

Tutorial 7

11/15/2023

100/100 Points

Attempt 1



Review Feedback

Offline Score:
100/100



View feedback

Unlimited Attempts Allowed

11/15/2023

Details

Part 1 - Fun

Activity 1

Activity 2

Activity 3

Activity 4

Activity 5

Activity 6 (C)

Part 2 - A ro

student

roster

Activity 7


Activity 8

Getting Cre

IF YOU ARE

IF YOU ARE

Note: Unfortunately, there's no advanced tutorial this week. If you have an idea for one (related to subtyping or inheritance...let me know! I tried a couple of variations of adding this sort of thing to our interpreter...but they were way too hard to complete in one 50 minute period...).

Speaking of which... If you were working on the adv tutorial last week and didn't finish...why not work on that now? [Advanced Tutorial 6](https://bain-cs111.github.io/assignments/adv-tutorial-6)  (<https://bain-cs111.github.io/assignments/adv-tutorial-6>) . It'd probably be more fun. But who am I to say what's fun. Especially since I'm having a conversation with myself. In a blockquote...that maybe no one will read. Oh well. Who knows. Come find me if you're bored, we can find something for you to do!

Our goals for today are pretty simple:

1. Understand why subtyping is useful
2. Practice implementing subtypes in Racket
3. Understand why methods are useful
4. Practice implementing methods in Racket

In this tutorial, you'll deal with these two concepts separately but in this week's exercise, you'll have to deal with them at the same time while designing interactive quizzes.



(<https://canvas.northwestern.edu/courses/201068/modules/items/2851496>)

(<https://canvas.northwestern.edu/courses/201068/modules/items/2851496>)

Part 1 - Fun at the zoo

Today we'll be defining a whole zoo's worth of animals. However, as we saw on Monday, we want to use *subtyping* to make our object design more efficient.

Activity 1

First define a *abstract* struct called `animal` that has the following fields:

- `name`
- `age`
- `weight`

Note: if you're still struggling with structs, I HIGHLY recommend listing out all of the automatically defined functions you get as comments in your program like I did on Monday and in the demo video.

Activity 2

Now define a subtype of `animal` called `cat` which has an additional field:

- `sleeping-spot`
-

Activity 3

Now define a subtype of `cat` called `cat` that has an additional field:



Activity 4

Now define a subtype of `animal` called `mouse` that has an additional field:

- `hiding-spot`
-

Activity 5

Now write a function called `feed-animal!` that takes as input a single `animal` and **mutates** its weight by adding 2 to it.

Make sure to write at least a couple of tests to make sure its working the way you think!

Hint: Remember, that now that we have imperatives...writing tests is a little confusing. You need to take into account what the weight WAS in order to find out what the weight will be AFTER calling the function.

Activity 6 (Challenge)

If you're in attendance and it's already the 25 minute mark, move on to Part 2. Come back to this question if you have time.

Write a function called `feed-animals-with-favortism!` that takes in a list of animals (`loa` for short) and uses `feed-animal!` *for each* animal in the list with one big caveat...

Bob is gizmo's best friend. He is also the zoo keeper! Bob wants to feed all of the animals in the zoo, but he will feed any animal twice who lists him as their `best-friend`.

We've included some tests for you on this one.



In the Racket file you downloaded scroll down until you see “PART 2”. Here you’ll find two different struct definitions.

`student`

A `student` has several fields:

Name	Type	Description
name	<code>string</code>	the student’s name
grad-year	<code>number</code>	what year the student will graduate in
major	<code>string</code>	the student’s major

This struct *does not* have any methods.

`roster`

This struct represents the roster of a course. Here are its fields:

Name	Type	Description
course-name	<code>string</code>	the course’s name
instructor	<code>string</code>	the name of the instructor
students	<code>(listof student)</code>	the list of <code>student</code> s in the class

It also has a method:

Name	Inputs	Description
<code>display</code>	takes as input ONLY a <code>roster</code>	Displays the roster in the interactions window, class name, instructor name, and then each of the students in the class.

Activity 7



Name	Inputs	Description
<code>search-by-major</code>	takes a <code>roster</code> and a <code>string</code> as input	searches through the <code>roster</code> , returning the list of students with the provided major

Activity 8

Next (and last) implement one final method:

Name	Inputs	Description
<code>names-by-grad-year</code>	takes a <code>roster</code> and a <code>number</code> as input	searches through the <code>roster</code> , returning the list of students' names with the provided grad-year

Getting Credit for Your Work

IF YOU ARE IN CLASS OR AT ALT TUTORIAL

Find one other person in your group that is finished and peer review each other's work. Here are the things to check:

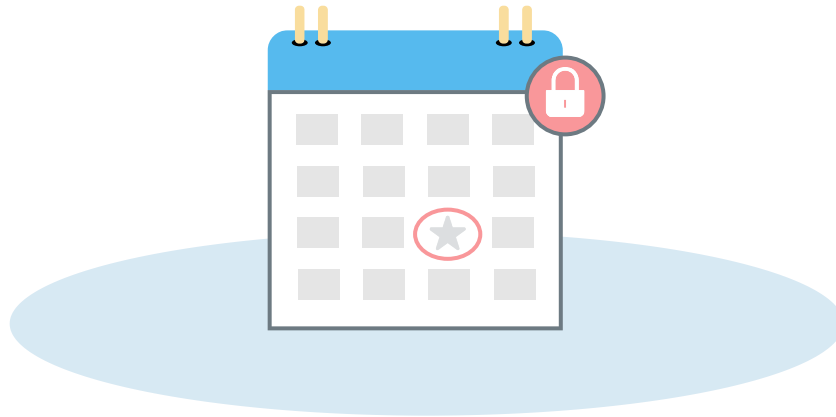
1. Does their code look readable and neat?
2. Can you understand what their code does by reading it?
3. How was their solution different from yours (if at all)?

Once you've each taken a look, take a second to debrief. Anything either of you found difficult? Easy? Fun? Mind-blowing? Once you've debriefed, **both of you should fill out this [attendance Google Form](https://forms.gle/vhKxTCXHJUtmOD7v8)** (<https://forms.gle/vhKxTCXHJUtmOD7v8>). **NOTE: You will need the NetID of the person's whose code you reviewed.** You do not need to submit your Tutorial RKT file unless you want to.

IF YOU ARE SUBMITTING REMOTELY



Before turning your assignment in, **run the file one last time** to make sure that it runs properly and doesn't generate any exceptions, and all the tests pass. You might also take a look at the Autograder/Submission FAQs on Canvas under the [Class Resources](#) page.



Availability dates

11/15/2023

