Android アプリ開発演習：計算ゲーム(九九)

Android プログラミングの基礎を身につけるため、簡単な計算ゲームを以下の手順で作ってみましょう。

表 1 計算アプリのフェーズ概要

フェーズ	実装概要	参考レシピ	難易度
1	Android プロジェクトをウィザードに沿って作成して、まずは一番シンプルな構成で計算ゲームが作れることを目指す。 ● レイアウトファイルに TextView、EditText、Button を配置 ● TextView に「7 * 8 = 」などの形式で、九九の計算問題を表示する（乗数・被乗数はそれぞれ 1～9 からランダム） ● プレイヤーが EditText に回答を入力して Button をクリックすると、正誤を Toast で表示して次の問題を表示する	007, 019 073, 077 黒本 p.160	1
2	より複雑な画面レイアウトの実装を目指す。プレイヤーがより早く回答を入力できるようにテンキーを画面表示する。 ● フェーズ 1 の EditText を TextView に変更(入力表示用) ● TableLayout の中に 0～9 と Clear･Enter のボタンを表示して、ボタンから回答を入力できるようにする ● 正解をカウント（XX 問中、YY 問正解） ● 正答時/誤答時にそれぞれ異なる音を鳴らす	074 125 黒本 7 章	2
3	複数のアクティビティを連携させることを目指す。また、1 分間の得点を競うようゲーム内容を変更するために AsyncTask を用いたマルチスレッド処理を導入する。 ● ゲーム画面とは別に、タイトル画面と結果画面を追加する ● 起動時はタイトル画面を表示するよう変更、「スタート」ボタンクリックでゲーム画面に遷移する ● プレイヤーはゲーム画面で 1 分間問題を解き続ける。この際の回答数と正答数はシステム内でカウントしておく ● 1 分後に自動的に結果画面に遷移、結果を下記形式で表示する ➢ 正答数と回答数： 例） 5 / 6 ➢ 正答率： 例） 83.333 %	092 152 黒本 1 章 黒本 4 章	5
4	プリファレンスを用いたデータ保存の簡易実装を目指す。 ● 結果画面表示時、今回の正答数がハイスコア以上だったらハイスコアを更新する（ハイスコアはプリファレンスで管理する） ● タイトル画面にハイスコアを表示する	212 黒本 5 章	2

■フェーズ１

1) Android プロジェクトをウィザードに沿って作成します。

- プロジェクト名：keisan
- パッケージ名：com.example.keisan
- アクティビティのクラス名：MainActivity（作成時 Empty を選択）
- レイアウトファイル名：activity_main

