**Android Final vs Private textviews**

simple maybe stupid question. I have a login activity which launches another activity, and here's the code:

public class LoginActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.login);

ActionBar actionBar = getActionBar();

actionBar.hide();

Button btnLogin = (Button) findViewById(R.id.btnLogin);

final TextView tvUsername = (TextView) findViewById(R.id.tvUsername);

final TextView tvPassword = (TextView) findViewById(R.id.tvPassword);

btnLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if (tvUsername.getText().toString().length()<1) {

Toast msg = Toast.makeText(LoginActivity.this, "Enter a Username", Toast.LENGTH\_LONG);

msg.setGravity(Gravity.TOP|Gravity.LEFT, 0, 70);

msg.show();

tvUsername.requestFocus();

} else if (tvPassword.getText().toString().length()<1) {

Toast msg = Toast.makeText(LoginActivity.this, "Enter a Password", Toast.LENGTH\_LONG);

msg.setGravity(Gravity.TOP|Gravity.LEFT, 0, 200);

msg.show();

tvPassword.requestFocus();

} else {

startActivity(new Intent(LoginActivity.this,CrewScheduleActivity.class));

finish();

}

}

});

}

}

My question is about the textviews. Eclipse basically said i had to make them final in order to use them in the onClick event of the button. NP so i did that and it worked.

The question is What is the difference between putting these above the @Override as private vs inside the OnCreate as final?

This has to do with [closure in Java](http://en.wikipedia.org/wiki/Closure_%28computer_science%29#Local_classes_.28Java.29). Basically, when you use an anonymous class, the values (not objects themselves) used within it are copied to that class for usage. Therefore, it does not make sense to return or modify those variables within the class, hence they must be final.

However, if the variable is instead part of the class *containing* the anonymous class, that's different. Basically, your inner class has a reference to the LoginActivity object (as LoginActivity.this), and can use and modify its members and methods.

When you put them "above the @Override", you are making them member variables of the LoginActivity class. Therefore, they can be accessed by the anonymous class.

Succinctly, the difference is that: final variables are local to the method, and copied to the anonymous class; member variables are local to the containing class and are modified by the anonymous class.

If you want to reuse the data from the anonymous class later, use a member variable. If you only need it within onCreate() and the anonymous class, then a final variable will suffice.