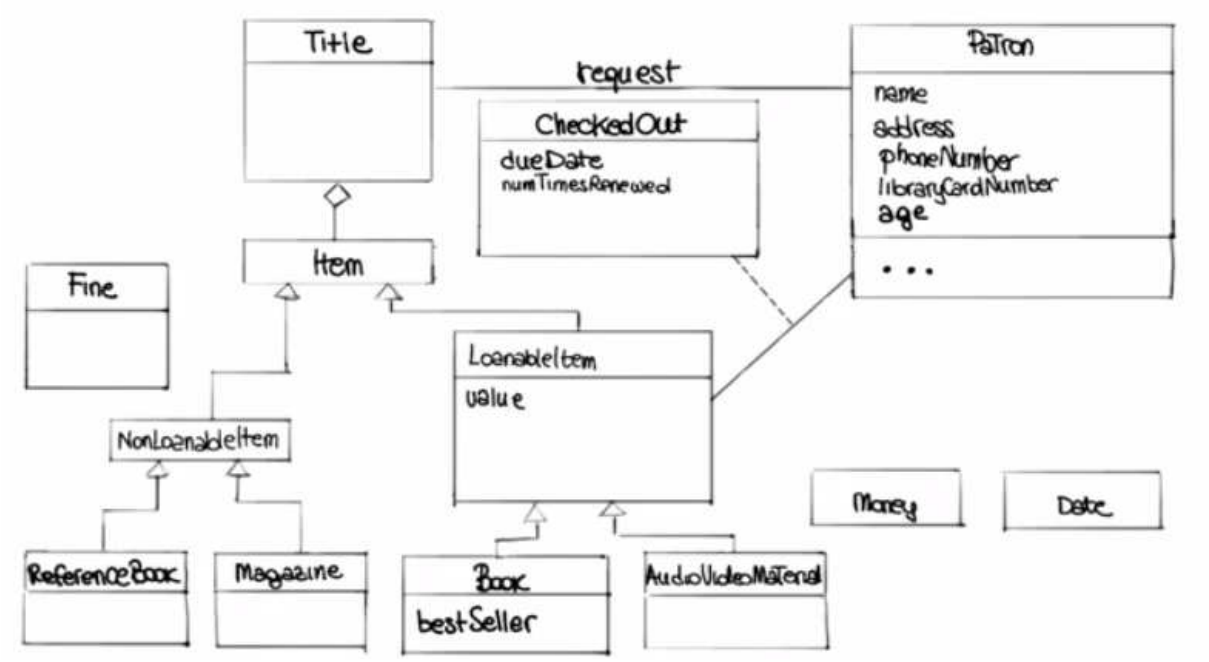
- And so, and so we can, we can move our request line, up from loanable item to title, because that's what their really requesting.
- Definitely and in fact that represents exactly the situation that you are mentioning, at this point when the patron makes a request. It makes a request to a title and not to a loanable item. And then, when the actual loan will take place, then that will be connected to a specific item.
- Right. Okay that makes sense.
- Makes sense?
- Yeah.
- Okay, good.



Refining the Class Diagram

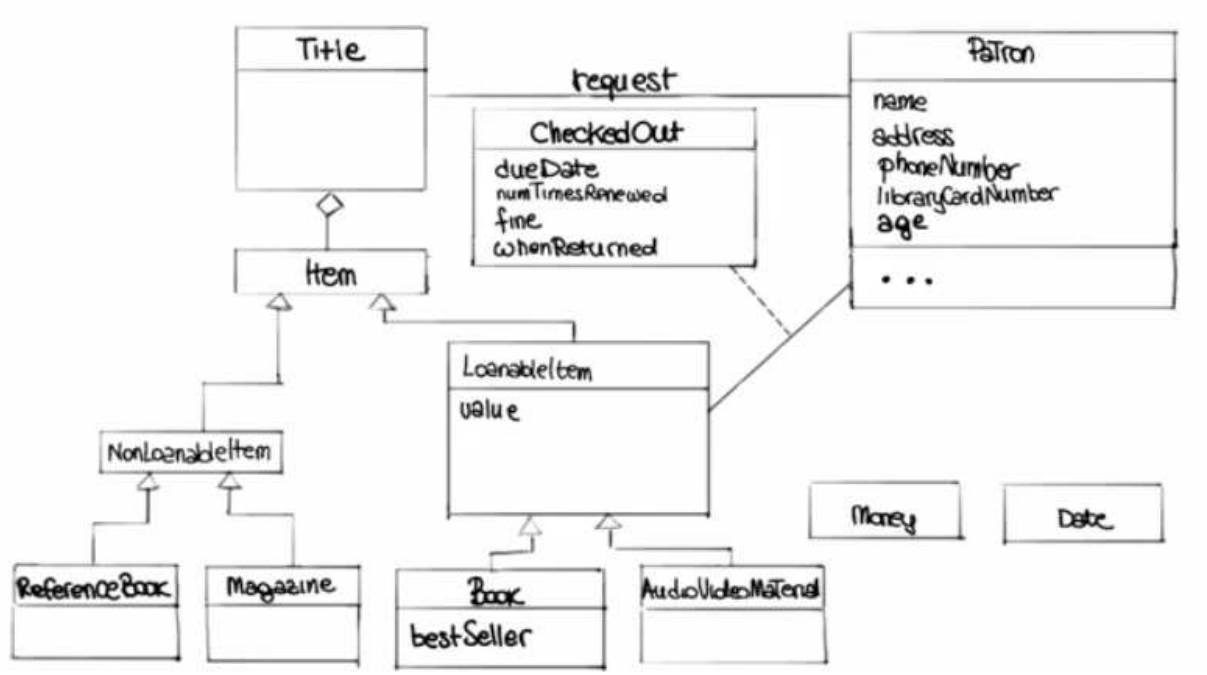
- Okay, so let me see if anything changed after we did this last modification. Actually, there is something that I would like to do here. Cause looking at this a little bit more, I noticed that there are two attributes, renewed and due date. That we have in loanable Item, right? But they don't seem to be really, attributes or characteristics of loanable Item. They, they're more characteristics of the association between the loanable Item and the patron. Would, wouldn't you agree?
- Well, yeah it, it's not like you could only renew a book once in its entire history.
- Right. Exactly. So, that's, that's why what I would like to do is I would like to move those out of loadable item. And actually there is a construct that we can use to express this. It's called, we haven't seen it yet, but **it's a special kind of class. It's called an association class.** So, it's a class that is connected to a specific association. So what we can do here, we can create this class, which I'm going to call checked out. I'm going to,

