Encapsulation (Keep Data Safe)

```java
// Private variable: cannot be accessed directly from outside
private String username;

// Getter: to get the value of username
public String getUsername() {
    return username;
}

// Setter: to change the value of username
public void setUsername(String username) {
    this.username = username;
}
```

Explanation:

- private means no one can directly touch "username" from outside.
- getUsername() lets you safely read the username.
- setUsername() lets you safely change it.

Use Constructors to Build Objects

```java
// Constructor inside User class
public User(String username, String email) {
    this.username = username; // Save given username
    this.email = email;       // Save given email
}
```

Explanation:

- User is the name of the class.
- When you make a new User, you MUST give username and email!
- this.username means "the variable inside the object."

One Class, Many Objects

```
User user1 = new User("Alice", "alice@mail.com");
User user2 = new User("Bob", "bob@mail.com");
```

Explanation:

- user1 and user2 are different Users with different names and emails!
- They were both created from the same User class.

Abstraction (Hide Complexity)

```
public void login() {
    // complex steps like checking database, passwords
}
```

Explanation:

- When you use login(), you don't care how it checks passwords.
- You just call login(), and it "just works."

Inheritance (Reuse Code)

```
Member class is a child of User class
lic class Member extends User {
 private int loyaltyPoints;

 public Member(String username, String email, int loyaltyPoints)
     super(username, email); // Call parent class constructor
     this.loyaltyPoints = loyaltyPoints; // New property
 }
```

Explanation:

- Member is a special kind of User.
- It has all the properties of User **plus** its own property: loyaltyPoints.

Polymorphism (One Name, Many Behaviors)

```
public class User {
    public void showDetails() {
        System.out.println("User details");
    }
}

public class Member extends User {
    @Override
    public void showDetails() {
        System.out.println("Member details with loyalty points");
    }
}
```

Explanation:

- showDetails() is called in both classes.
- But depending on which object it is (User or Member), it shows different messages**.**

RestaurantRegistration/