Java Class: User - Explanation

This PDF explains the functionality of the `User` class written in Java, which represents a basic model for a user entity with attributes such as name, CPF (Brazilian ID number), and birthdate.

1. Constructor:

The constructor method is used to initialize the object when it is created. It takes three parameters: `name`, `cpf`, and `birthdate`. The static field `id_user` is incremented every time a new object is instantiated, acting as a counter.

2. Fields:

- `name`: stores the name of the user
- `cpf`: stores the CPF number (a unique identifier)
- `birthdate`: stores the birthdate of the user in yyyy-mm-dd format
- `id_user`: a static field that tracks how many User objects have been created

3. toString() Method:

Overrides the default `toString()` method from the Object class. It returns a human-readable String with the user's name, CPF, and birthdate. Without this, printing a User object would show only the memory reference (e.g., User@1a2b3c).

4. Getters and Setters:

These are methods to access (get) or modify (set) the private attributes of the class. They are used to enforce encapsulation, a core principle of Object-Oriented Programming.

5. Java Record (Optional):

The comment at the bottom suggests replacing the class with a Java record, like:
`public record User(String name, String cpf, String birthdate) {}`

This is useful for immutable data-holding classes, and automatically generates the constructor, `toString()`, `equals()`, and `hashCode()` methods.