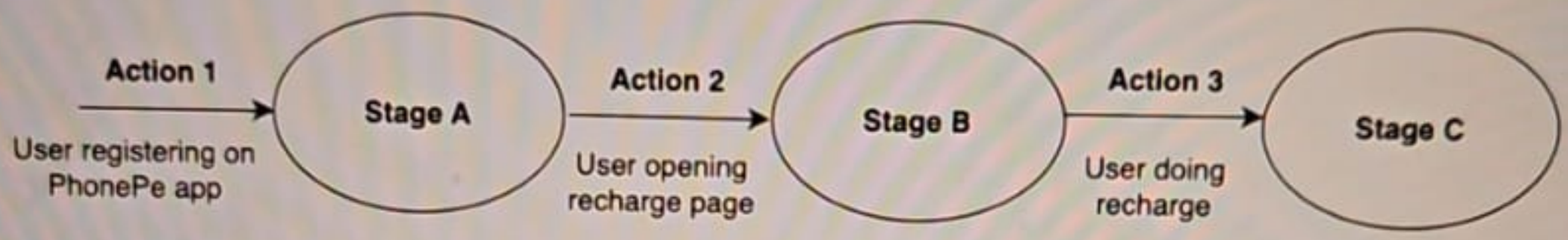
Design and implement a journey framework to track users across multiple use-cases. This will be a rule-based journey framework where a user will onboard a journey and move across it on the basis of certain conditions.

Requirements:

Journey: A journey is a Directed Acyclic Graph (DAG) that represents a predefined path to track a user on. A Journey will have multiple stages and corresponding actions to be performed in order to move the user from one stage to another. Multiple users can onboard to the same journey. A user can onboard a journey only once.
There will be complete isolation across different journeys i.e. anything in one journey won't depend on any other journey.



Journeys can be of 2 types, time-bound (having a start-date, end-date) and perpetual (no validity and will remain live till it's in active state).

- First stage in the journey will be the onboarding stage, middle stages will be the onward stages and the final stage will be the terminal stage.
- A journey will initially be in CREATED state and you will need to create a state machine of allowed transitions for a journey.
- Also when the validity period of a journey is over, mark it expired for time-bound journeys.

Mandatory Implementations:

1. **createJourney(Journey journey)**

This will help in creating a journey in the system.

2. **updateState(String JourneyId, JourneyState)**

Update state of the journey.

Mandatory Implementations:

1. **createJourney(Journey journey)**

This will help in creating a journey in the system.

2. **updateState(String JourneyId, JourneyState)**

Update state of the journey.

3. **getJourney(String journeyId)**

To get the journey details

4. **evaluate(String userId, Payload payload)**

Input payload can contain some fixed params and a Map of <key, value>
Basis the payload, 3 things may happen:

1. The payload may match the condition for onboarding the user to a journey. In this case, the user will get onboarded to the journey.
2. The payload may match the condition required for moving to the next stage for a user already onboarded on a specific stage. In this case, the user will move to the next stage in that journey.
3. The payload will not match any condition, in this case it will be simply ignored.

Users onboarding/transitions in a journey will happen only when the journey is in ACTIVE state.

5. **getCurrentStage(String userId, String journeyId)**

Find the current stage of a user in the given journey id.

6. **isOnboarded(String userId, String journeyId)**

Return true/false basis on the condition that whether the user is part of the journey or not.

Optional 1:

Provide a way where we can send SMS to a user as soon as any stage transition happens for that user (onboarding to the journey or moving from one stage to another). It should have the capability to decide for which stage we want to provide this feature.

Optional 2:

Add support for recurring journey, where a single user can onboard the journey multiple times.

Points to note

- Your code should cover all the mandatory functionalities explained above.
- Your code should be executable and clean.
- All necessary validations must be present.
- In case of an exception, proper errorCode must be present.
- Store the data in-memory for journeys and the user-transitions within the journey.

How will you be evaluated?

- Code should be working.
- Code readability and testability
- Separation Of Concerns
- Object-Oriented concepts.
- Language proficiency.
- SOLID principles

Expectations(Interviewer):

- Code should be working properly.
- All validations are present or not. For journey creation e.g. for time-bound journey start-date should be less than end-date.
- Inheritance is present or not.
  There are 2 places where inheritance can be present, at Journey level and at stage level (onboarding stage and onward stage)
- Should maintain the state-machine for allowed state transitions for a journey.
- Using proper code syntaxes or not.
- More focus on the code style and implementation.