2016 年 10 月 13 日（木）実施

Java プログラムの基礎

Java 言語プログラムの構成

Java 言語ではプログラムの基本は**クラス**と呼ばれる単位である。クラスの定義は，一般に次の形式となる。この中で，**メソッド**に手続きを記述する。

```
修飾子 class クラス名 {
  型 フィールド名;
  修飾子 戻り値のデータ型 メソッド名( 引数 ){
    メソッドの定義
  }
}
```

変数

プログラム中の命令を通じて，メインメモリ上にデータを格納する領域を確保し，必要に応じてその場所に格納されるデータを上書きすることが可能である。このような領域を**変数**という。Java 言語では，変数の取り扱いに関して，次の様な特徴がある。

1. プログラム中で用いる変数は必ず**宣言**しておく。 ⇒ メインメモリ上にデータを格納する領域を確保せよ，という命令に相当する。

2. 変数の宣言時には，**データ型を指定**する。 ⇒ データ型毎にデータを格納する領域のサイズが固定されている。

   例） `int x;`　// 整数（integer）のデータ型の変数 x を宣言

3. 変数にデータを格納するには，**代入**を行う。

   例） `x=10;`　/* 変数 x に定数 10 を代入（=の右辺を評価し，その値を左辺の変数に格納）
   なお，宣言時に**初期化**することも可能である。 ⇒ `int x=4;` */

4. 代入式の左辺以外に変数名を用いると**参照**が行われ，変数に格納されたデータが取り出されて利用される。

   例） `System.out.println(x);`　// 変数 x の中身を画面に表示

データ型

Java 言語には数値や文字のデータを扱う**基本型(プリミティブ型)**と，クラス等を扱う**参照型**と

がある。基本型には，次のものがある。

分類	型名	サイズ	内容
整数型	char	16 ビット	文字型とも呼ばれる　Unicode 文字（\u0000 ～ \uffff）
	byte	8 ビット	整数（-128 ～ 127）
	short	16 ビット	整数（-32768 ～ 32767）
	int	32 ビット	整数（-2^{31} ～ $2^{31}-1$ = -2147483648 ～ 2147483647）
	long	64 ビット	整数（-2^{63} ～ $2^{63}-1$）
実数型	float	32 ビット	単精度浮動小数点数
	double	64 ビット	倍精度浮動小数点数
論理型	boolean	1 ビット	論理値（true または false）

算術演算

Java 言語で，2 個の変数 a，b の間で加減乗除の演算を行うには，a+b，a−b，a*b，a/b の様に，加減演算子（+，−），乗除演算子（*，/）を用いる。また，乗除演算子には a を b で割った余りを求める際に用いられる『%』もある。

なお，整数を整数で割った場合には，小数点以下は切り捨てられ，結果は整数部のみ残る。

String クラス

Java 言語では，文字列を扱う String クラスがパッケージ java.lang の中に用意されている。String クラスは，作成後に変更できない文字列を表す。

例）　String str = "CUC";

右辺の"CUC"の様に，文字列を半角の二重引用符で挟んだものは**文字列リテラル**と呼ばれ，String クラスの**インスタンス**（"実体"）への参照として扱われる。

プログラムの制御構造

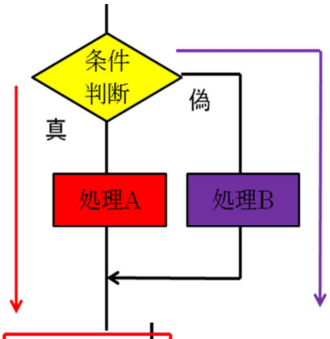
1960 年代後半にダイクストラが提唱した**構造化プログラミング**という考え方では，手続き型のプログラムを記述する際には，**順次**，**選択**，**反復**という標準的な**制御構造**のみを用い，先ずプログラムの概略構造を設計し，その大まかな単位を段階的に詳細化して処理を記述していく。

