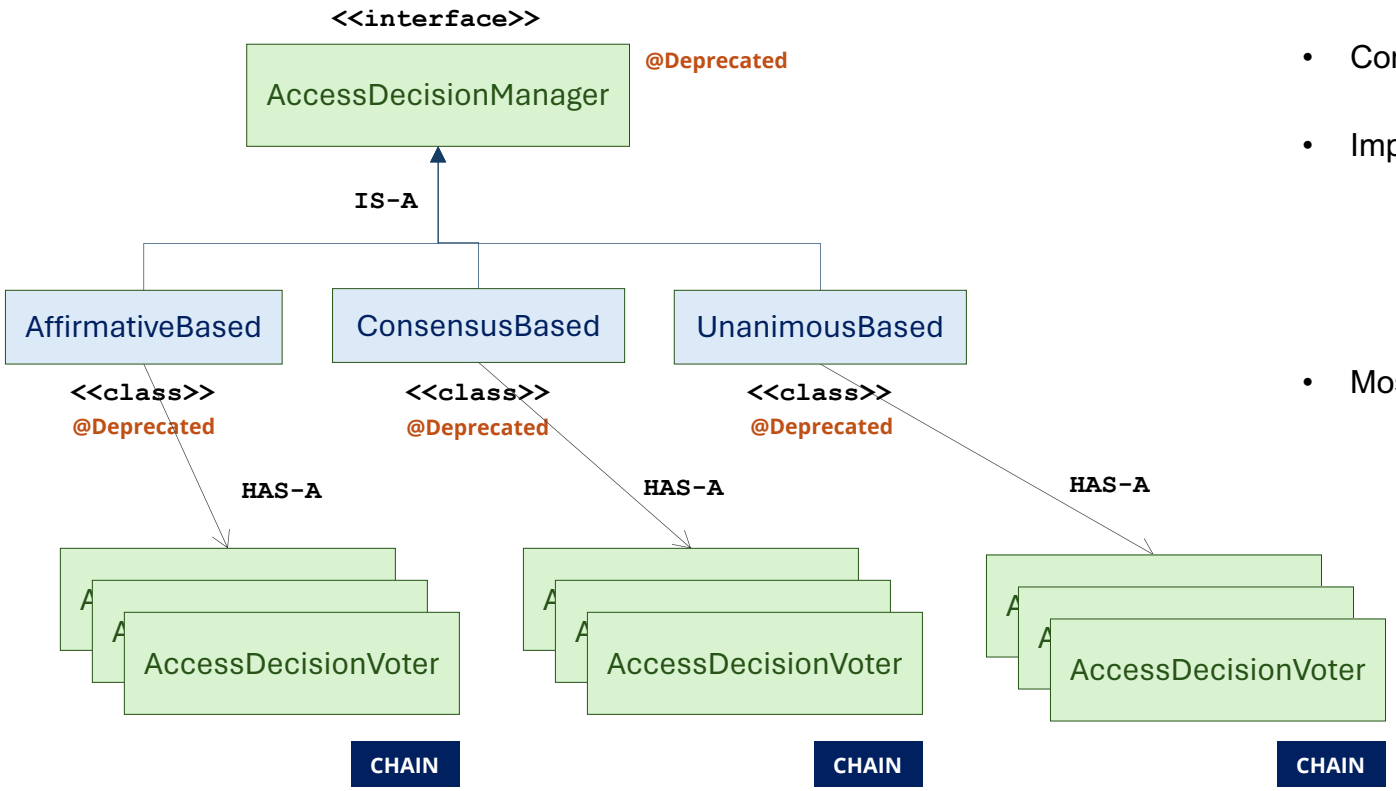
Authorization Outline

- AccessDecisionManager
- AccessDecisionVoter
- ConfigAttribute

AccessDecisionManager



- Core strategy
- Implementations
  - AffirmativeBased, ConsensusBased and UnanimousBased.
  - All three delegate to a chain of AccessDecisionVoter instances
- Most people use AffirmativeBased as **default** AccessDecisionManager
  - access is granted if any voters return affirmatively

```
void decide(Authentication authentication, Object object, Collection<ConfigAttribute> configAttributes)
      throws AccessDeniedException, InsufficientAuthenticationException;

boolean supports(ConfigAttribute attribute);

boolean supports(Class<?> clazz);
```

AccessDecisionVoter

- considers an Authentication (representing a principal) and a **secure** Object, which has been decorated with ConfigAttributes.

```
int vote(Authentication authentication, S object, Collection<ConfigAttribute> attributes);

boolean supports(ConfigAttribute attribute);

boolean supports(Class<?> clazz);
```

- Object is completely generic in the signatures of AccessDecisionManager and AccessDecisionVoter. It represents anything that a user might want to access:
  - i.e., a web resource or a method in a Java class.
- ConfigAttributes represent a decoration of the **secure** Object with some metadata that determines the ***level of permission required to access it***.

ConfigAttribute

<<interface>>

ConfigAttribute

```
String getAttribute();
```

- Has only one API, getAttribute() which returns String.
- These strings encode the **intention of the owner of the resource**, expressing rules about who is allowed to access it:
  - A typical example is the name of a user role
    - i.e., **ROLE_ADMIN** or **ROLE_AUDIT**
  - represent **expressions** that need to be evaluated
    - Spring Expression Language (SpEL) expressions
    - isFullyAuthenticated() && hasRole('user')