Entity (Model):

* AWS Reference: Relational Database Service (RDS)
* Definition: Represent persistent information tracked by the system

Boundary:

* AWS Reference: S3 storage bucket
* Represent interactions between actors and the system

Controller:

* AWS Reference: Lambda functions
* Realize use cases

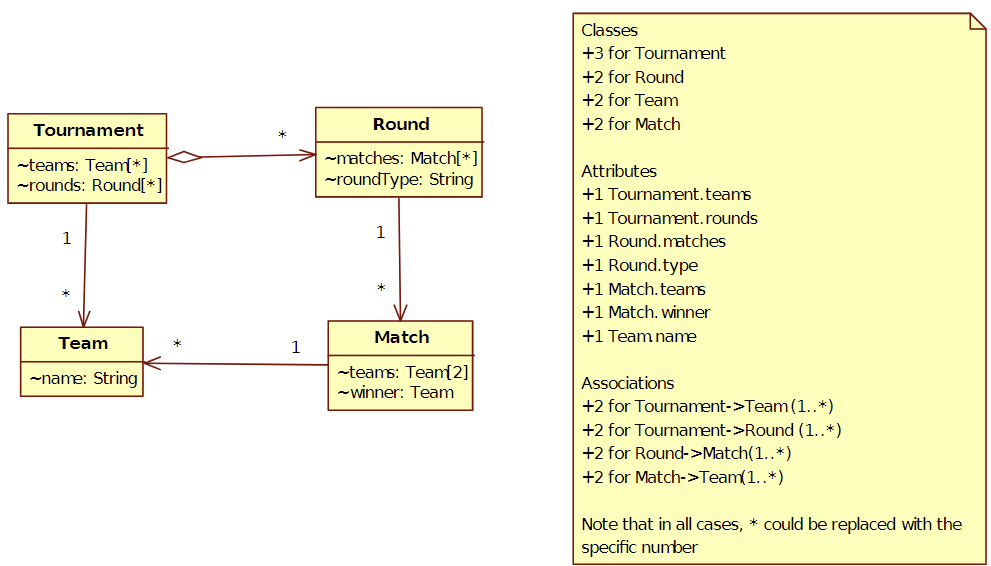
AWS services definitions:

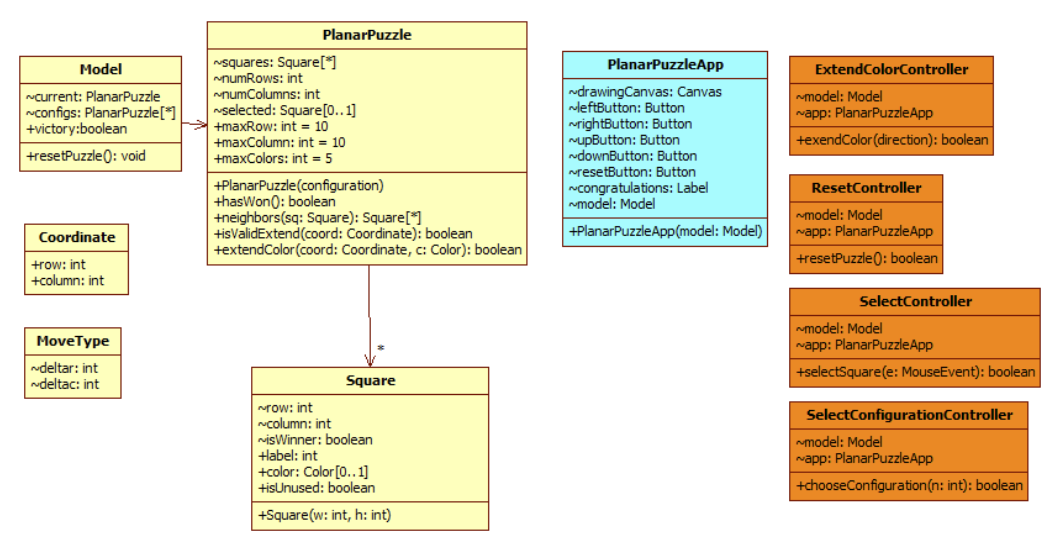
<https://docs.aws.amazon.com/index.html>

