Trần Minh Quang – 63CNTT.VA

1. Activity\_main.xml

<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/bill\_amount\_label"  
 *android:id*="@+id/billAmountLabel"  
 *android:textSize*="20sp"  
 *android:textStyle*="bold"  
 *android:layout\_alignParentTop*="true"  
 *android:layout\_alignParentLeft*="true"  
 *android:layout\_alignParentStart*="true"  
 *android:paddingTop*="10dp" />  
  
<EditText  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:inputType*="numberDecimal"  
 *android:ems*="8"  
 *android:id*="@+id/billAmountEditText"  
 *android:text*="@string/bill\_amount"  
 *android:textSize*="20sp"  
 *android:layout\_alignTop*="@+id/billAmountLabel"  
 *android:layout\_toRightOf*="@+id/billAmountLabel"  
 *android:layout\_toEndOf*="@+id/billAmountLabel"  
 *android:layout\_marginLeft*="16dp" />  
  
*<!-- The tip percent -->*<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/tip\_percent\_label"  
 *android:id*="@+id/percentLabel"  
 *android:textSize*="20sp"  
 *android:textStyle*="bold"  
 *android:layout\_alignBottom*="@+id/percentDownButton"  
 *android:layout\_alignParentLeft*="true"  
 *android:layout\_alignParentStart*="true" />  
  
<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/tip\_percent"  
 *android:id*="@+id/percentTextView"  
 *android:layout\_alignTop*="@+id/percentLabel"  
 *android:layout\_alignLeft*="@+id/billAmountEditText"  
 *android:layout\_alignStart*="@+id/billAmountEditText"  
 *android:textSize*="20sp" />  
  
<Button  
 *android:layout\_width*="58dp"  
 *android:layout\_height*="45dp"  
 *android:text*="@string/decrease"  
 *android:id*="@+id/percentDownButton"  
 *android:layout\_below*="@+id/billAmountEditText"  
 *android:layout\_toLeftOf*="@+id/percentUpButton"  
 *android:layout\_toStartOf*="@+id/percentUpButton"  
 *android:textSize*="20sp" />

<Button  
 *android:layout\_width*="58dp"  
 *android:layout\_height*="45dp"  
 *android:text*="@string/increase"  
 *android:id*="@+id/percentUpButton"  
 *android:layout\_alignTop*="@+id/percentDownButton"  
 *android:layout\_alignRight*="@+id/billAmountEditText"  
 *android:layout\_alignEnd*="@+id/billAmountEditText"  
 *android:textSize*="20sp" />  
  
*<!-- The tip amount -->*<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/tip\_amount\_label"  
 *android:id*="@+id/tipLabel"  
 *android:textSize*="20sp"  
 *android:textStyle*="bold"  
 *android:layout\_marginTop*="23dp"  
 *android:layout\_below*="@+id/percentDownButton"  
 *android:layout\_alignParentLeft*="true"  
 *android:layout\_alignParentStart*="true" />  
  
<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/tip\_amount"  
 *android:id*="@+id/tipTextView"  
 *android:layout\_alignTop*="@+id/tipLabel"  
 *android:layout\_alignLeft*="@+id/percentTextView"  
 *android:layout\_alignStart*="@+id/percentTextView"  
 *android:textSize*="20sp" />  
  
*<!-- The total amount -->*<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/total\_amount\_label"  
 *android:id*="@+id/totalLabel"  
 *android:textSize*="20sp"  
 *android:textStyle*="bold"  
 *android:layout\_below*="@+id/tipLabel"  
 *android:layout\_alignParentLeft*="true"  
 *android:layout\_alignParentStart*="true"  
 *android:layout\_marginTop*="24dp" />  
  
<TextView  
 *android:layout\_width*="wrap\_content"  
 *android:layout\_height*="wrap\_content"  
 *android:text*="@string/total\_amount"  
 *android:id*="@+id/totalTextView"  
 *android:textSize*="20sp"  
 *android:layout\_alignTop*="@+id/totalLabel"  
 *android:layout\_alignLeft*="@+id/tipTextView"  
 *android:layout\_alignStart*="@+id/tipTextView" />

Giải thích:

0. TextView (hiển thị thông tin như tiêu đề, mô tả), EditText(Nơi nhập văn bản, dữ liệu), Button(Cho phép người dung tương tác bằng cách nhấn vào nó)

1. android:layout\_width: Xác định chiều rộng của thành phần giao diện (ví dụ: "wrap\_content" hoặc "match\_parent").