2) レイアウトファイルを開き、図１のように TextView、EditText、Button を配置します。

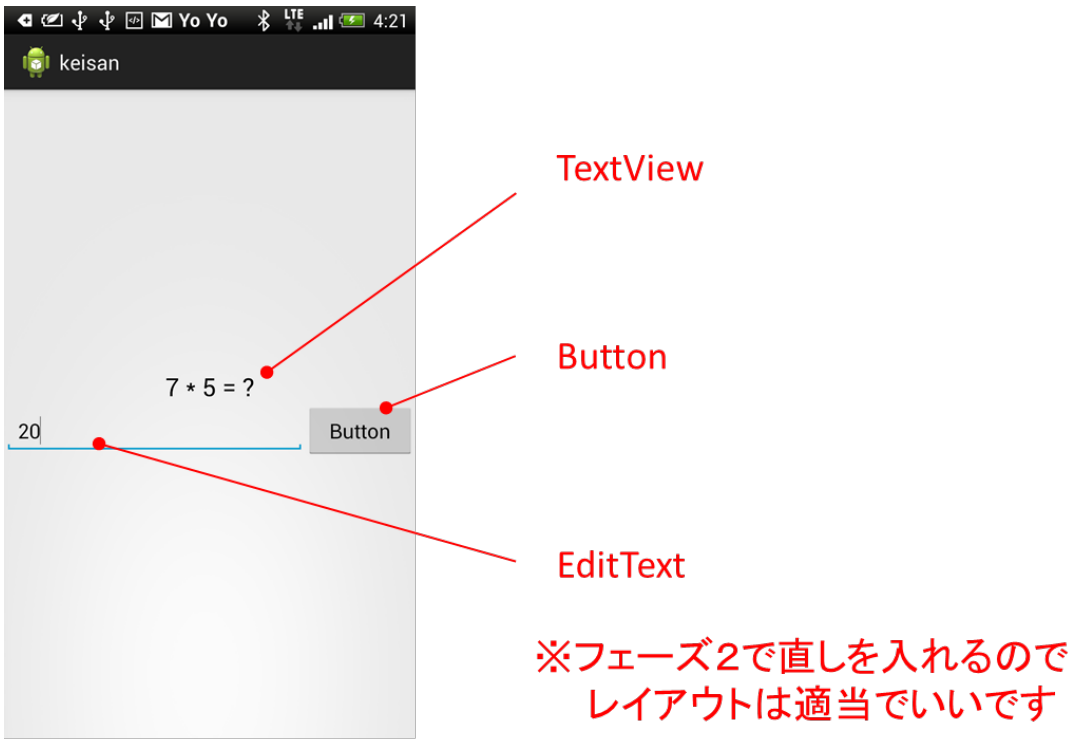

図 1 ゲーム画面イメージ(フェーズ１)

- TextView： 問題文の表示箇所です
- EditText： 回答の入力箇所です
- Button： 回答を確定するためのボタンです

3) MainActivity の実装を行います（表 2)。MainActivity は Activity クラスのサブクラスですが、ボタンをクリックしたときのリスナーとして自身を登録できるよう、OnClickListener を実装させます。

表 2 MainActivity の実装内容（フェーズ１）

メソッド名	新規/追加	実装内容
onCreate	追加	ウィザードで自動生成されたメソッド。既存処理の後ろに下記の内容を追加する。 1. TextView、EditText、Button を取得して、インスタンス変数に保持しておく 2. Button の setOnClickListener で自分自身を登録する 3. newQuestion メソッドを呼び出す
onClick	新規	OnClickListener の抽象メソッドをオーバーライトする。実装内容は下記の通り。 1. プレイヤーが EditText に入力した文字列を取得して、int 型のローカル変数 answer に変換する 2. answer と result を比較し、正解/不正解を判定する 3. 結果を Toast で表示する 4. newQuestion メソッドを呼び出す
newQuestion	新規	次の計算問題を準備するためのプライベートメソッド。実装内容は下記の通り。 1. TextView に「7 * 8 =」などの九九の問題文を表示する。この際、インスタンス変数 result に問題文の回答を保存しておく 2. EditText のテキストをクリアする(空文字列をセット)

4) 実装が終わったら動作確認を行い、問題なければ git リポジトリを作成してコミットしてください。以降、自分の変更が戻せなくて行き詰まらないように、各フェーズが終わる前でもまめにコミットしていってください。各フェーズが終わったときのコミット後は、該当フェーズが完了した箇所だという目印として、タグを打ちましょう。（分からない場合は講師か TA を呼んでください）

コマンド（以下の順序で実行）	解説
cd (ワークスペースの場所)/keisan	プロジェクトのルートディレクトリに移動
git init	リポジトリの初期化（初回のみ必要）
git add .	全てのファイルをコミット対象に設定
git commit -m “コメント”	コミット（コメントの部分は適宜書き換えること）
git tag phase_1_end	現在位置にタグを追加（フェーズ完了時のみ）