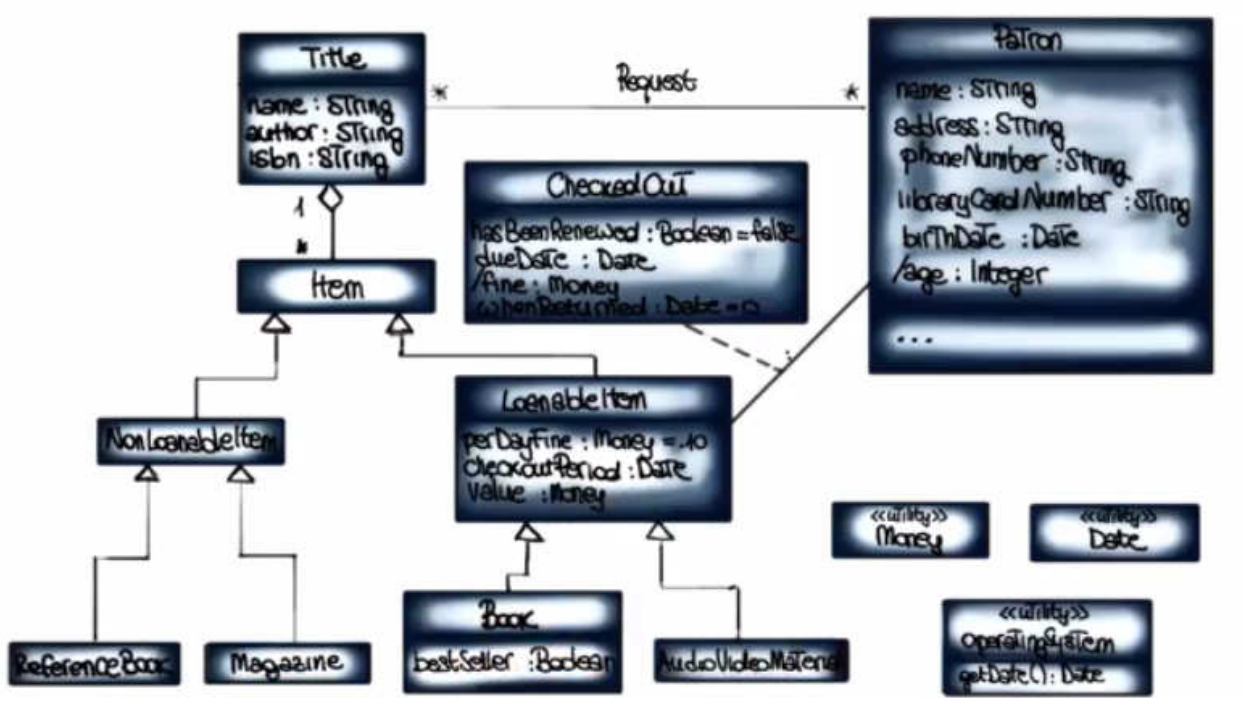
associate it with this, association. I'm going to connect it with this association. And then I'm going to move the due date and the renewed attributes From the LoanableItem here in this checked out class. So in this way, seems to me that it makes it very explicit for somebody looking at this class diagram, that these characteristics are characteristics of the loan, and not of the elements involved in the loan.