2. android:layout\_height: Xác định chiều cao của thành phần giao diện (ví dụ: "wrap\_content" hoặc "match\_parent").

3. android:text: Thiết lập văn bản của thành phần (ví dụ: "@string/hello" để lấy văn bản từ tệp chuỗi hoặc "Hello, World!" để đặt trực tiếp văn bản).

4. android:id: Xác định một ID duy nhất cho thành phần, giúp trong việc xác định và thao tác với nó từ mã Java.

5. android:textSize: Đặt kích thước của văn bản.

6. android:textStyle: Thiết lập kiểu văn bản (ví dụ: "bold" cho văn bản đậm).

7. android:layout\_alignParentTop: Căn chỉnh thành phần với phía trên của cha của nó.

8. android:layout\_alignBottom: Căn chỉnh phía dưới của thành phần với phía dưới của một thành phần khác.

9. android:layout\_alignParentLeft: Căn chỉnh thành phần với phía trái của cha của nó.

10. android:layout\_alignParentStart: Căn chỉnh thành phần với phía bắt đầu (trái trong ngôn ngữ viết từ phải sang trái hoặc phải trong ngôn ngữ viết từ trái sang phải) của cha của nó.

11. android:paddingTop: Đặt khoảng cách phía trên của thành phần.

12. android:inputType: Xác định kiểu dữ liệu đầu vào cho thành phần (ví dụ: "textPassword" cho mật khẩu, "number" cho số).

13. android:ems: Xác định số lượng ký tự em trong EditText.

14. android:layout\_alignTop: Căn chỉnh phía trên của thành phần với phía trên của một thành phần khác.

15. android:layout\_alignRight: Căn chỉnh phía phải của thành phần với phía phải của một thành phần khác.

16. android:layout\_toRightOf: Đặt vị trí của thành phần bên phải của một thành phần khác.

17. android:layout\_toEndOf: Đặt vị trí của thành phần bên phải (trong ngôn ngữ viết từ phải sang trái) của một thành phần khác.

18. android:layout\_marginLeft: Thiết lập lề bên trái của thành phần.

19. android:layout\_marginTop: Thiết lập lề phía trên của thành phần.

20. android:layout\_below: Xác định 1 thành phần giao diện cần được đặt phía dưới của 1 thành phần khác.

1. MainActivity.java

*import* android.os.Bundle;  
  
*import* androidx.activity.EdgeToEdge;  
*import* androidx.appcompat.app.AppCompatActivity;  
*import* androidx.core.graphics.Insets;  
*import* androidx.core.view.ViewCompat;  
*import* androidx.core.view.WindowInsetsCompat;  
  
*import* java.text.NumberFormat;  
*import* android.app.Activity;  
*import* android.os.Bundle;  
*import* android.view.KeyEvent;  
*import* android.view.View;  
*import* android.view.View.*OnClickListener*;  
*import* android.view.inputmethod.EditorInfo;  
*import* android.widget.Button;  
*import* android.widget.EditText;  
*import* android.widget.TextView;  
*import* android.widget.TextView.*OnEditorActionListener*;  
*import* android.content.*SharedPreferences*;  
*import* android.content.*SharedPreferences*.*Editor*;

1. `import android.os.Bundle;`: Import lớp Bundle từ gói android.os, được sử dụng để lưu trữ cặp key-value trong Android.

2. `import androidx.activity.EdgeToEdge;`: Import lớp EdgeToEdge từ androidx.activity, được sử dụng để điều chỉnh giao diện với biên của màn hình.

3. `import androidx.appcompat.app.AppCompatActivity;`: Import lớp AppCompatActivity từ androidx.appcompat.app, một lớp cơ sở để tạo các hoạt động (activities) trong ứng dụng Android, hỗ trợ các tính năng của thư viện AppCompat.

