平成２７年度

卒業論文

題目	Androidアプリケーション「Fortune Game」の 企画・開発及び評価

担当教員 （自署）		印

学籍番号　201214070

氏　名　長田　竜太朗

広島経済大学

広島経済大学

要旨

この卒業論文は、運試しアプリケーション「Fortune Game」開発についてのものです。まず第１章では、私が開発したアプリケーションの本体、スマートフォンについて説明しています。スマートフォンとは、従来の携帯電話の機能にアプリケーションが追加された多機能携帯電話のことです。私はそのスマートフォンの普及率や一人当たりのアプリケーション利用個数を調べ、説明しました。第２章では「Fortune Game」について説明しています。第２章第１節では、アプリケーションを開発するきっかけになった説明しました。続いて第２項では、アプリケーションを開発するために必要な開発環境「Eclipse®」と「Android SDK®」についての説明をしました。第３章では、プログラミングの概要について説明しています。第３章第１節では、第２節から用語が多く出てくるため用語の説明に引用を用いながら書きました。第２節では先ほどの説明を理解したと踏まえて、MainActivity.java で実際に書いたコードを使いながら説明しています。第４章では、第３章で書いたコードを実装して、Android アプリケーション「Fortune Game」がどのように動いているかの説明をフローチャートと Android SDK®の画面を使用して説明しています。第５章では、今回開発した「Fortune Game」についての評価をもらうため、アンケート調査を行いました。第５章第１項では、アンケートを作成するにあたって、何を基準に書けばいいのかを説明し、第２項ではアンケートの内容の説明と評価の方法について述べ、第３節では集計結果について円グラフを用いながら書きました。第６章は最終章で、今回開発した Android アプリケーション「Fortune Game」について行ったアンケート調査の結果の考察、アンケートで指摘されたコードの追加、今度この開発をもとにどのような課題があげられるかをまとめました。

目次

第1章 はじめに

今の私たちにとってスマートフォンとはなくてはならないものです。スマートフォンとは、通常の携帯電話の機能に付け加えて LINE や Twitter などの SNS、YouTube や USTREAM などの動画投稿サイト、モンスターストライクやパズル＆ドラゴンなどのアプリケーションゲームが、１つの機種で出来る多機能携帯電話のため若者に人気です。総務省が発表した平成２６年度の１０代から６０代のモバイル機器（スマートフォン、フィーチャーフォン)、タブレット利用率〔１〕では、１０代は 68.6%、２０代は 94.1%、３０代は 82.2%と数字で見ても現代の若者のみならず、１０代から５０代まで幅広い人が生活に必要不可欠な存在となっているのです（図の 1.1 に示しています)。