1. Lambda: is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. Amazon Web Services (AWS) Lambda is a serverless compute service offered by AWS. It allows you to run code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table. With AWS Lambda, you can write and run code without having to worry about managing the underlying infrastructure. AWS Lambda automatically scales your code to run in parallel and processes events as they occur, so you only pay for the compute time that you actually use. AWS Lambda supports a wide range of programming languages, including Node.js, Java, Python, and C#. You can write your code in the language of your choice and upload it to AWS Lambda as a function. AWS Lambda then runs the code in response to the specified events, and you can use the AWS Lambda console, the AWS SDKs, or the AWS CLI to manage your functions and monitor their performance.Overall, AWS Lambda is a convenient and scalable way to run code in response to events without having to manage any underlying infrastructure. It allows you to focus on writing and running code, while AWS takes care of the rest. The structure of the input to a JavaScript AWS Lambda function depends on the event source that triggers the function. Different event sources have different input formats, but in general, the input to a Lambda function is a JavaScript object that contains the event data and metadata, usually as an HTTP request. This input object contains information about the S3 event that triggered the function, such as the name of the bucket and the key of the object that was created. It also contains metadata about the event, such as the event time, the AWS region where the event occurred, and the identity of the user who triggered the event, containing the data that must be used to perform the specified function of the lambda, like retrieving or modifying data, and sending it back as a response, or responding with a 400-error code.
2. API-Gateway: is a API management tool that sits between a client and a collection of backend services. The API gateway acts as a reverse proxy to accept all application programming interface (API) calls, aggregate the various services required to fulfill them, and return the appropriate result.
3. Code Commit: AWS CodeCommit is a secure, highly scalable, managed source control service that makes it easier for teams to collaborate on code. Allows for hosting code for version control and an centralized place to store code.
4. Cloud9: a cloud-based integrated development environment, with a code editor, debugger and terminal. The editor allows for sharing the editing environment to allow for paired programming sessions.
5. RDS (Relational Database Service): An AWS Lambda function can retrieve data from an Amazon Relational Database Service (RDS) instance by using the RDSDataService client in the AWS SDK for JavaScript. In order to establish a proper connection, the endpoint URL, username,password and schema must be provided. The lambda functions source code specify the SQL queries to be made on the database depending on the input received from the API gateway.
6. EC2: Elastic Compute Cloud provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.
7. IAM (Identity and Access Management): is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
8. S3: Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements. To host a React application on Amazon S3, you first need to build the application using the npm run build command. This will create a production-ready build of your application in the build directory. Next, you need to create an S3 bucket where you will store your application files. You can do this using the AWS Management Console, the AWS CLI, or the S3 API. After you have created the S3 bucket, you can upload the files from the build directory to the bucket using the AWS Management Console, the AWS CLI, or the S3 API.Once the files are uploaded to the S3 bucket, you need to make the bucket contents publicly accessible. You can do this by setting the bucket's permissions to allow public read access to all of the objects in the bucket. Finally, you need to configure the S3 bucket as a static website. This will allow users to access your React application by visiting the S3 bucket's URL in a web browser. To do this, you can use the AWS Management Console, the AWS CLI, or the S3 API to enable static website hosting for the bucket and specify the name of the index document that will be served when users access the bucket's URL.Once you have completed these steps, your React application will be hosted on Amazon S3 and available to users at the S3 bucket's URL. You can use the AWS Management Console, the AWS CLI, or the S3 API to manage your application files and monitor the performance of your application.

UML Diagram Example:

Only need to show one to one or one to many





Q1 [22 points]. Imagine adding the following functionality to the ****SlidingPuzzleApp****, the case study completed in the Individual Project module.

*Upon solving the puzzle, a proud user can print out the sequence of moves that led to the solution*

Use Case: Print Solution Sequence

* Participating Actor: Initiated by User
* Entry Condition
  + User has solved the puzzle
* Exit Criteria
  + None or Result Printed
* Flow of Events