順次構造

順次構造とは，プログラム中の文を処理していく順に記述したものである。これまで扱ったプログラムは，全て順次構造によって記述されたものであり，最も基本的な制御構造と言える。

プログラムの処理の流れを図示する手法の一つに**流れ図**がある。この流れ図で順次構造を表すと右図のようになる。(色矢印は処理の流れを補足)

選択構造



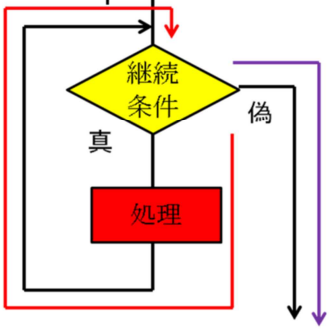
選択構造とは，条件や式の値によってプログラムの処理の流れを分ける構造である。選択構造の基本は **2 分岐**と呼ばれる構造で，この構造を流れ図で表すと右図のようになる。

また，式の値によって，幾つもの異なる処理が必要なときには，**多分岐**という選択構造も利用可能である。

反復構造

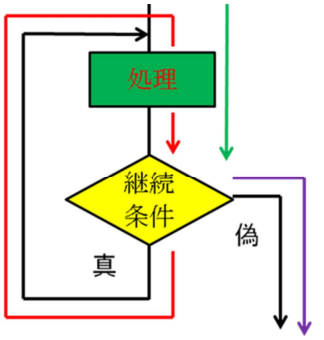
反復構造とは，**継続条件**が満たされている間，定められた範囲内の文に記述された処理を繰り返して実行する構造である。（Java 言語以外のプログラム言語では，**終了条件**が満たされない間，文を繰り返して実行する構造を持つものもある）

なお，反復構造には，右の流れ図で表される 2 種類の場合がある。



1）0 回以上の繰り返し（右上図）

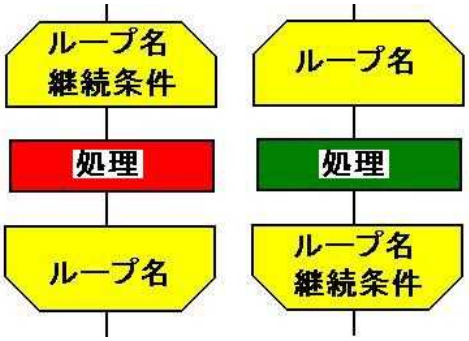
先ず継続条件が判定され，真であれば定められた範囲内の文に記述された処理を実行する。始めから継続条件が満たされない場合には，文は全く実行されないため，0 回以上の繰り返しと呼ばれる。



2）1 回以上の繰り返し（右下図）

先ず定められた範囲内の文に記述された処理が実行され，その後に継続条件が判定される。始めから継続条件が満たされない場合でも，最初の 1 回は定められた範囲内の文に記述された処理が実行されるため，1 回以上の繰り返しと呼ばれる。

＊ 反復構造に対する流れ図に表れる閉線図形を**ループ**と呼ぶことから，反復構造のプログラム構造をループと称することがある。



複雑な処理は，順次構造，選択構造，反復構造の組み合わせで実現される。プログラムの構造は，最も大掴みにした概略構造で見ると順次構造となる。選択構造や反復構造を中間の概略構造と看做した場合，その詳細構造として，それぞれの構造の流れ図で『処理』と書かれた箇所に，選択構造や反復構造を埋め込んだ構造も可能である。そこで，反復構造の開始位置と終了位置とを『ループ端』によって明示し，構造を見易くした右のような流れ図も利用される。

条件式

条件式では，条件が満たされる（真となる）場合には，値が true となり，条件が満たされない（偽となる）場合には，値が false となる。true 及び false は論理型(型名は boolean)のリテラルである。

条件式で用いられる**関係演算子**及び**等価演算子**を次に挙げる。

関係演算子	書式	意味
<	x < y	x が y より小さければ true，それ以外は false
>	x > y	x が y より大きければ true，それ以外は false
<=	x <= y	x が y より小さいか，両者が等しければ true，それ以外は false
>=	x >= y	x が y より大きいか，両者が等しければ true，それ以外は false