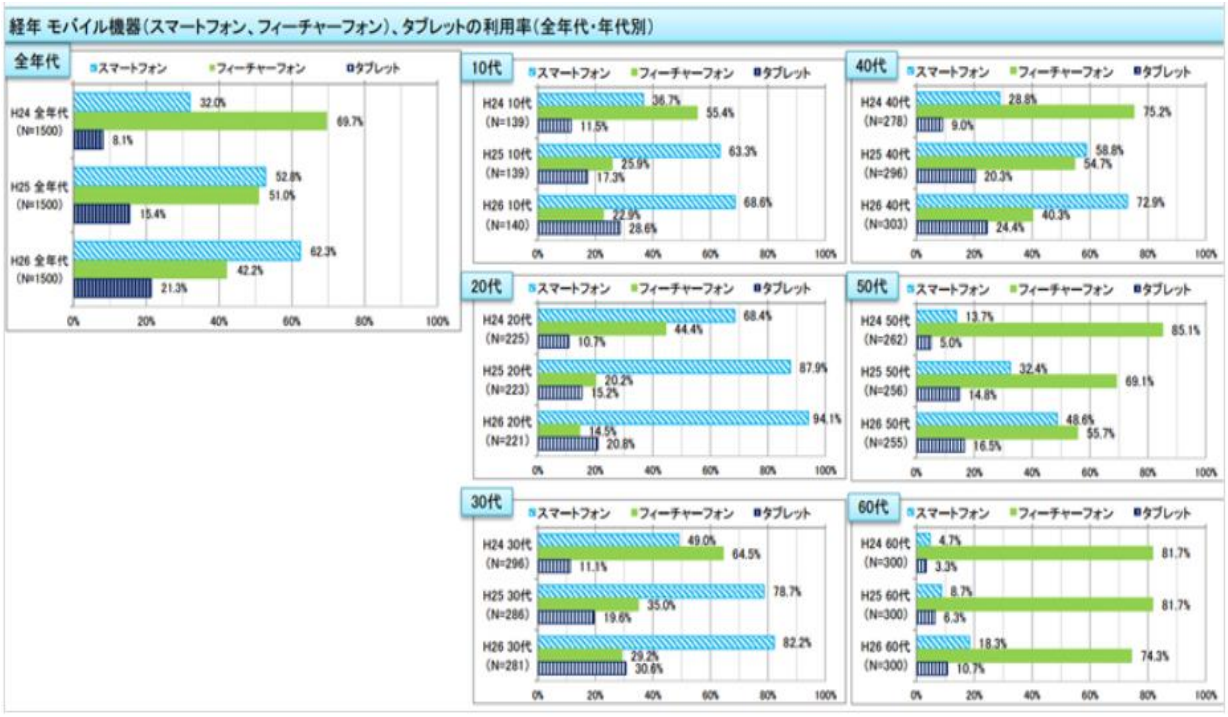


図 1.1　マイナビニュースの調べによる各年代のスマートフォン所持率

私はその生活の一部に取り込まれたスマートフォンの中にあるアプリケーションの中で、ゲームコンテンツに着目しました。何故このコンテンツに着目したかというと、先ほどのスマートフォン所持率を見たときに、仕事効率を重視したアプリケーションや旅行やドライブなどで役立つアプリケーションなど多種多様なカテゴリの中で、どのカテゴリが１番人気あるか調べたためです。調べた結果、セガネットワークスのインターネット調査

で「各ジャンルアプリ利用者における平均アプリ利用個数」で結果を見るとゲームアプリが平均３．８９個と最も高く、「性年代別のゲームアプリの利用状況」も１５～２４歳までの男女は減少しているものの平均利用個数も 3.92 個となっていたためです〔２〕。図の 1.2 に引用した結果があります。