■フェーズ２

1) レイアウトを以下のように修正します（指定以外のレイアウト内容は自由）。

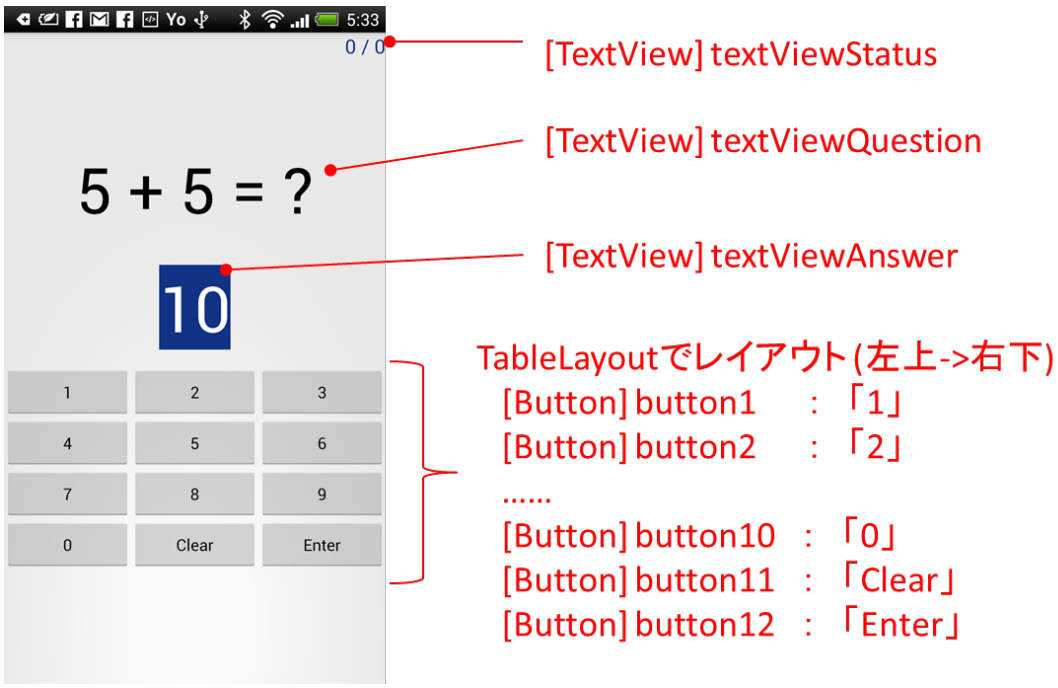

図 2 ゲーム画面イメージ(フェーズ２)

button1～button12 をクリックしたときに実行されるメソッドを、それぞれ以下の通りに設定します。

- button1～button10 ： public void inputNumber(View view)
- button11： public void inputClear(View view)
- button12： public void inputEnter(View view)

2) 上記で指定した３つのメソッドを実装します。それぞれの処理内容は以下の通りです。ここで示した内容に合わせて、onCreate も適宜修正してください。また、フェーズ１で作成した onClick は不要になるので、削除してください。

表 3 フェーズ２の実装メソッド

メソッド名	実装内容
inputNumber	押されたボタンのテキスト（button1 なら”1”、button10 なら”0”）を取得し、textViewAnswer の末尾に追加する。
inputClear	textViewAnswer に空文字列を設定する
inputEnter	フェーズ１の onClick と同内容。

3) inputEnter がクリックされた時、回答数(countAnswer)と正答数(countCorrect)を集計して、textViewStatus の表示が「正答数 / 回答数」の形式になるよう inputEnter に処理を追加します。

4) SoundPool を利用して、正答時・誤答時にそれぞれ異なる音が再生されるよう、MainActivity を変更してください。音源はフリー素材等から各自適当に調達してください。レシピ本を持っている人は、SoundPool のサンプルとして 125 が利用可能です。

5) 上記の動作確認が終わったら、git でコミットとタグ打ち(タグ名：phase_2_end)を行います。

■フェーズ３

1) タイトル画面(TitleActivity)と結果画面(ResultActivity)を追加した後の画面遷移を示します（図 3)。まず、タイトル画面と結果画面をウィザードで作成して、レイアウトを調整しています（画面デザイン等は自由に決めて構いません)。また、ランチャーからアプリを起動したさいにタイトル画面が表示されるよう、マニフェストファイルの設定を変更しておきます。