等価演算子	書式	意味
==	x == y	x と y とが等しければ true，それ以外は false
!=	x != y	x と y とが等しくなければ true，それ以外は false

また，複数の条件式を組み合わせるために用いられる**論理演算子**を次に挙げる。

論理演算子	書式	意味
&&	式 1 && 式 2	式 1 及び式 2 が共に true であれば true，それ以外は false
||	式 1 || 式 2	式 1 または式 2 のどちらか一方が true であれば true(両者が true の場合も含む)，それ以外は false
!	!式	式が true であれば false，式が false であれば true

for 文

Java 言語で 0 回以上の繰り返しのプログラムを実現するための文として，for 文と while 文とが用意されている。for 文の構文は次のようになる。

for （ 初期化処理; 継続条件式; 更新処理 ） 文

先ず，初期化処理を実行し，続いて継続条件式を評価する。ここで，継続条件式が true であれば，文を実行する。次に更新処理を実行した上で，再度，継続条件式を評価する，という繰り返しを行う。継続条件式が true でなければ，文を実行せず，for 文から抜け出す。最初から継続条件式が true でなければ，全く文を実行しないので，0 回以上の繰り返しと呼ばれる。

for 文は通常，特定の処理を決まった回数繰り返す目的で用いられる。

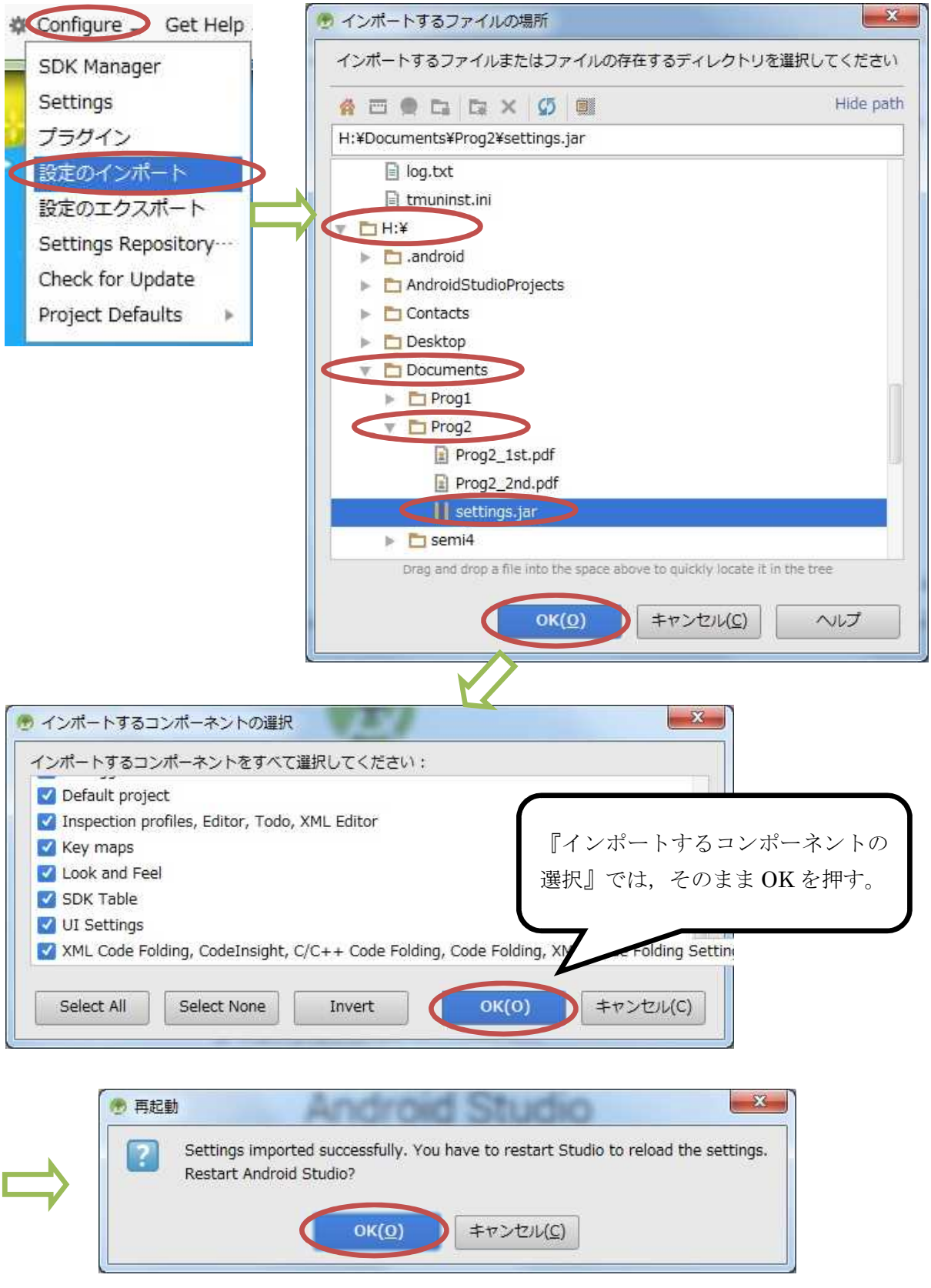
＊ 複数の処理を繰り返す場合は，{文 1　文 2 … 文 n}の様に中括弧でブロック化する。