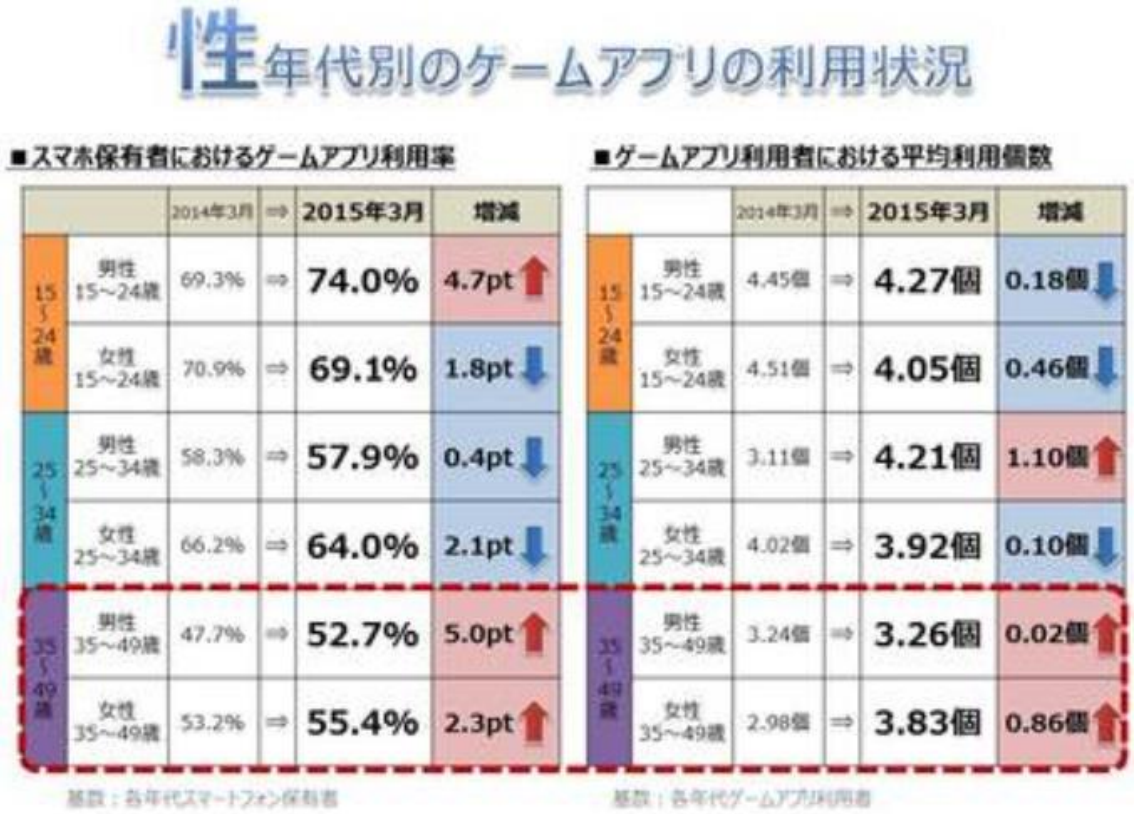


図 1.2　セガネットワークスによる性年代別のゲームアプリの利用状況

この結果からゲームのアプリケーションは、１０代から５０代まで幅広い年代に人気があり且つ利用率が高く多いということが分かったため、この幅広い層をターゲットにした開発をしようと考えました。

第2章 アプリケーションについて

第1節 アプリケーションを開発するきっかけ

Android アプリケーション「Fortune Game」を開発したきっかけは、類似アプリケーションをしている時でした。その類似アプリケーションとは、地雷除去において爆発するかどうかの運試しをするものです。

第2節 アプリケーションの計画

開発したアプリケーションはマインスイーパのような地雷以外のマスをタップしてクリアを目指すゲームのアプリケーションです。ステージは２×２、３×３、４×４と複数の画面を用意しており、初めのステージが出来ずとも次のステージへ進めるようになっています。ここまではマインスイーパと同じですが、違うところが１つあります。それは数字の有無です。マインスイーパでは、自分が選択したマスに隣接する８マスのいずれかに地雷がある場合その個数が表示されますが､私が開発した Fortune Game にはそれがなく、自分の運が試されるようになっています。これはタイトルの通り、運試しという意味を込めて数字を入れていません。このゲームの対象とする層は特になく、どの世代でも遊べるように開発しています。このアプリケーションの計画は前述の通り、既存のものにプラスアルファしたものですが実際に開発し始めると爆弾の画像や爆発音はどのように入手するのか、最初の画面から次の画面へ進む際に使う Sub1 という変数が分からないなど苦戦しました。また Sub1 を挿入して次の画面へ進んでもボタンをタッチすれば前の画面へ戻る、爆発音が鳴らず散々悩みました（Sub1 については 4.2 節参照)。

第3節 開発環境

「Fortune Game」アプリを開発するために使った開発環境は Eclipse®です。オープンソースの統合開発環境（様々なものを１つにまとめたソフトウェアの開発環境）として２００１年に配布された Eclipse®は、プラグインによる機能拡張に伴い、開発環境の共通

