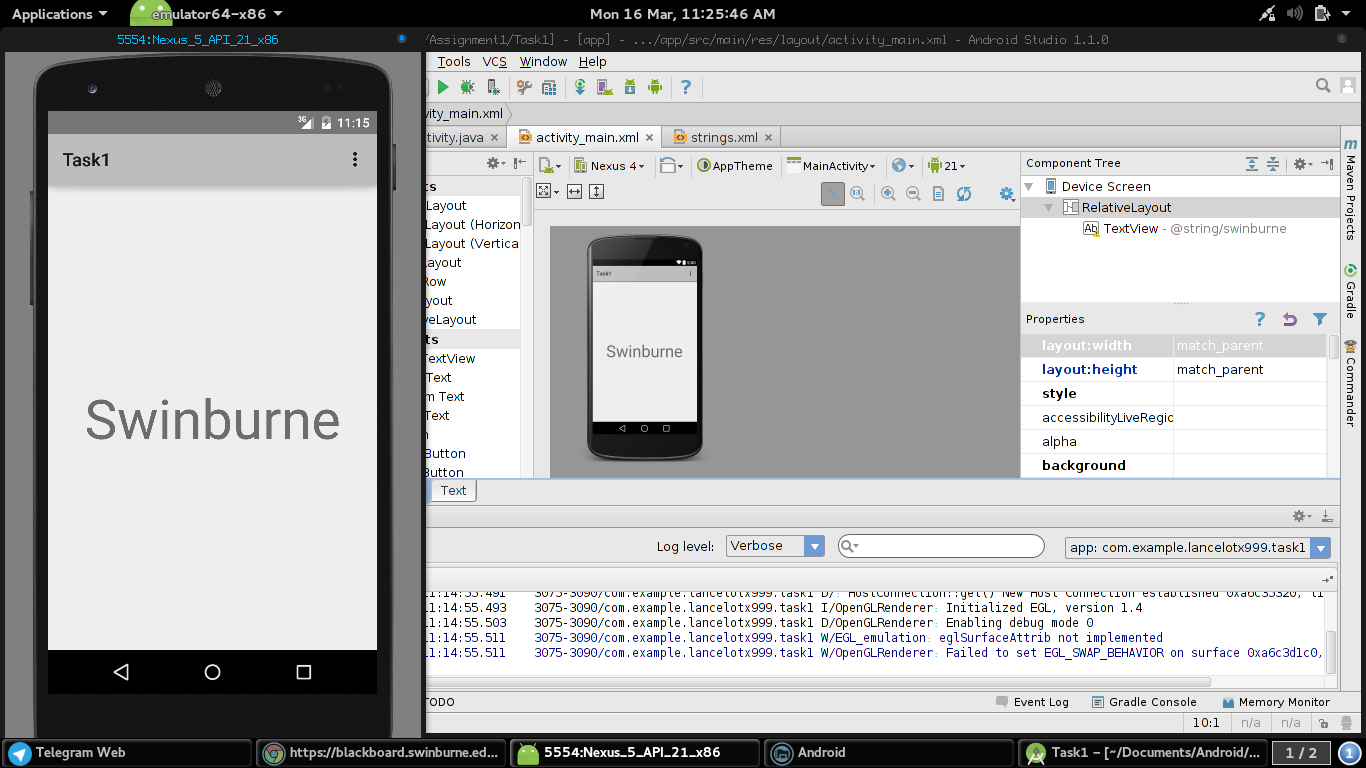
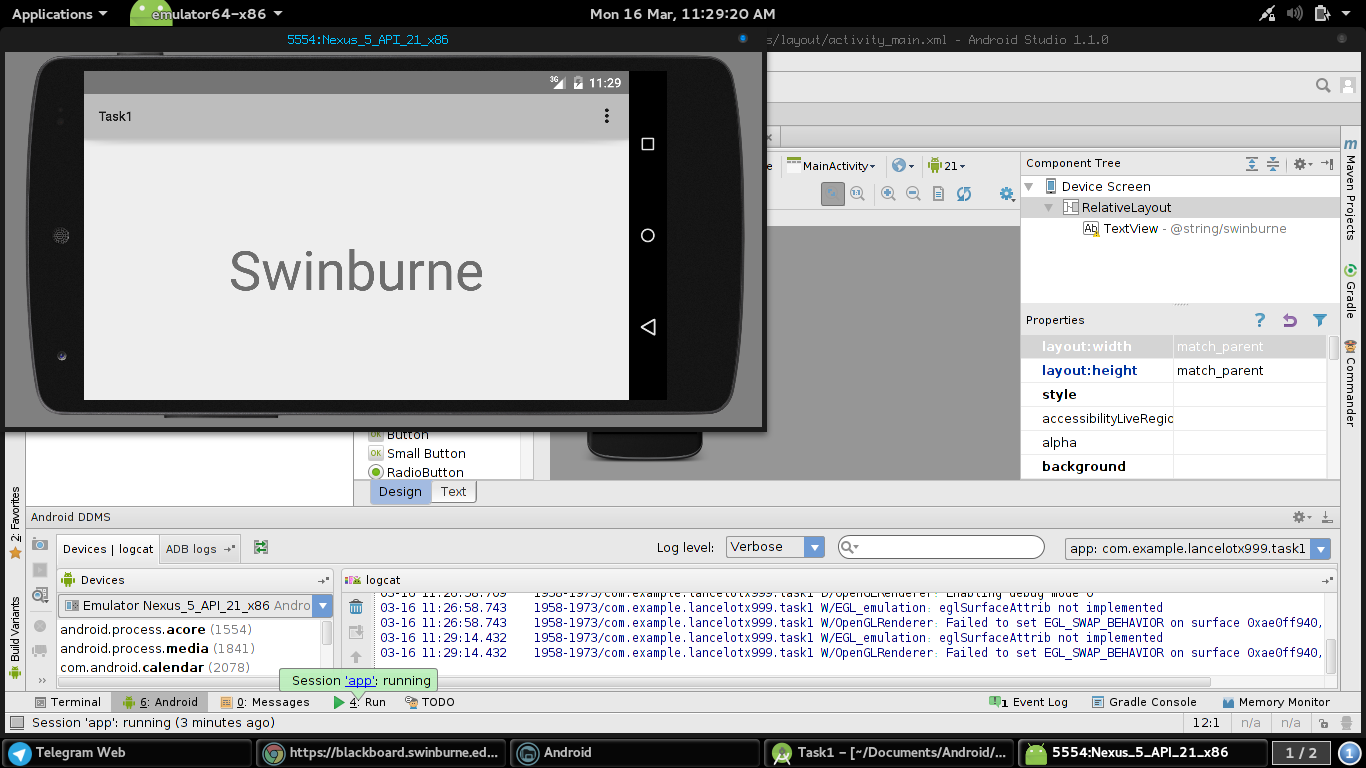
**Submission For Assignment 1**