1. User requests to print solutions sequence

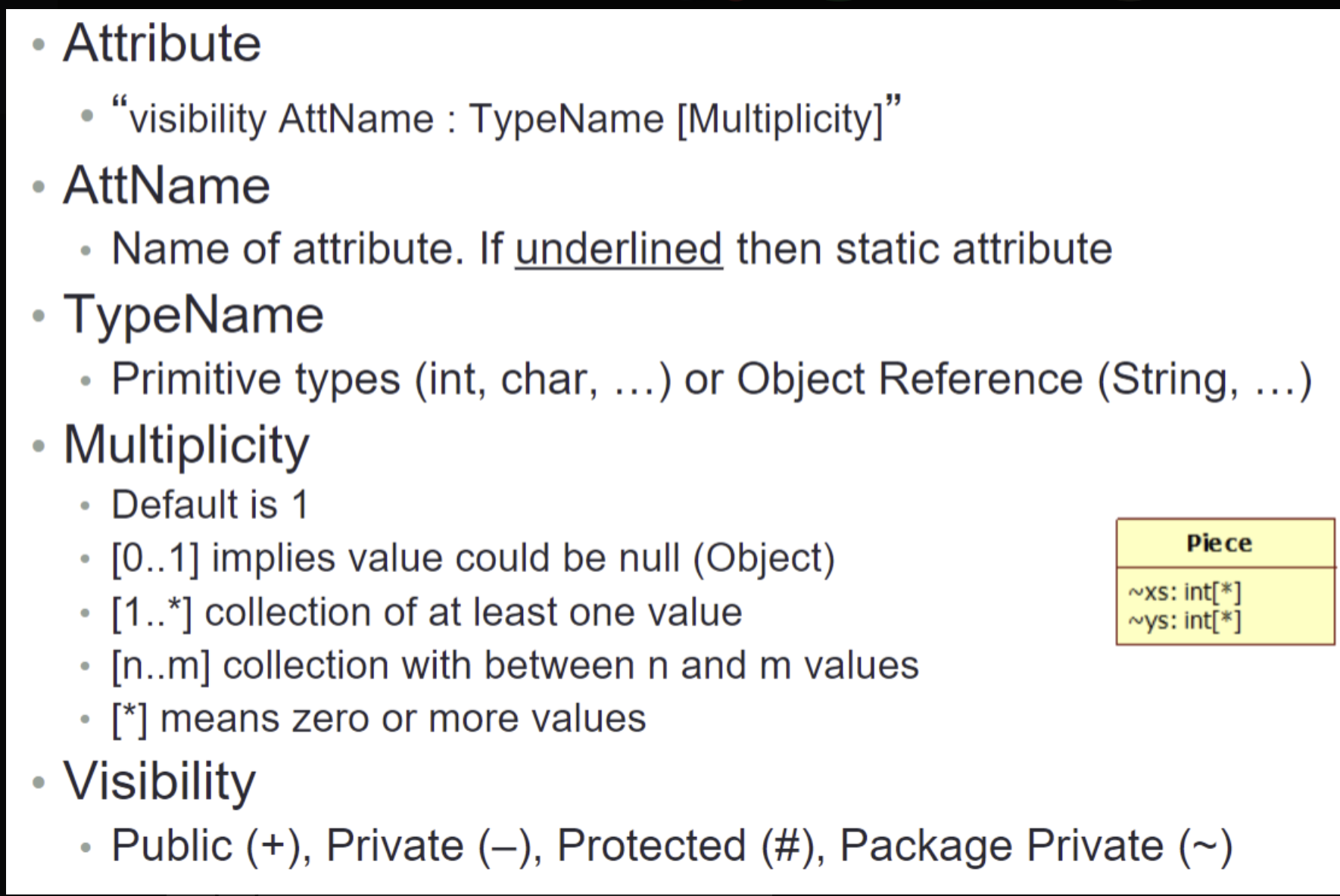
2. SlidingPuzzleApp prints the solution to the console and refreshes display