なお，今後の教材で必要に応じて Java 言語の文法事項に触れていくが，プログラミング 1 非履修者は，プログラミング 1 の教材に目を通しておいて欲しい。

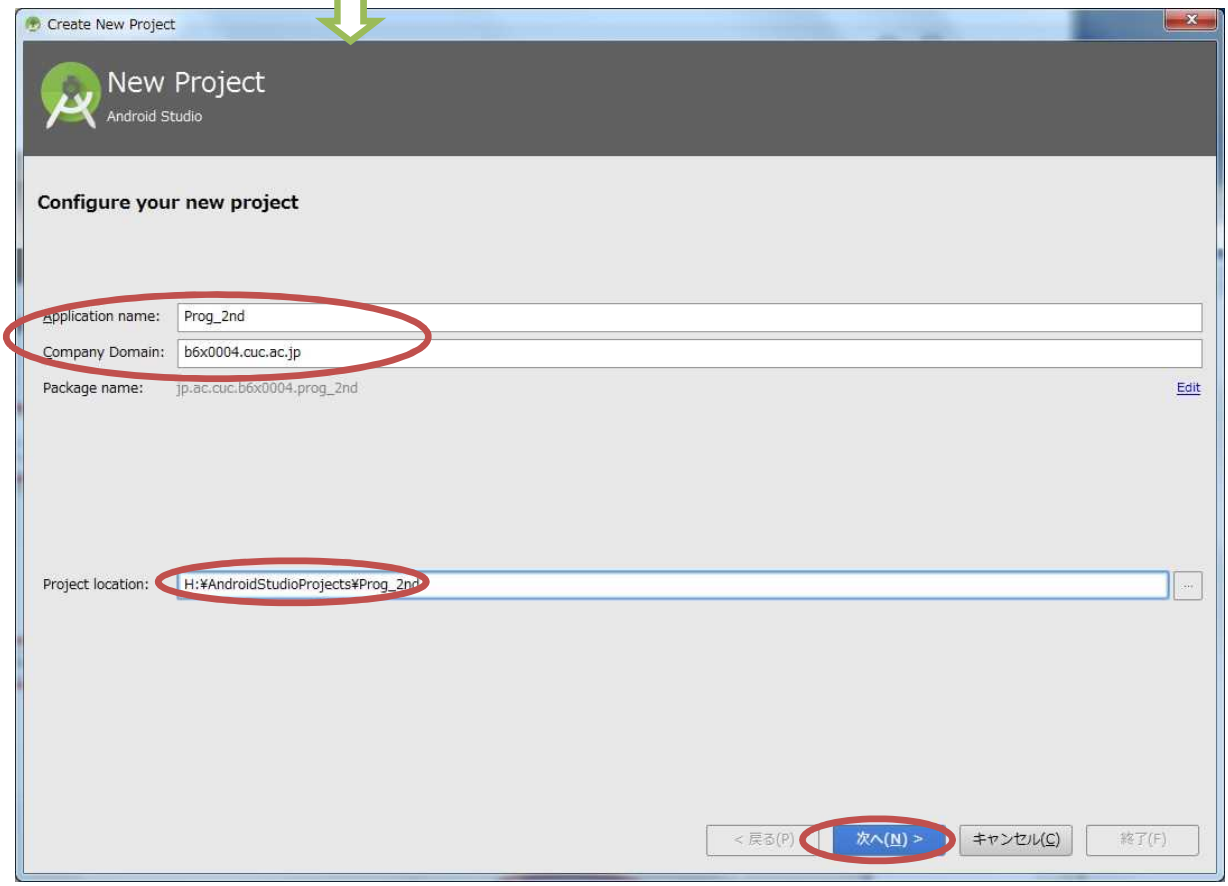
授業の準備

1）Android Studio の初期設定

Android Studio を起動し，『Configure』→『設定のインポート』を選択する。