**Task 1**



**Task 2**

Convention over configuration is a design paradigm which is aimed to reduce the number of decision that a developer needs to make, making thing simpler but not reducing the flexibility of the design by much.

Convention over configuration is intended to design a framework so a standard naming convention is needed to map classes to resources or events. A programmer only needs to rewrite the mapping configuration when the naming convention fails. This is very important when designing big projects so that a standard of coding can be used and the project can be passed on to others without needed major explanation.

Convention over configuration basically means that a developer only needs to specify the unconventional aspect of the software. For example, a class Item in a model, the corresponding table in the database will be called “item” by default. Only if a developer deviates from the convention such as renaming the table “items”, that a developer must write codes regarding these names.

When the tool implemented by the convention matches the behavior that is desired, it behaves as it should have without needing to write any sort of configuration files. Only when there is a deviation from the expected behavior that a configuration files is needed.

Examples of frameworks that use this paradigm:

1. Ruby on Rails
2. Laravel

**Task 3**

Mobile devices require device independent font to ensure that regardless of the type or size of font, software is able to function and display properly on different devices. For example, an application designed on Samsung device is able to perform flawlessly on a Sony device. This is where SP font settings come in, SP stands for scale-independent pixels which means regardless of screen size the font size will appear the same. If software is made using a font unavailable to a certain device it will not display properly or not display at all. Image sizing is done to ensure that images display properly on all devices and all sizes of screens. For example an image displayed on a tablet is able to be displayed properly on a smartphone. An image is resized according to scale to ensure the proper display is maintained. This ensures an image is not undersized or oversized.

**Task 4**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent"

android:layout\_height="match\_parent" android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:paddingBottom="@dimen/activity\_vertical\_margin" tools:context=".MainActivity">

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:layout\_centerVertical="true"

android:layout\_centerHorizontal="true">

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/Header">

<ImageView

android:layout\_width="119dp"

android:layout\_height="122dp"

android:id="@+id/ImgLogo"

android:src="@drawable/logo"

android:contentDescription="@string/app\_name" />

</LinearLayout>

<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:id="@+id/Form">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/Email"

android:id="@+id/TxtEmail"

android:textSize="20sp" />

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="textEmailAddress"

android:ems="10"

android:id="@+id/FieldEmail" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/Password"

android:id="@+id/TxtPassword"

android:textSize="20sp" />

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="textPassword"

android:ems="10"

android:id="@+id/FieldPassword" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/Login"

android:id="@+id/BtnLogin"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="50dp" />

</LinearLayout>

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:id="@+id/Footer"

android:background="@drawable/footer"></LinearLayout>

</LinearLayout>

</RelativeLayout>

**Task 5**

package com.example.lancelotx999.radiobutton;

import android.app.Activity;

import android.media.Image;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.CompoundButton;

import android.widget.ImageView;

import android.widget.RadioButton;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.RadioGroup;

import android.widget.RadioGroup.OnCheckedChangeListener;

public class MainActivity extends Activity {

RadioButton RbBean;

RadioButton RbKenny;

RadioButton RbBart;

RadioGroup RbGrpDisplay;

ImageView ImageDisplay;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

RbBean = (RadioButton)findViewById(R.id.RbBean);

RbKenny = (RadioButton)findViewById(R.id.RbKenny);

RbBart = (RadioButton)findViewById(R.id.RbBart);

RbGrpDisplay = (RadioGroup)findViewById(R.id.RbGrp);

ImageDisplay = (ImageView)findViewById(R.id.ImgDisplay);

RbGrpDisplay.setOnCheckedChangeListener(new OnCheckedChangeListener()

{

@Override

public void onCheckedChanged(RadioGroup group, int checkedId)

{

if (RbBean.isChecked())

{

ImageDisplay.setImageDrawable(getResources().getDrawable(R.drawable.mrbean));

}

else if (RbKenny.isChecked())

{

ImageDisplay.setImageDrawable(getResources().getDrawable(R.drawable.kenny));

}

else if (RbBart.isChecked())

{

ImageDisplay.setImageDrawable(getResources().getDrawable(R.drawable.bart));

}

}

});

}