プラットホームとしてスタンダートになった感があります〔３〕。

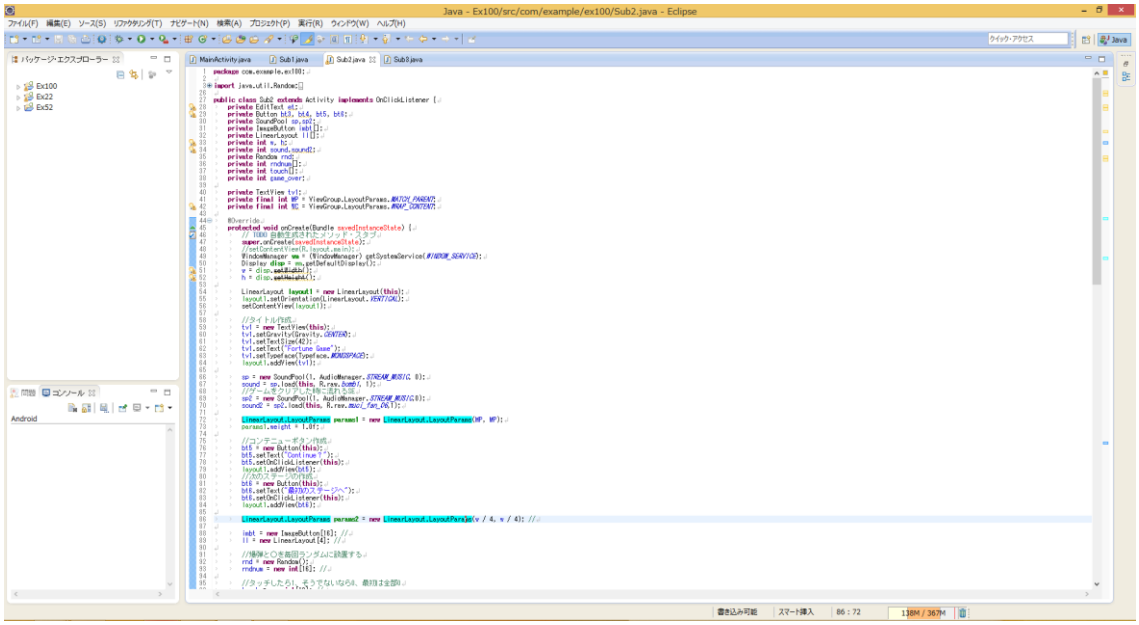


図 1.3　Eclipse の画面。コードは今回開発した「Fortune Game」のもの。

言語は Java や C など計２５言語に対応していますが、多くの人が Java 開発のために使用しています。また、開発する際にもう１つ Android SDK®を使用しました。これはパーソナルコンピュータで Android アプリケーションを開発する際に、端末を再現するエミュレータです。Google 社が無償で公開しており、Windows 版、Mac OS X 版、Linux 版があり先ほどの Eclipse と併用しました。この Android SDK は Android OS を搭載したスマートフォンやタブレット端末で動作するプログラムを開発するために必要なソフトウェアをひとまとめにしたパッケージで、コンパイラやデバッガ、ライブラリ、デバイスドライバ、ドキュメント、サンプルコード、パソコン上で端末を再現するエミュレータなどで構築されています。

第3章　アプリケーションの実装

ここでは今回私が開発した Android アプリケーション「Fortune Game」がどのように動いているか説明します。まずこのアプリケーションの処理の流れをフローチャートにして図 1.4 として示しました。