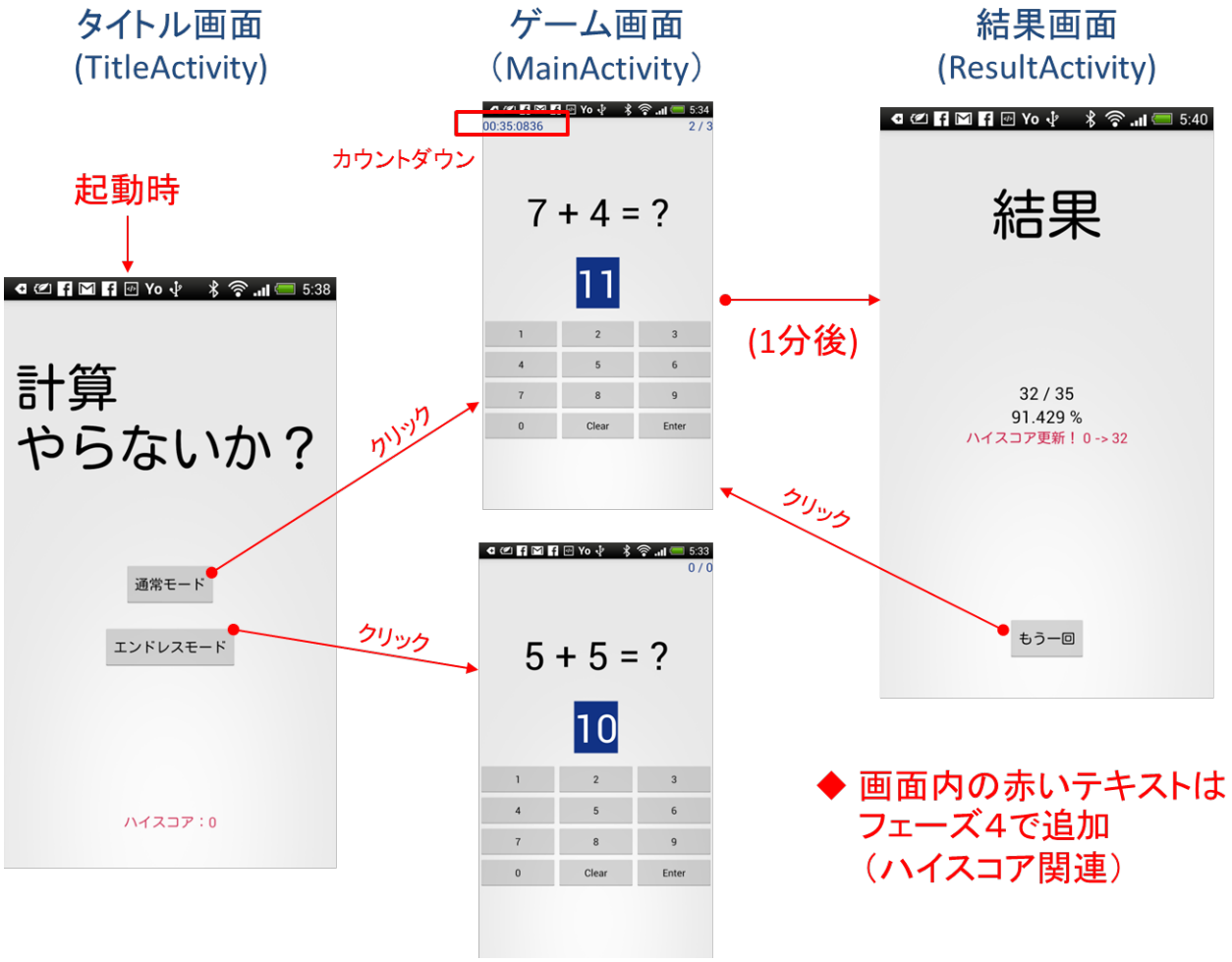

図 3 画面遷移と画面レイアウト(フェーズ３以降)

2) 図３での「(1 分後)」の画面遷移を実現するため、AsyncTask のサブクラス NormalModeTask を用いた非同期処理を実装します（AsyncTask に関しては、黒本の４章参照)。