4. `import androidx.core.graphics.Insets;`: Import lớp Insets từ androidx.core.graphics, được sử dụng để đại diện cho các biên của một phần tử giao diện.

5. `import androidx.core.view.ViewCompat;`: Import lớp ViewCompat từ androidx.core.view, cung cấp các phương thức tiện ích để làm việc với các thành phần giao diện người dùng.

6. `import androidx.core.view.WindowInsetsCompat;`: Import lớp WindowInsetsCompat từ androidx.core.view, đại diện cho các thay đổi trong kích thước và biên của cửa sổ.

7. `import java.text.NumberFormat;`: Import lớp NumberFormat từ gói java.text, được sử dụng để định dạng số.

8. `import android.app.Activity;`: Import lớp Activity từ gói android.app, lớp cơ sở cho tất cả các hoạt động trong ứng dụng Android.

9. `import android.view.KeyEvent;`: Import lớp KeyEvent từ gói android.view, được sử dụng để xử lý sự kiện liên quan đến các phím trên bàn phím.

10. `import android.view.View;`, `import android.view.View.OnClickListener;`: Import lớp View và OnClickListener từ gói android.view, được sử dụng để tương tác với các thành phần giao diện người dùng và xử lý sự kiện khi chúng được nhấn.

11. `import android.view.inputmethod.EditorInfo;`: Import lớp EditorInfo từ gói android.view.inputmethod, được sử dụng để định nghĩa hành vi của bàn phím khi nhập liệu vào một EditText.

12. `import android.widget.Button;`, `import android.widget.EditText;`, `import android.widget.TextView;`, `import android.widget.TextView.OnEditorActionListener;`: Import các lớp Button, EditText, TextView và OnEditorActionListener từ gói android.widget, được sử dụng để tạo và điều khiển các thành phần giao diện người dùng như Button, EditText và TextView.

13. `import android.content.SharedPreferences;`, `import android.content.SharedPreferences.Editor;`: Import lớp SharedPreferences và Editor từ gói android.content, được sử dụng để lưu trữ dữ liệu của ứng dụng trong các cặp key-value.

*public class* MainActivity *extends* AppCompatActivity  
 *implements OnEditorActionListener*, *OnClickListener* {

1. `public class MainActivity extends AppCompatActivity`: Đây là khai báo của lớp `MainActivity. Lớp này mở rộng từ lớp `AppCompatActivity.

2. Interface `OnEditorActionListener`: Định nghĩa một phương thức `onEditorAction()`, được gọi khi một hành động nhập liệu được thực hiện trên một EditText. Lớp `MainActivity` phải cung cấp một cài đặt cho phương thức này để xử lý sự kiện khi một hành động nhập liệu được thực hiện trên một EditText.

3. Interface `OnClickListener`: Định nghĩa một phương thức `onClick()`, được gọi khi một View (như Button) được nhấn. Lớp `MainActivity` phải cung cấp một cài đặt cho phương thức này để xử lý sự kiện khi một View được nhấn.

*private* EditText billAmountEditText;  
*private* TextView percentTextView;  
*private* Button percentUpButton;  
*private* Button percentDownButton;  
*private* TextView tipTextView;  
*private* TextView totalTextView;  
*private SharedPreferences* savedValues;  
*private* String billAmountString = "";  
*private float* tipPercent = .15f;

1. private EditText abc: Được dùng để tham chiếu đến một đối tượng EditText .

2. private TextView abc: Được dùng để tham chiếu đến một đối tượng TextView .

3. private Button abc: Được dùng để tham chiếu đến một đối tượng Button .

4. private SharedPreferences savedValues: Đây là một đối tượng SharedPreferences được sử dụng để lưu trữ các giá trị đã được lưu trữ trước đó trong ứng dụng.

5. private String billAmountString = "": Biến này là một chuỗi dùng để lưu trữ số tiền hóa đơn dưới dạng chuỗi ký tự.