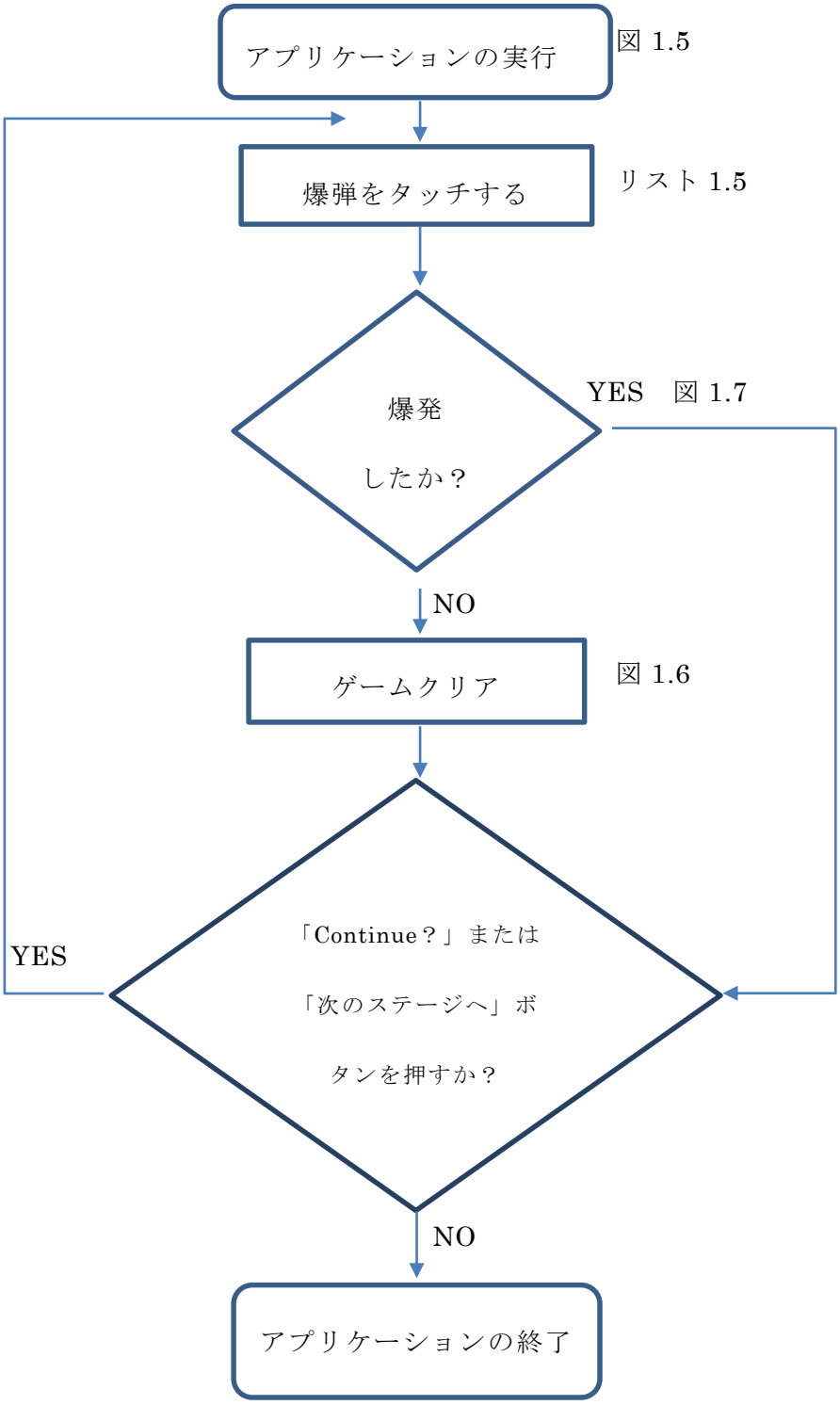


図 1.4　「Fortune Game」アプリケーションの処理の流れ

このフローチャートをもとに説明すると、アプリケーションを実行し、爆弾をタッチします。そこで爆弾が爆発した場合、「Continue？」または「次のステージへ」ボタンを押すか？の選択肢に移動し、コンテニューや次のステージに行く場合は YES のほうに行き、もう一度爆弾をタッチするところから始まります。爆発したか？の選択肢で NO の場合

はゲームクリアとなり、先ほどと同様「Continue？」または「次のステージへ」ボタンを押すか？の選択肢に移動します。そこで NO の場合、ゲームの終了となり、アプリケーションも終了します。これを画面で詳細に説明していきます。まず、アプリケーションを実行し、最上部に「Fortune Game」のタイトルが表示されます。その下に「Continue？」と「次のステージへ」のボタンが置かれ、さらにその下には２×２の計４つのマスと爆弾があります（図の 1.5 参照)。

図 1.5　アプリケーション起動時の画面

はじめは全てのボタンが動くようになっており、そのまま次のステージへも行けることができます。ゲームの順序としてまず、爆弾をタッチすると、〇か爆弾が爆発する画面になります。最初に〇が出てきた場合の説明をすると、次の〇を探すか「Conglatulations！」が出てくる２つの場合があります。下方部に

