Creatures!

What is this challenge for?

We want to understand the way you think about problems, and how you write code to tackle them. We're not looking for the most efficient algorithm or the purest design, we're looking for the simplest solution to the problems in front of you.

We're not going to give much in the way of guidance as to the specifics of implementation. If, for example, you think a class needs an attribute or a method, you go ahead and do it. You're in charge.

What do you need to prepare?

You'll share your screen with Google Meet and code in any IDE of your choice. Please test your mic, camera and screen sharing setup before the call. If you have chosen to code live, we recommend that you have a working environment ready (for example Typescript and Jest running)

How should solutions be presented?

Ideally, solutions will be written in a typed language with some basic tests. That being said, you should use the language with which you feel most fluent. We've seen pretty much every language so far so don't be afraid to use what you know!

So, what's the challenge?

Imagine you're a collector and all around you there's a world with Creatures that you can catch and collect.

Your task is to write a program that allows a Collector (like you) to interact with these Creatures. The challenge is broken down into four parts.

Part 1

Define some domain classes to model this application.

There are many different species of Creature, each belonging to a family (e.g. flyer, swimmer, runner) and a species (e.g. Bird, Shark, Lion). Feel free to invent your own test-cases!

Both Collectors and Creatures need to have a Position, so we can know how close different Entities are to each other. You may need some concept of a World or Map to hold references to the Entities within it.

Part 2

Gotta collect 'em all!

In this part we'll implement the means for a Collector to catch a Creature and add it to their collection.

First, we need to know which Creatures are nearby to a Collector. Write code to calculate this. How you define "nearby" is completely up to you.

A Collector can catch a random *nearby* Creature and add them to his/her collection. Write code to allow this.

At the end of this part of the interview, we'll ask an extra coding question to implement live a new function for the game.

Part 3

Multi-player

This part is a theoretical question on the topic of distributed systems, so we don't anticipate code to be written. Instead, write down your thoughts on the problem and the questions raised.

We're going to introduce the idea that multiple Collectors can try and catch Creature *at the same time*.

Scenario:

1. I'm sat at my computer and I see a Creature I want to catch. I try to catch it.

2. At the exact same time, my neighbour sees the same Creature and also tries to catch it.

3. My neighbour succeeds and adds the Creature to his collection before I've finished catching it.

4. I am unable to catch the Creature because it has already been caught.

5. I now feel sad.

How can we avoid this situation? Does your program handle this elegantly? If not, how can you change it to allow a Creature to be caught by multiple people? What are the ramifications of allowing that?

It may help to imagine that instead of just two people, it's 10,000 people cramming into Central Park to catch the same Creature - would the same solution work then? What are the limitations? How could you mitigate this?