6. private float tipPercent = .15f: Biến này là một số thực dùng để lưu trữ phần trăm tip mặc định, ở đây được thiết lập là 15%.

@Override  
*public void* onCreate(Bundle savedInstanceState) {  
 *super*.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 *// get references to the widgets* billAmountEditText = (EditText) findViewById(R.id.*billAmountEditText*);  
 percentTextView = (TextView) findViewById(R.id.*percentTextView*);  
 percentUpButton = (Button) findViewById(R.id.*percentUpButton*);  
 percentDownButton = (Button) findViewById(R.id.*percentDownButton*);  
 tipTextView = (TextView) findViewById(R.id.*tipTextView*);  
 totalTextView = (TextView) findViewById(R.id.*totalTextView*);  
  
 *// set the listeners* billAmountEditText.setOnEditorActionListener(*this*);  
 percentUpButton.setOnClickListener(*this*);  
 percentDownButton.setOnClickListener(*this*);  
  
 *// get SharedPreferences object* savedValues = getSharedPreferences("SavedValues", ***MODE\_PRIVATE***);  
}

1. `super.onCreate(savedInstanceState);`: Gọi phương thức `onCreate()` của lớp cơ sở, thường là `AppCompatActivity`, để thực hiện các khởi tạo cần thiết.

2. `setContentView(R.layout.activity\_main);`: Đặt giao diện người dùng cho activity từ tệp layout có tên là "activity\_main.xml". Điều này xác định cách các thành phần giao diện được hiển thị trên màn hình.

3. Khối comment `// get references to the widgets` giải thích rằng các đoạn mã tiếp theo đều là để lấy tham chiếu đến các thành phần giao diện từ tệp layout.

4. `billAmountEditText = (EditText) findViewById(R.id.billAmountEditText);`: Gán tham chiếu đến EditText có ID là "billAmountEditText" trong layout "activity\_main.xml" vào biến `billAmountEditText`.

5. `percentTextView = (TextView) findViewById(R.id.percentTextView);`: Tương tự như trên, nhưng lần này là TextView với ID là "percentTextView".

6. `percentUpButton = (Button) findViewById(R.id.percentUpButton);`: Tương tự như trên, nhưng lần này là Button với ID là "percentUpButton".

7. `percentDownButton = (Button) findViewById(R.id.percentDownButton);`: Tương tự như trên, nhưng lần này là Button với ID là "percentDownButton".

8. `tipTextView = (TextView) findViewById(R.id.tipTextView);`: Tương tự như trên, nhưng lần này là TextView với ID là "tipTextView".

9. `totalTextView = (TextView) findViewById(R.id.totalTextView);`: Tương tự như trên, nhưng lần này là TextView với ID là "totalTextView".

10. `billAmountEditText.setOnEditorActionListener(this);`: Thiết lập một OnEditorActionListener cho EditText `billAmountEditText`, điều này có nghĩa là phương thức `onEditorAction()` của `MainActivity` sẽ được gọi khi một hành động nhập liệu được thực hiện trên EditText này.

11. `percentUpButton.setOnClickListener(this);`: Thiết lập một OnClickListener cho Button `percentUpButton`, điều này có nghĩa là phương thức `onClick()` của `MainActivity` sẽ được gọi khi Button này được nhấn.

12. `percentDownButton.setOnClickListener(this);`: Tương tự như trên, nhưng lần này áp dụng cho Button `percentDownButton`.

13. `savedValues = getSharedPreferences("SavedValues", MODE\_PRIVATE);`: Lấy đối tượng SharedPreferences với tên là "SavedValues". Đối tượng này được sử dụng để lưu trữ và truy cập các giá trị trong toàn bộ ứng dụng.

@Override  
*public void* onPause() {  
 *// save the instance variables  
 Editor* editor = savedValues.edit();  
 editor.putString("billAmountString", billAmountString);  
 editor.putFloat("tipPercent", tipPercent);  
 editor.commit();  
  
 *super*.onPause();

}

1. `Editor editor = savedValues.edit();`: Tạo một đối tượng Editor từ đối tượng SharedPreferences `savedValues`. Đối tượng Editor này được sử dụng để chỉnh sửa và lưu trữ các giá trị trong SharedPreferences.