「Conglatulations！」が出てきた場合、そのステージでのゲームはクリアになります。クリアした場合の画面は、図の 1.6 で示しています。

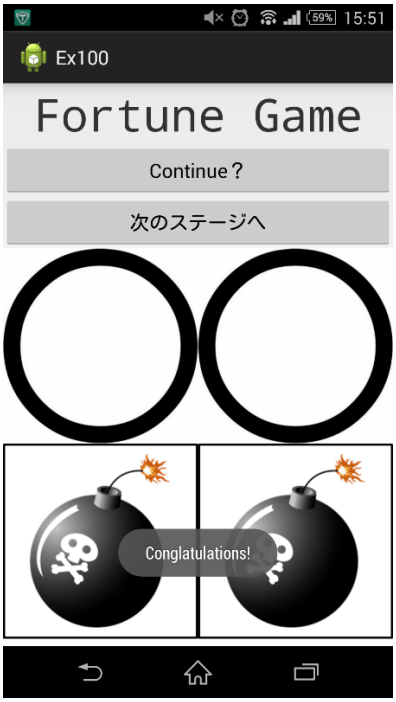


図 1.6　ゲームクリア画面

反対に爆弾を押して爆発した場合、爆発した様子が１０コマで表現され、先ほどの「Conglatulations！」同様、「Game Over！」と「Please continue！」が下方部に表示されます。図の 1.7 でそのコメントの表示を示しています。他の爆弾を押しても反応せず「Continue？」ボタンか「次のステージへ」ボタンを押す以外は全ての反応が無効となり、動かすことができません。

図 1.7　Game Over した際のコメントの画面

（左が Game Over、右が Please continue！）

クリアして次の画面へ進むと Main Activity から Sub1 のタブに切り替わるため、画面が切り替わり３×３マスと増えています。図の 1.8 に画面を置いています。

図 1.8　2 つ目の画面

ゲーム的にはここから少し難しくなり、このアプリケーションのコンセプトである「運」の要素が強いゲームになっています。実際に同じ田中ゼミに所属しているゼミの友人達にしてもらうと、１回でクリアできた人は９人中２人しかいませんでした。

第4章 プログラム概要

第1節 用語説明

ここでは、次のプログラミング説明の際に使用する用語について説明します。IT 用語辞典〔４〕を参考にしています。

はじめに、アクティビティについてです。アクティビティとは今回開発したアプリケーションで最も重要な画面のことです。これがなければ画面が見えず、途中の試作が運用できません。次にメンバ変数です。メンバ変数とはインスタンス変数とも言われ、オブジェクト指向プログラミング言語でオブジェクトの個々のインスタンス毎に固有の変数です。クラスとは、オブジェクト指向プログラミングにおいて、データとその操作手順であるメソッドをまとめたオブジェクトの雛型を定義したもので、これを定義することで同種のオブジェクトをまとめて扱うことができるようになります。ストリームとは、IT の分野では連続したデータの流れや、データの送受信や処理を連続的に行うことなどを意味します。インターフェースとは、二つのものが接続・接触する箇所や、両者の間で情報や信号などをやりとりするための手順や規約を定めたものを意味し、完全限定名とはパッケージ名を省略しない形での名前のことです。インスタンスとは、あらかじめ定義されたコンピュータプログラムやデータ構造などを、メインメモリ上に展開して処理・実行できる状態にしたものです。最後に intent とは、アクティビティからほかの指定したアクティビティへ届けたい情報などをやり取りするための存在です。これらを踏まえたうえで、次のプログラミングの説明に入ります。

第2節　プログラム概要

ここでは Fortune Game のプログラムについて説明していきます。まず、import 関数を使用し、android app（アプリ）、content（コンテンツ）、graphics（グラフィック）、media（メディア）、os（オペレーティングシステム）、view（画面）、widget（ウィジェット）など合計 19 個を入れ、特定のパッケージ内のクラスやインターフェースを完全限定名以外でも呼び出せるようにしました。次に、Private というメンバ変数を

使用し、Button、SoundPool、ImageButton、LinearLayout、Random、Text View を sp、imbt などに略してボタンや音、ビュー配置などの、レイアウトに必要なクラスを用意しました（リスト 1.1）。

```
        private Button bt, bt2;
        private SoundPool sp,sp2;
        private ImageButton imbt[];
        private LinearLayout ll[];
        private int w, h;
        private int sound,sound2;
        private Random rnd;
        private int rndnum[];
        private int touch[];
        private int game_over;
        private TextView tv1, tv2, tv3, tv4;
        private final int MP = ViewGroup.LayoutParams.MATCH_PARENT;
        private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
```