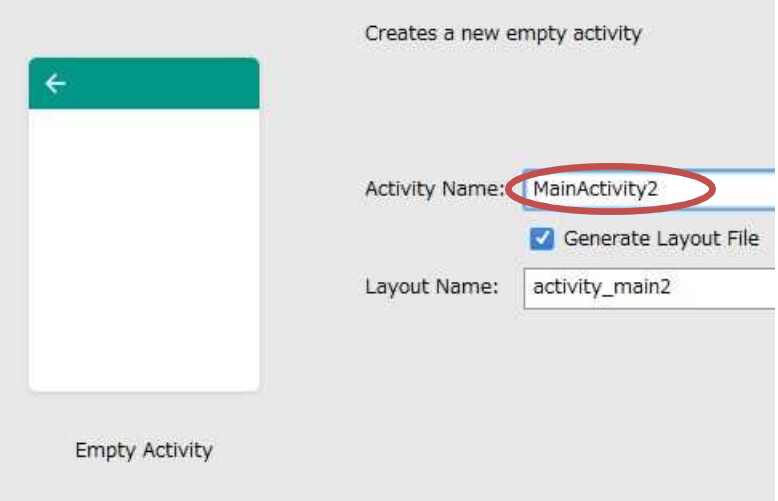
2）プロジェクトの新規作成



『Application name』(アプリ名) を「Prog_2nd」(先頭は大文字,「_」は下線),『Company Domain』を「b6a0xxx.cuc.ac.jp」に書き換え,『Project Location』の先頭の「C:¥Users¥ b6a0xxx」を『H:』に書き換えて,『次へ』ボタンを押す。