2. `editor.putString("billAmountString", billAmountString);`: Lưu trữ giá trị của biến `billAmountString` vào SharedPreferences với key là "billAmountString". Điều này giữ cho giá trị của biến này được lưu lại khi activity tạm dừng hoặc bị hủy.

3. `editor.putFloat("tipPercent", tipPercent);`: Tương tự như trên, nhưng lần này là lưu trữ giá trị của biến `tipPercent` dưới dạng số thực.

4. `editor.commit();`: Áp dụng các thay đổi vào SharedPreferences bằng cách gọi phương thức `commit()`. Điều này đảm bảo rằng các giá trị được lưu lại ngay lập tức.

5. `super.onPause();`: Gọi phương thức `onPause()` của lớp cơ sở để thực hiện bất kỳ hoạt động dọn dẹp hoặc tiếp tục nào mà nó định nghĩa.

@Override  
*public void* onResume() {  
 *super*.onResume();  
  
 *// get the instance variables* billAmountString = savedValues.getString("billAmountString", "");  
 tipPercent = savedValues.getFloat("tipPercent", 0.15f);  
  
 *// set the bill amount on its widget* billAmountEditText.setText(billAmountString);  
  
 *// calculate and display* calculateAndDisplay();  
}

1. `super.onResume();`: Gọi phương thức `onResume()` của lớp cơ sở để thực hiện bất kỳ hoạt động khởi đầu hoặc tiếp tục nào mà nó định nghĩa.

2. `billAmountString = savedValues.getString("billAmountString", "");`: Đọc giá trị của chuỗi "billAmountString" từ SharedPreferences `savedValues`. Nếu không có giá trị nào được lưu trữ với key "billAmountString", mặc định sẽ là một chuỗi rỗng.

3. `tipPercent = savedValues.getFloat("tipPercent", 0.15f);`: Tương tự như trên, nhưng lần này là đọc giá trị của số thực "tipPercent" từ SharedPreferences. Nếu không có giá trị nào được lưu trữ, mặc định sẽ là 0.15f (tức là 15%).

4. `billAmountEditText.setText(billAmountString);`: Đặt văn bản trên EditText `billAmountEditText` thành giá trị của `billAmountString`. Điều này đảm bảo rằng số tiền hóa đơn được hiển thị lại cho người dùng khi activity được khôi phục.

5. `calculateAndDisplay();`: Gọi phương thức `calculateAndDisplay()` để tính toán và hiển thị số tiền tip và tổng số tiền cần thanh toán. Điều này đảm bảo rằng các giá trị tip và tổng số tiền được hiển thị lại cho người dùng khi activity được khôi phục.

*public void* calculateAndDisplay() {  
  
 *// get the bill amount* billAmountString = billAmountEditText.getText().toString();  
 *float* billAmount;  
 *if* (billAmountString.equals("")) {  
 billAmount = 0;  
 }  
 *else* {  
 billAmount = Float.*parseFloat*(billAmountString);  
 }  
  
 *// calculate tip and total  
 float* tipAmount = billAmount \* tipPercent;  
 *float* totalAmount = billAmount + tipAmount;  
  
 *// display the other results with formatting* NumberFormat currency = NumberFormat.*getCurrencyInstance*();  
 tipTextView.setText(currency.format(tipAmount));  
 totalTextView.setText(currency.format(totalAmount));  
  
 NumberFormat percent = NumberFormat.*getPercentInstance*();  
 percentTextView.setText(percent.format(tipPercent));  
}

1. Lấy số tiền hóa đơn từ EditText `billAmountEditText`.

2. Kiểm tra xem liệu số tiền hóa đơn có trống không (không nhập) hay không. Nếu trống, giá trị `billAmount` được gán là 0, ngược lại thì nó được chuyển đổi từ chuỗi thành số thực (`float`) sử dụng phương thức `Float.parseFloat()`.