- Can you do the same thing with Fine? isn't Fine a property of the loan?
- Yeah, that actually is right because a fine is a fine for a specific loan, right?
- That's correct.
- Okay, so yeah. Then we can do that. We don't need to represent fine as a class, we can just transform that into an attribute that we can put into the checked out association class.
- Gotcha.
- Anything else?
- Yeah. It occurred to me that there's another thing that happens. In one of my scenarios, I put down about the patron actually returning an item.
- Right. Okay, so we would probably need an additional operation, I guess, for the patron.
- Right.
- So, okay, so what I'm going to do, is pretty easy to do, I'm just going to add the return operation here in the patron, and when that happens, that will mean that I'll get rid of this association class because the item is returned. Is that right?
- Well, what happens if somebody drops the book in the book drop if they were due and doesn't pay the fine? Will that get rid of the information about what they owe?
- Oh, I see. So you can have the item available, but you still want to know if there are any pending fines on the book.
- Uh-huh, and how much those fines are.

- And how do you compute how much it is?
- It's how many days it was, from the time it was due, to when they returned it.
- I see. OK. So you know what we can do? I think we can put an additional attribute in the checked out class and I'm going to call it when returned and that item will have either a special value or it will contain the date in which the book was returned. So in this way you should be able to keep this in the system until it's paid, and also to compute how much the fine is. Is that, is that working?
- So the special value is for a normal situation when they haven't, they don't owe anything and haven't returned it yet.
- Exactly so that will tell us that, that, that the loan is still active basically.
- Right.
- Does that work for you?
- Yes.
- And you know, I like this. I mean, I feel pretty good about it. I think we have a nice class diagram. So what I'd like to do is just go off and clean it up a little bit, and put it in in IDE so I can print it and rearrange things a little bit. And then I'd like to sit down again and just go through it for a last time. And for some final consideration. So if you don't mind we will take a ten minute break and reconvene here.
- That's fine.
- Alright.





Final Considerations

- Okay. So this is what I've done. As you see, it looks a little nicer than it was before. And I didn't really change that much. I just made a few changes, so I just wanted to point them out to you, so that you know what they are. And one of the main things I did is really to introduce these **derived attributes. So these are attributes that are basically computed based on some other attributes. Okay, they don't have a value themselves, but their value is computed.** And I used two. The first one is age. So basically we know the age of the patron based on the birthday, of the patron. So you guys, I don't know if you have that information currently in the system.
- No, we'll have to add that to the form patrons fill out, when they get their card.
- Is that an issue? Can you do it?
- No. Yes, we can easily do that.
- Okay, so then, perfect. So we'll do it that way. I think it's a little cleaner. And similarly, since you told me that the fine was computed based on the amount of days that an item was late when the patron was late returning the item. So I also added this as a derived attribute that is computed based on the due date and when the item is actually returned. Makes sense?
- Makes sense.
- Okay. The rest is kind of really minor things. So the only thing I want to point out is I I added this, which is called **cardinality**, for some of these relationships. And what they say is basically is **how many elements are involved in the relationship**.
- So, you mean the stars?
- Yeah, like the stars and the one...
- Okay.

- Here for example, this is telling you that for each item there is only one title. And that for each title, there are multiple items.
- So, star means many.
- Stars means many, yeah.
- Okay, go you.
- Sorry that's kind of computer science lingo - we use the star for that kind of stuff. And, similarly, for the patron, it's telling me that, you know, each patron can request multiple titles, and that the same title can be requested by multiple patrons, which I think is the way the system works.
- Right.
- So except for these minor changes we already had a pretty good model in our hands, so I think we can finalize this and then just move to the low level design and then implementation, and be done with the system.
- Sounds good.

Debriefing

- So Spencer, now that we went through this process and, I'd just like to hear whether you enjoyed it, whether you think it was useful. What are your thoughts?
- Well, it was very interesting. I not only learned something about computers and about how you design information systems in UML, but I also learned something interesting about the library. And things that I knew but I never really, explicitly wrote down...
- Uh-huh.
- ...Came up during the course of doing this. And I think I now much better understand what this information system that you're going to build for us, is really all about.
- Okay, well, I mean, I'm very happy that you say that, because I really believe that, you know, doing **this kind of analogies and design exercises really helps you figuring out whether there's any issues with the requirements.** So for example, **you can find out whether there's any missing information, or maybe conflicting information**. And I think that's exactly what happened today. So I'm very glad to hear that it worked for you. That you enjoyed it.
- [To audience] I hope you enjoyed it as well. And I strongly encourage you to do this kind of exercises for different kinds of systems. So as you can become more familiar with analysis and design techniques.
- So, any final thoughts?
- I look forward to receiving your delivered software.
- All right. Will do.