Exercises up to: Chapter 10

1. Write a `Point` class that represents a point in X,Y space (i.e. 2D space). It should have two integer parameters representing it's X-Y coordinates. Think of a sensible constructor and methods (look at the example of how you might want to call a contractor to make a point object in 2. below).
2. Write a `Box` class that represents a rectangular box in X,Y space. The `Box` class should use two `Point` objects to define opposite corners. Again think of sensible constructors and methods, and include `height()` and `width()` methods that return how high and wide the box is. You might use these two classes like this:

```
Point topRight = new Point(5,3);
Point lowerLeft = new Point(1,1);
Box box = new Box(topRight, lowerLeft);
```

   which would be a box that is 4 wide (5 - 1), 2 high (3 - 1), with its lower left corner at X-Y coordinates 1,1 and it's top right corner at X-Y coordinates 5,3.

3. **Quite Tricky:** Add an method to Box called overlaps:

```
public boolean overlaps(Box box) {…
```

4. which checks if one box overlaps another. You can find discussion about the logic you need to check this here:

   http://gamedev.stackexchange.com/questions/586/what-is-the-fastest-way-to-work-out-2d-bounding-box-intersection

5. **Aside:** This kind of test is often the 'first pass check' to see if two objects are colliding in game logic.
6. **Challenge** - Repeat all this using 3D
7. Write a class that represents a `Person` - it should have a given name and a family name, both Strings. Include sensible constructors and methods.
8. **Challenge** - extend your `Person` class to include a date of birth using a Java `Date` object.
9. Write a class that represents a TV show - it should have a title, a short description, and a leading actor (I'm using 'actor' in the modern sense to mean female or male). The title and description should be strings but the leading actor should be a `Person` - using your `Person` class from the previous exercise. Again, include sensible constructors and methods.
10. There is usually more than one leading actor. Create a `PersonList` class that allows you to create lists of `Person` objects. Whenever I say something like 'includes a list' or 'contains a list' I mean 'use an `ArrayList`' - in this case store the actual `Person` data in an `ArrayList` - i.e. your code should have a line that looks like:

```
ArrayList<Person> personList = new ArrayList<>();
```

11. Modify your TV show class so that instead of a single leading actor there can be a list of leading actors - do not try to incorporate this directly into your TV show class but instead modify it so that it uses your `PersonList` class instead of `Person`. For example, you might create a TV show like this:

```
PersonList actorList = new PersonList();
actorList.add(new Person("Keri", "Russell"));
actorList.add(new Person("Matthew", "Rees"));
TvShow theAmericans = new TvShow("The Americans", "A show about " +
    "the cold War that resonates with people who are about " +
    "Neal Harman's age and has a leading actor who is Welsh",
     actorList);
```

12. Write a class called `ScheduleItem` that contains a `TvShow` object and a time at which it will be broadcast. You can just use two integers for the time - one for the hour and one for minutes (make sure they have values in the correct range).
13. **Challenge** - use Java `Date` objects instead of integers for the time.

14. Write a `ChannelSchedule` class that includes a list (remember, use ArrayList) of `ScheduleItem` objects - it should be possible to add (and remove?) ScheduleItem objects to a `ChannelSchedule`. The idea is this represents the shows that are on in a particular day.
15. Extend your `ChannelSchedule` class with a method that prints out the schedule in a 'nice' way - this is easiest if you have included sensible `toString()` methods in all the other classes you've written so far - so go back and put them in if you haven't:-) *Don't* try to do all the work in `ChannelSchedule`.