第 1 回と同様に『Minimum SDK』では『API 22』を選択する（第 1 回教材 p.7)。

『Activity name』は「MainActivity2」とする。



3）AVD の設定

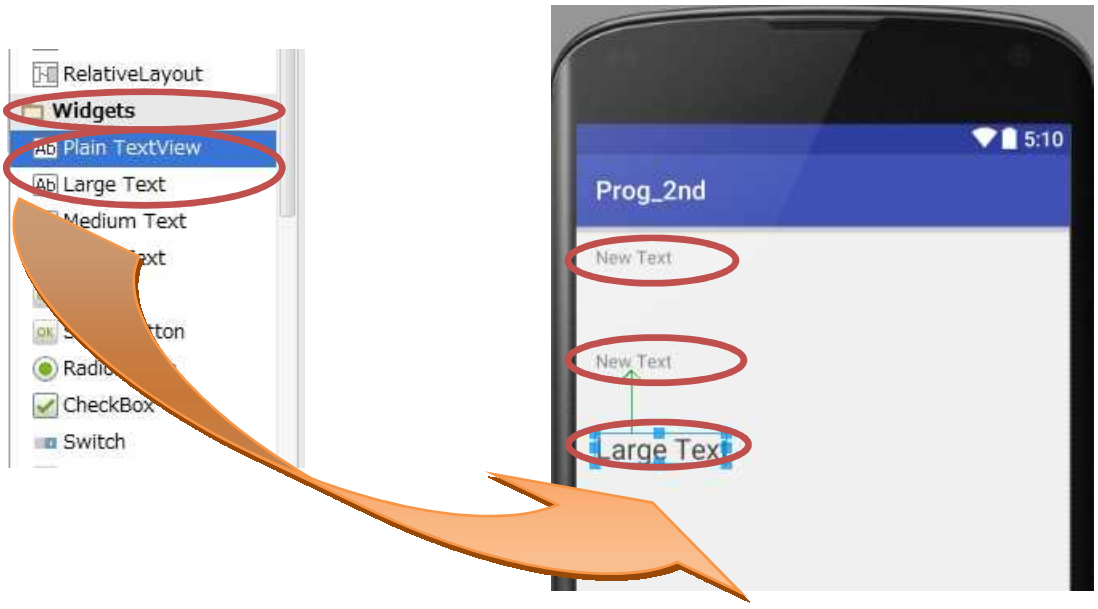
第 1 回の授業で作成した AVD の設定は H ドライブにあって残るが，SDK のシステムイメージは C ドライブにあるので，消失している。そこで，『Download』をクリックして，インストールし直す（第 2 回教材 p.4）。

課 題

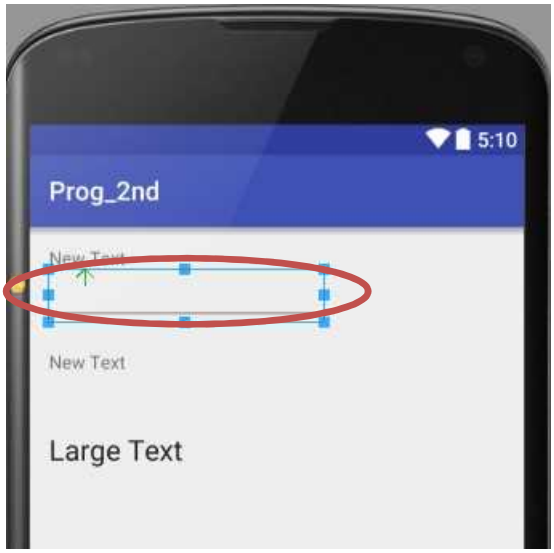
今回は，前回と同様なアプリを作成するが，for 文による反復構造を用いることで，Java 言語の基礎を学ぶ。

Android アプリの作成

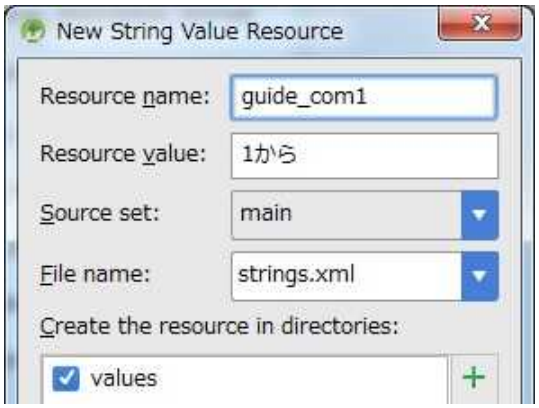
『activity_main2. xml』のタブを開く。先ず元から貼り付けられている「Hello World!」と表示されているテキストビューを選択し,『Delete』キーで削除する。続いて,『Palette』の『Widgets』から『Plain TextView』を 2 個，『Large Text』を 1 個ドラッグして貼り付ける。