リスト 1.1　private メンバ変数によって省略したクラス名

以上の変数を用意し、ここから画面作成になります。まず、縦または横方向にビューを配置していくために LinearLayout クラスを使用し、WindowManager でウィンドウの飾りつけや位置関係の、大きさ、操作方法を管理しました。また、ディスプレイの情報を取得するために、Display クラスのインスタンスから getDefaultDisplay()を呼び出しました。次に、本アプリケーションのタイトル Fortune Game を入れるために TextView から、setGravity、setTextSize、setText、setTypeface を用意し、文字位置の変更やテキストサイズの変更、「Fortune Game」テキストの挿入、フォントスタイルの指定を行いました（リスト 1.2）。

```
//タイトル作成
tv1 = new TextView(this);
tv1.setGravity(Gravity.CENTER);
tv1.setTextSize(42);
tv1.setText("Fortune Game");
tv1.setTypeface(Typeface.MONOSPACE);
    layout.addView(tv1);
```

リスト 1.2　タイトル「Fortune Game」の表示

コンテニューボタンと次のステージに進むためのボタンの作成もします。Button クラスと setText を用意し「Continue?」、「次のステージへ」のテキスト挿入と setOnClickListener を使い対象をなるボタンがイベントを受け取れるように設定しました（リスト 1.3)。

```
//コンテニューボタン作成
bt = new Button(this);
bt.setText("Continue？");
bt.setOnClickListener(this);
layout.addView(bt);
//次のステージの作成
bt2 = new Button(this);
bt2.setText("次のステージへ");
bt2.setOnClickListener(this);
                    layout.addView(bt2);
```

リスト 1.3　「Continue？」「次のステージへ」ボタンの作成

ここから本アプリケーションのメイン部分を作成していきます。最初は爆弾をタップし爆発した場合に使う音の作成です。パッケージエクスプローラーにある raw フォルダから

mp3 を取り出すために SoundPool を使用し、AudioManager で音の種類の指定、音量調整を行った後、今回使用した bomb1.mp3 を入れました。コードとしては、Private 変数で指定した sp クラスを使用し、AndroidManager クラスで定義された「STREAM_MUSIC」を呼び出しました（リスト 1.4 参照）。この bomb1.mp3 に限らず、今回開発したアプリケーションの音楽は全て著作権フリーの音楽サイトからダウンロードして使用しています（後述記載）。

```
        //音の設定
        //爆弾が爆発した時のSE
        sp = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
                                sound = sp.load(this, R.raw.bomb1, 1);
```

リスト 1.4　音の設定

次に params という、オブジェクトの実行に必要な設定値であるパラメータを指定するタグを用意し、縦と横を何マスにするかの指定をしました。次に爆弾の絵を表示し、クリックすると応答するコントロール ImageButton を用意し、最初のステージのため、[] 内を 4 に、LinearLayout の [] 内を 2 に指定しました。次にランダム変数を使用して爆弾のある位置とそうでない位置をランダムに設置し、[] 内を 4 と設定とました。次に爆弾をタッチした後、もう一度タッチしても反応がなくなるようにするため、初期値と爆弾をタッチしない場合は 0、爆弾をタッチすれば 1 と変換されるようにしました。このプログラムの例をリスト 1.5 に示しました。

```
    touch = new int[4];
        For (int i = 0; i < 2; i++){
            Touch[ i ] = 0;
```

リスト 1.5　爆弾をタッチしたときに反応するかのプログラム

また、爆弾が爆発した場合ほかの爆弾をタッチしても反応が無くなるようにするため Game Over の場合を 0 とし、0 の場合はほかをタッチしても動くようにし、1 になった場合は後述記載とします。これはリスト 1.6 で示しています。

```
    game_over = new int[4];
    for(int i = 0; i < 2 ; i ++){
      ll[i] = new LinearLayout(this);
       for(int j = 0 ; j < 2 ; j ++){
           Rndnum[i * 1 +j] = rnd.nextInt(2);
           Imbt[i * 2 + j ] =new ImageButton(this);
           Imbt[i * 2 + j ] . setLayoutParams(params2);
           Imbt[i * 2 + j ] . setBackgroundResource(R.drawble.c0);
           Ll[i].addView(imbt[i*2+j]);
           Imbt[ i * 2 + j ] . setOnClickListener(this);
       }
           Layout.addView(ll[i]);
   }
}
```

