1. **Giải thích các đoạn code sau:**

* android:layout\_width="match\_parent": Thuộc tính này chỉ định rằng chiều rộng của View sẽ mở rộng để lấp đầy toàn bộ chiều rộng của View cha của nó. Nếu View cha là màn hình, nó sẽ lấp đầy toàn bộ chiều rộng màn hình.
* android:layout\_height="wrap\_content": Thuộc tính này chỉ định rằng chiều cao của View sẽ tự điều chỉnh để vừa với nội dung bên trong nó. Nếu nội dung thay đổi, chiều cao của View cũng sẽ tự động thay đổi theo.
* android:orientation="vertical": Thuộc tính này được sử dụng trong LinearLayout để chỉ định cách các View con bên trong nó được sắp xếp. vertical có nghĩa là các View con sẽ được xếp chồng lên nhau theo chiều dọc, mỗi View con nằm dưới View con trước đó.

1. **So sánh sự khác nhau giữa 2 câu lệnh sau:**

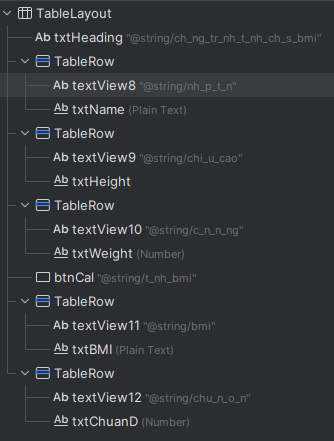
* android:gravity="center": Thuộc tính này kiểm soát cách nội dung bên trong một View được căn chỉnh. Khi đặt là center, nội dung bên trong View đó (ví dụ: văn bản trong TextView, hình ảnh trong ImageView) sẽ được căn giữa theo cả chiều ngang và chiều dọc.
* android:layout\_gravity="center": Thuộc tính này kiểm soát cách một View con được căn chỉnh bên trong View cha của nó. Khi đặt là center, View con đó sẽ được căn giữa theo cả chiều ngang và chiều dọc trong không gian có sẵn của View cha.
* padding & margin:
  + **Padding (Đệm):** Là khoảng trống bên trong một View, giữa viền của View và nội dung của nó. Padding làm tăng kích thước thực tế của nội dung bên trong View. Khi bạn thêm padding, nội dung sẽ bị đẩy vào trong.
  + **Margin (Lề):** Là khoảng trống bên ngoài một View, giữa viền của View đó và các View khác hoặc viền của View cha. Margin tạo ra khoảng cách giữa các View với nhau.

1. **Giải thích ý nghĩa các dòng lệnh sau:**

* DecimalFormat dcf=new DecimalFormat("#.0"); Dòng này tạo một đối tượng DecimalFormat mới. Đối tượng này được sử dụng để định dạng số thập phân thành một chuỗi theo một mẫu cụ thể. Mẫu "# .0" có nghĩa là:
  + #: Đại diện cho một chữ số, nhưng chỉ hiển thị nếu cần (không hiển thị số 0 ở đầu).
  + .: Dấu phân cách thập phân.
  + 0: Đại diện cho một chữ số, và sẽ luôn hiển thị (ngay cả khi là số 0).
  + Vì vậy, mẫu #.0 sẽ định dạng số để luôn có **một chữ số sau dấu thập phân**, ngay cả khi phần thập phân là 0. Ví dụ: 22 sẽ thành 22.0, 22.56 sẽ thành 22.6.
* txtBMI.setText(dcf.format(BMI)); Dòng này lấy giá trị của biến BMI (là một số double), sau đó sử dụng đối tượng dcf (DecimalFormat) để định dạng BMI thành một chuỗi theo mẫu #.0 đã định nghĩa. Cuối cùng, chuỗi kết quả này được đặt làm văn bản hiển thị cho EditText có ID là txtBMI.
* double BMI = W/Math.pow(H\_m, 2); Dòng này tính toán chỉ số Khối lượng cơ thể (BMI).
  + W: Đại diện cho cân nặng của người (đơn vị kg).
  + H\_m: Đại diện cho chiều cao của người (đơn vị mét).
  + Math.pow(H\_m, 2): Là một hàm toán học dùng để tính lũy thừa, ở đây là Hm​ mũ 2 (tức là Hm​×Hm​).
  + Công thức này tính BMI bằng cách chia cân nặng cho bình phương chiều cao.
* double H\_cm = Double.parseDouble(heightStr); Dòng này chuyển đổi một chuỗi (heightStr) thành một số thập phân (double).
  + heightStr: Là một chuỗi chứa giá trị chiều cao (ví dụ: "165").
  + Double.parseDouble(): Là một phương thức tĩnh của lớp Double dùng để phân tích cú pháp một chuỗi và trả về giá trị double tương ứng.
  + Kết quả là biến H\_cm sẽ chứa giá trị số của chiều cao được nhập từ trường văn bản

1. **Vẽ cây phân cấp**

* Chỉ số BMI



* Chuyển đổi nhiệt độ