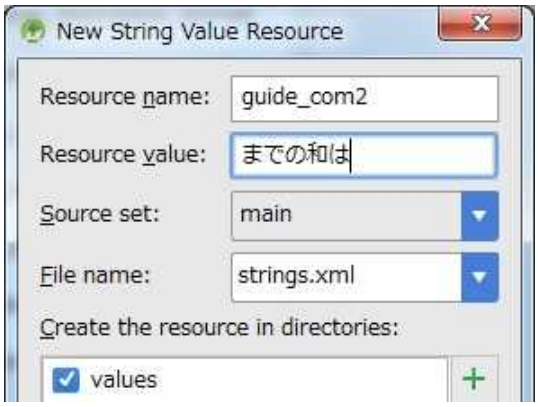


『Text Fields』から『Number (Decimal)』をドラッグして貼り付ける。
『Widgets』から『Button』をドラッグして，一番下に貼り付ける。



それぞれのテキストビューをダブルクリックし，『…』ボタンを押して出てきた『Resources』では『New Resource』 →『New string Value』を選択して，その値を変更していく。





texView　Resource name:　guide_com1
　　Resource value:　1 から

texView2　Resource name:　guide_com2
　　Resource value:　までの和は

『textView3』では「New Text」の文字列を削除する。

ボタンをダブルクリックして，表面の文字列を前回と同様に変更する（第 2 回教材 p.11）。
この段階で保存しておく。

『MainActivity2.java』のタブを開いて，ボタンをクリックすると計算結果が表示される機能を付け加える。

```
    private int sum = 0;
    private String res;
```

変数 sum の宣言及び初期化，変数 res の宣言

```
        Button btn = (Button)this.findViewById(R.id.button);

        btn.setOnClickListener(
                new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                        EditText et = (EditText)findViewById(R.id.editText);
                        int n = Integer.parseInt(et.getText().toString());

                        for(int i=1; i<=n; i++)
                            sum += i;

                        res = String.valueOf(sum);
                        TextView tv = (TextView)findViewById(R.id.textView3);
                        tv.setText(res);
                    }
                }
        );
```

Button ウィジェットのインスタンス及びイベントリスナー

【フィールド，Button ウィジェットのインスタンス及びイベントリスナーを付加】

このページの上の橙色の枠内にある内容を次ページの図の橙色の枠の位置（MainActivity2 クラスのフィールド），このページの上の赤色の枠内にある内容を次ページの図の赤色の枠の位置（onCreate メソッドの内部；中括弧『{』と『}』との間）に入力する。

activity_main2.xml × MainActivity2.java ×

```
1  package jp.ac.cuc.b6x0004.prog_2nd;
2
3  import android.os.Bundle;
4  import android.support.v7.app.AppCompatActivity;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.EditText;
8  import android.widget.TextView;
9
10 public class MainActivity2 extends AppCompatActivity {
11     private int sum = 0;
12     private String res;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main2);
18
19         Button btn = (Button)this.findViewById(R.id.button);
20
21         btn.setOnClickListener(
22                 new View.OnClickListener() {
23
24                     @Override
25                     public void onClick(View v) {
26                         EditText et = (EditText)findViewById(R.id.editText);
27                         int n = Integer.parseInt(et.getText().toString());
28
29                         for(int i=1; i<=n; i++)
30                             sum += i;
31
32                         res = String.valueOf(sum);
33                         TextView tv = (TextView)findViewById(R.id.textView3);
34                         tv.setText(res);
35                     }
36                 }
37         );
38     }
39 }
```

橙色の枠内を付加

赤色の枠内を付加

『保存』のアイコンをクリックして，MainActivity2. java を上書き保存し，実行ボタンをクリックする。

＊ エラーがある場合には，修正して保存してから実行する。

起動したアプリで１からいくつまでを足し合わせるのかを入力して，ボタンをクリックする。

上の様に動作確認が出来たら，完成となる。

提出物：

1）画面のレイアウト設定ファイル activity_main2. xml
2）アクティビティのソースファイル MainActivity2. java