3. Tính toán số tiền tip và tổng số tiền cần thanh toán.

4. Định dạng số tiền tip và tổng số tiền cần thanh toán thành chuỗi có định dạng tiền tệ bằng cách sử dụng `NumberFormat.getCurrencyInstance()`.

5. Đặt văn bản của TextView `tipTextView` và `totalTextView` thành các chuỗi đã định dạng để hiển thị số tiền tip và tổng số tiền.

6. Định dạng phần trăm tip và hiển thị nó trong TextView `percentTextView` sử dụng `NumberFormat.getPercentInstance()`.

@Override  
*public boolean* onEditorAction(TextView v, *int* actionId, KeyEvent event) {  
 *if* (actionId == EditorInfo.***IME\_ACTION\_DONE*** ||  
 actionId == EditorInfo.***IME\_ACTION\_UNSPECIFIED***) {  
 calculateAndDisplay();  
 }  
 *return false*;  
}

1. `public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {`: Đây là khai báo của phương thức `onEditorAction()`. Nó nhận ba đối số: TextView `v` (EditText mà sự kiện được kích hoạt), mã hành động `actionId` (loại hành động nhập liệu) và sự kiện KeyEvent `event` (nếu có).

2. `if (actionId == EditorInfo.IME\_ACTION\_DONE || actionId == EditorInfo.IME\_ACTION\_UNSPECIFIED) {`: Kiểm tra xem hành động nhập liệu có phải là `IME\_ACTION\_DONE` (hoặc không xác định) hay không. `IME\_ACTION\_DONE` là hành động khi người dùng ấn phím "Done" trên bàn phím ảo. `IME\_ACTION\_UNSPECIFIED` là một hành động không xác định cụ thể, cho phép xử lý nhiều loại hành động.

3. `calculateAndDisplay();`: Gọi phương thức `calculateAndDisplay()` khi hành động nhập liệu là `IME\_ACTION\_DONE` hoặc `IME\_ACTION\_UNSPECIFIED`. Điều này đảm bảo rằng khi người dùng hoàn thành nhập liệu trên EditText (bằng cách nhấn "Done" trên bàn phím), phương thức `calculateAndDisplay()` sẽ được gọi để tính toán và hiển thị số tiền tip và tổng số tiền cần thanh toán.

4. `return false;`: Trả về false để chỉ ra rằng bạn không muốn tiêu thụ sự kiện này và bạn muốn để TextView xử lý tiếp theo theo luật mặc định của nó.

@Override  
 *public void* onClick(View v) {  
 *if*(v.getId() == R.id.*percentDownButton*) {  
 tipPercent = tipPercent - .01f;  
 calculateAndDisplay();  
 }  
 *else if*(v.getId() == R.id.*percentUpButton*){  
 tipPercent = tipPercent + .01f;  
 calculateAndDisplay();  
 }  
 }  
}

1. `public void onClick(View v)`: Đây là khai báo của phương thức `onClick()`. Nó nhận một đối số là View `v`, đại diện cho View (Button) đã được nhấn.

2. `if(v.getId() == R.id.percentDownButton)`: Kiểm tra xem View đã được nhấn có ID là `percentDownButton` hay không. Nếu có, đây là Button `percentDownButton` được nhấn.

- `tipPercent = tipPercent - .01f;`: Giảm giá trị của `tipPercent` đi 0.01 (tương đương với 1%).

3. `else if(v.getId() == R.id.percentUpButton) `: Nếu không phải là Button `percentDownButton`, kiểm tra xem View đã được nhấn có ID là `percentUpButton` hay không. Nếu có, đây là Button `percentUpButton` được nhấn.

- `tipPercent = tipPercent + .01f;`: Tăng giá trị của `tipPercent` thêm 0.01 (tương đương với 1%).

4. `calculateAndDisplay();`: Gọi phương thức `calculateAndDisplay()` sau khi giá trị `tipPercent` đã được thay đổi. Điều này đảm bảo rằng khi người dùng nhấn nút tăng hoặc giảm phần trăm tip, phương thức `calculateAndDisplay()` sẽ được gọi để tính toán và hiển thị lại số tiền tip và tổng số tiền cần thanh toán với phần trăm tip mới.