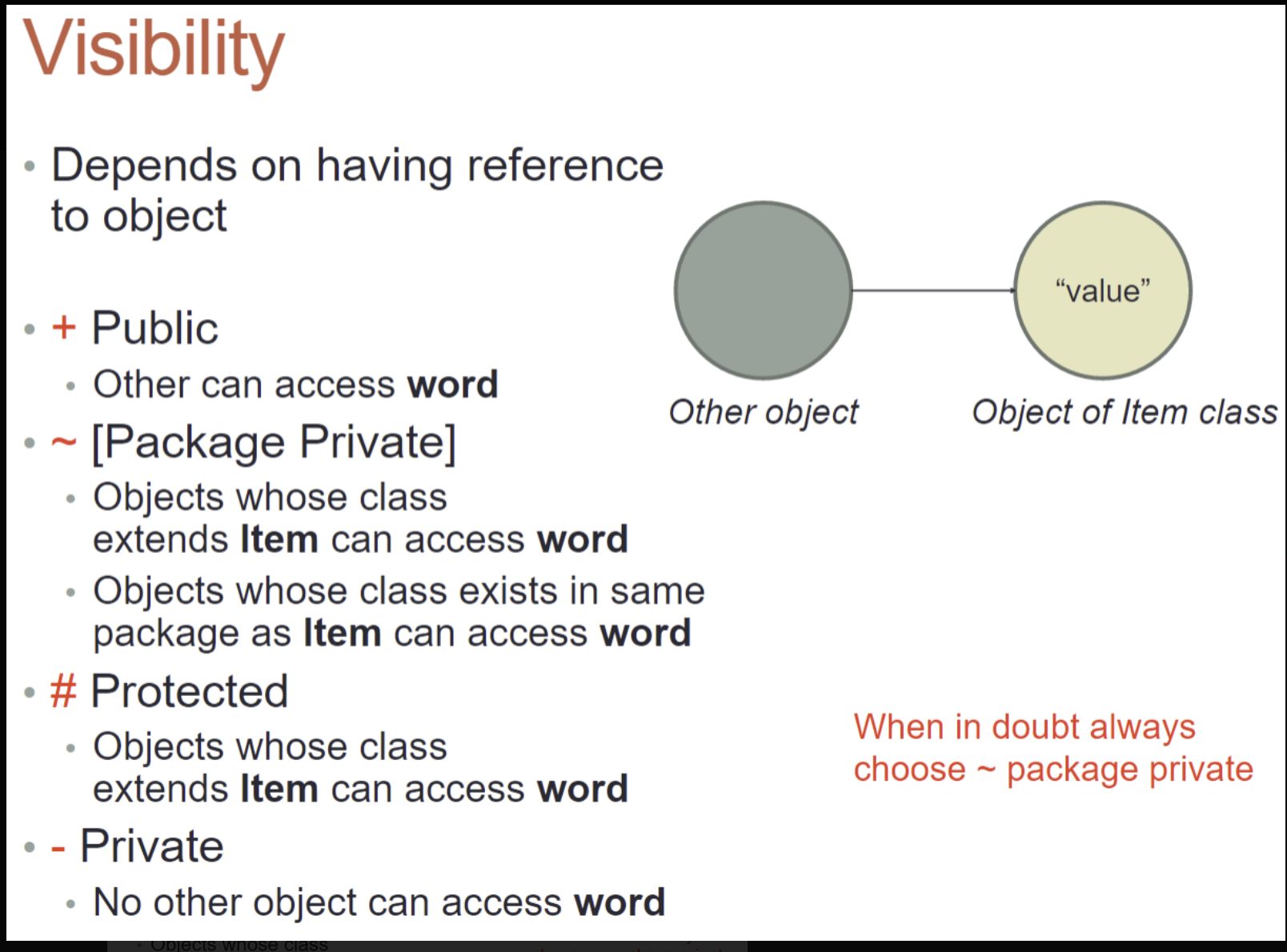
Q2. Given the following application domain, design a UML class diagram (including attributes and associations) for the following context shown by example:

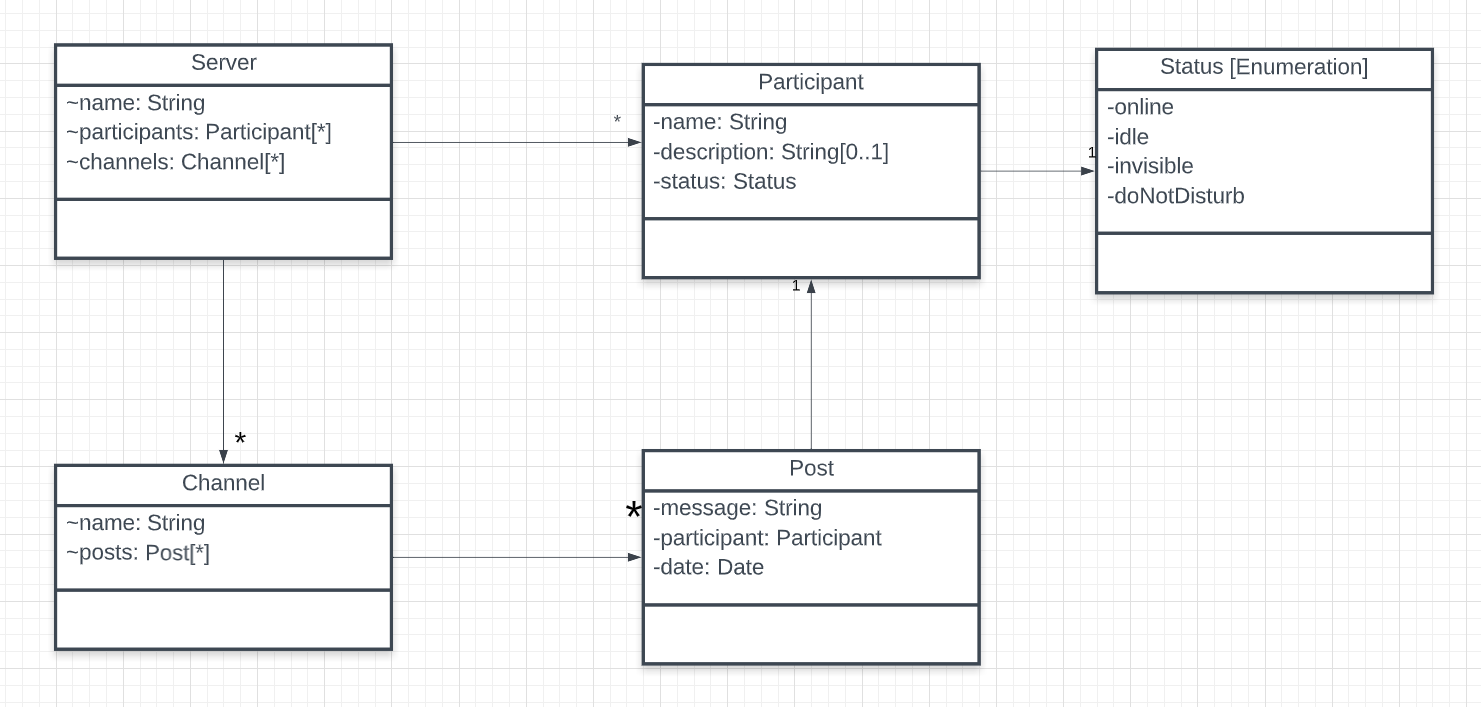
*A Discord server has a name and a number of participants. Each participant has a name, an optional string description and a status (either Online, Idle, Do Not Disturb or Invisible). The server contains a number of named channels. A channel keeps track of a sequence of posts. Each post contains a text message by a participant at a given date.*

*Note: You can assume there is a class Date that records dates like “11/3/2022 9:30 PM”*

**You must show attributes and relationships between classes. Do not show methods or constructors.**







Q3. Given the following problem, show the UML class diagram for the appropriate React **entity**, **boundary** and **controller** classes that you would use. This application has two use cases:

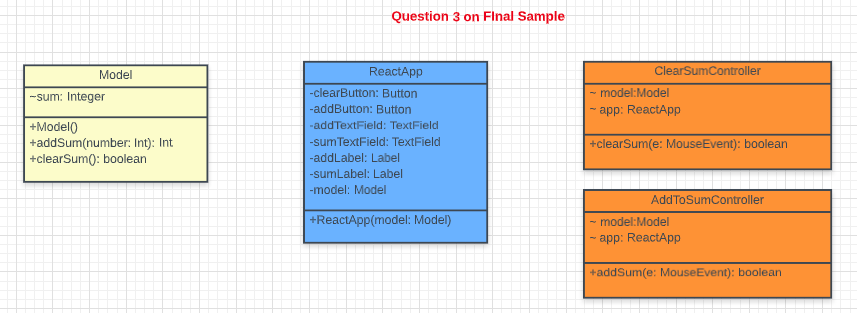
* Clear Sum
* Add to Sum

There is no need to show relationships (i.e., arrows) between the classes.

Shape

Description automatically generated with medium confidence

Use proper UML class diagrams to capture the attributes and methods



Q4. Party API design

There should be three resource entry points. Note names are subjective and should align with the use case names.

-----------------------------------------

## Create PartyList

/create

POST { "item" : [ { "item" : "large pizza" }, {"item" : "folding table"}} ] }

200 SUCCESS:

PARTY-LIST

400 FAILURE:

{ "error" : "No items in list” }

-----------------------------------------

## Show PartyList

/show

GET // no need for payload

200:

PARTY-LIST

400:

{ "error" : "No List" }

-----------------------------------------

## Claim Item

/claim

POST  
 { "item" : "1. Two Large Cheese pizza", "name" : "George Heineman" }

200 SUCCESS:

PARTY-LIST

400 FAILURE:

{ "error" : "No Such Item" }

{ "error" : "Item Already Claimed" }

-----------------------------------------

## Payload Definitions

* PARTY-LIST  
    
  { "items": [

{ "item" : "1. Two Large Cheese pizzas" },

{ "item" : "2. Two large Veggie pizzas", "person" : "Taylor Swift"}

...

]  
}

* Others as you need…