表 4 NormalModeTask の実装メソッド

メソッド名	実装内容
doInBackground	「現在時刻 - 開始時刻 ＜ １分」である間、下記をループ実行 1. 100 ミリ秒程度スレッドをスリープする(CPU の負担軽減) 2. カウントダウンの表示文字列 remainText を作成 3. publishProgress(remainText)を用いて、UI スレッド経由でカウントダウンの表示を更新
onProgressUpdate	MainActivity に新規追加する timeUpdate メソッドを実行。 （timeUpdate への引数として、remainText を渡す）
onPostExecute	MainActivity に新規追加する gameEnd メソッドを実行。 gameEnd メソッド内では、ボタンクリックによる画面処理同様の方法で、結果画面へ画面遷移してください。

3) 図３での画面遷移を、レシピ本の 092 で例示されているように startActivityForResult メソッドと onActivityResult を用いて実装してください。それぞれのメソッド呼び出し時は、インテントを指定することができますので、これを介してデータ(extra)の送受信も行えます。

ただし、Android では端末の「戻るボタン」による画面遷移も考慮する必要があります。ここでは、結果画面から「戻るボタン」で戻った場合のゲーム画面の状態などを意識せずに済むよう、一旦タイトル画面を経由させます(次ページ、図４)。これは、下記実装で実現できます。

1. [遷移元] setResult メソッドで、画面遷移したい旨の resultCode を指定する
2. [遷移元] finish メソッドで、遷移元アクティビティを終了する
3. [TitleActivity] onActivityResult メソッドをオーバーライド。resultCode を元に条件分岐して、startActivityForResult メソッドで本来の画面遷移先を呼び出す

