This is a Java class named `Voter`, which represents the entity for voters in the application. The class is annotated with various Lombok annotations (`@Data`, `@Builder`, `@AllArgsConstructor`, and `@NoArgsConstructor`) to automatically generate getter, setter, equals, hashCode, and toString methods. Additionally, the class implements the `UserDetails` interface provided by Spring Security to represent user details for authentication and authorization. Let's explain each part of the class:

1. `@Entity`: This annotation marks the class as a JPA entity, representing a table in the database.

2. `@Data`: This Lombok annotation automatically generates getter and setter methods for all class fields, as well as the `toString`, `equals`, and `hashCode` methods.

3. `@Builder`: This Lombok annotation generates a builder pattern for creating instances of the class with a concise and readable syntax.

4. `@Table(name = "voter_details")`: This annotation specifies the name of the table in the database where the `Voter` entity will be stored.

5. `@AllArgsConstructor`: This Lombok annotation generates a constructor with arguments for all fields in the class.

6. `@NoArgsConstructor`: This Lombok annotation generates a default constructor with no arguments.

7. `@Id`: This annotation marks the `id` field as the primary key of the table.

8. `@GeneratedValue`: This annotation specifies that the value for the `id` field will be automatically generated by the database upon insertion of a new record.

9. `private Integer id;`: This field holds the unique identifier for each voter.

10. `private String email;`: This field holds the email of the voter.

11. `private String password;`: This field holds the password of the voter.

12. `@Column(name = "voter_age")`: This annotation specifies the column name in the database table where the `voterAge` field will be stored.

13. `private Integer voterAge;`: This field holds the age of the voter.

14. `private String voterName;`: This field holds the name of the voter.

15. `@Enumerated(EnumType.STRING)`: This annotation specifies that the `role` field will be mapped as a string in the database. It is an enumeration representing the role of the voter, such as "User," "Admin," etc.

16. `private Role role;`: This field holds the role of the voter, represented by the `Role` enum.

17. The class implements the `UserDetails` interface, which is provided by Spring Security to represent user details for authentication and authorization. It overrides several methods from the `UserDetails` interface, including `getAuthorities()`, `getPassword()`, `getUsername()`, and methods related to account status (account non-expired, non-locked, non-expired credentials, and whether the account is enabled). These methods provide necessary information for Spring Security to perform user authentication and authorization.

In summary, the `Voter` class represents the entity for voters in the application. It includes fields for voter information such as email, password, age, name, and role. The class also implements the `UserDetails` interface to provide necessary user details for authentication and authorization using Spring Security.