4) 画面遷移の作成が一通り終わったら、「戻るボタン」も含めた動作について全体的に整合性がとれたものになっているのかを確認してください。

5) 上記の動作確認が終わったら、git でコミットとタグ打ち(タグ名：phase_3_end)を行います。

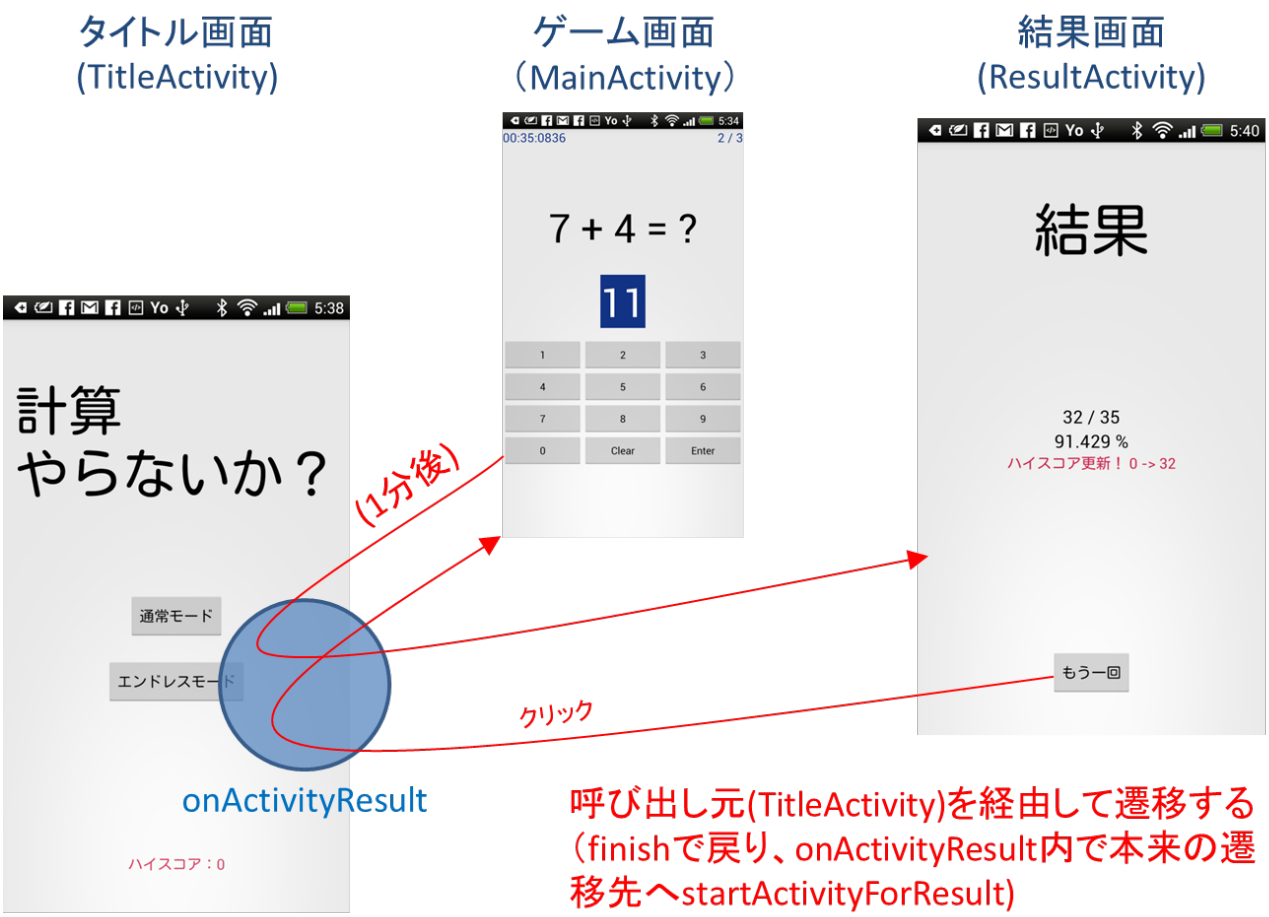

図 4 ゲーム画面<->結果画面での実際の画面遷移

■フェーズ4

1) ハイスコア情報をプリファレンスに記録するようにします（レシピ本の 212 参照）。結果画面を表示する際、下記手順でハイスコアの更新処理を行います。

   1. ハイスコアをプリファレンスから取得する（デフォルト値 0）
   2. 今回の正答数がハイスコアを超えている場合、プリファレンス内のハイスコアを今回の正答数で上書きする

2) ハイスコアを結果画面に表示します

3) ハイスコアをタイトル画面に表示します。ハイスコア更新後の結果画面から「戻るボタン」で戻った場合にも、きちんと更新後のハイスコアが表示されるよう工夫してみてください。

4) 上記の動作確認が終わったら、git でコミットとタグ打ち(タグ名：phase_4_end)を行います。

■オプション

ゲームとしての完成度を上げるため、更に興味に応じた作り込みを加えていきましょう。
以下に、作り込みの例を示します。

表 5 計算アプリの更なる作り込み例

機能追加	作業内容	難易度
問題の種類を増やす	● 加減乗除もランダム生成する ● 乗数・被乗数の範囲を「0～9」に変更する ● (ゼロ除算、負数の回答の対応を行う)	2
端末の向きが変わっても問題ないにする	● 向きを固定する or ライフサイクル処理追加	2
何か絵的に洗練させる	● アプリのアイコンを作り込む ● ボタンを画像に置き換える ● ハイスコアの文字列を点滅させる	1
結果画面に結果一覧を表示する	● ゲーム画面でのゲームプレイ中、回答ごとにその記録(回答内容+正答内容)の配列を履歴に残す ● 結果画面に ListView を追加し、ArrayListAdapter を用いて履歴を一覧表示する ● ListView に表示する項目のレイアウトを　自分の好みにカスタマイズする	3
オンラインランキング	● 下記の API を備えた Web サーバを用意する 1.　ハイスコア送信 API 2.　上位 N 件のハイスコアランキング取得 API ● 結果画面で 1. の API を利用して、ハイスコアを送信する ● ランキング画面を新規作成する。ランキング画面では 2. の API を利用してハイスコアランキングを取得、LlistView で表示する ● タイトル画面にボタンを 1 つ追加し、そこからランキング画面に遷移できるようにする	5
MainActivity の中身を別のゲームにする	● SDK のサンプルに付属の「snake」に差し替える ● テトリスに差し替える	?