リスト 1.6　爆弾をタッチしても何も起こらないようにするプログラム

今回の開発では自分のオリジナルさを出すためにクリアしなくてもコンテニューボタンを押せば次に進めるようにし、その際に書いたコードの説明です。If 関数を使い、はいの場合の選択肢といいえの場合の選択肢を用意しました。まず AlertDialog クラスから setTitle に「Continue？」、setMessage に「やり直しますか？」というメッセージを表示し、setPositiveButton に「はい」と「いいえ」を入力したプログラムをリスト 1.7 に載せました。

```
if (v == bt) {
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
                alertDialogBuilder.setTitle("Continue?");
                        alertDialogBuilder.setMessage("やり直しますか？");
```

リスト 1.7　コンテニューボタンの作成とメッセージの表示

「はい」の場合 makeText から「Let's Go！」と表示され For 文で Random、ImageButton を呼び出し、初期画面の状態に戻ります。これはリスト 1.8 に示しています。

```
        alertDialogBuilder.setPositiveButton("はい", new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
        Toast.makeText(MainActivity.this, "Let's
Go!",Toast.LENGTH_LONG).show();
                        for (int i = 0; i < 2; i++) { //
                                for (int j = 0; j < 2; j++) { //
                                 rndnum[i * 2 + j] = rnd.nextInt(2); //
                                imbt[i * 2 +
j].setBackgroundResource(R.drawable.c0); //
                                        }
                                }
                        //タッチしたら1、そうでないなら0、最初は全部0
                        touch = new int[4]; //
                                for (int i = 0; i < 2; i++) { //
                                touch[i] = 0;
                        }
```

```
                    //game_overが0ならまだOK、1なら終了
                        game_over = 0;
                        }
                    });
```

リスト 1.8　コンテニューボタンから「はい」を選択した場合のプログラム

「いいえ」をタッチした場合「GameOver！」と「Please Continue！」表示され、「Continue？」か「次のステージへ」のボタンを押さない限り、爆弾をタッチしても動かない状態にしました。こちらもリスト 1.9 として示しています。

```
    //いいえ　の場合
alertDialogBuilder.setNegativeButton("いいえ", new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
        // TODO 自動生成されたメソッド・スタブ
            Toast.makeText(MainActivity.this, "Game Over.",
Toast.LENGTH_LONG).show();
        }
});
alertDialogBuilder.setCancelable(true);
AlertDialog alertDialog = alertDialogBuilder.create();
alertDialog.show();
} else {
    if (game_over == 0) {
        for (int i = 0; i < imbt.length; i++) {
            if (v == imbt[i]) {
                if (rndnum[i] == 1) {
```

```
                    imbt[i].setBackgroundResource(R.drawable.c8);
                    imbt[i].setBackgroundResource(R.drawable.animation);
                      AnimationDrawable frameAnimation = (AnimationDrawable)
imbt[i].getBackground();
                    frameAnimation.start();
                    sp.play(sound, 50.0f, 50.0f, 0, 0, 1.0f);
                    game_over = 1;
                    Toast.makeText(this, "Game Over!",
Toast.LENGTH_LONG).show();
  } else {
                Toast.makeText(this, "Please continue!",
Toast.LENGTH_LONG).show();
  }
```

リスト 1.9　コンテニューボタンから「いいえ」を選択した場合のプログラム

「いいえ」を選択した場合これで終わりではなく、先ほど GameOver の 1 を押した場合、反応しなくなると記述しましたが、これはゲームのクリアとなるため maketext に「GameOver」ではなく「Conglatulations！」と書き、ステージのクリアとなるためそれをリスト 1.10 に示しました。

```
  } else {
        imbt[i].setBackgroundResource(R.drawable.maru);
        touch[i] = 1;
        game_over = 1;
        for (int j = 0; j < 2; j++) { //
              if (rndnum[j] != 1 && touch[j] == 0) {
                    game_over = 0;
```

```
                }
        }
        if (game_over == 1) {
                Toast.makeText(this, "Conglatulations!",
Toast.LENGTH_LONG).show();
                sp2.play(sound, 50.0f, 50.0f, 0, 0, 1.0f);
        }
         }
      }
  }
```

リスト 1.10　ゲームをクリアした際出てくる「Conglatulations!」の表示

最後に「次のステージへ」のボタンを押した際、「Sub1」の画面に移るために Intent を作成、「Sub1」の画面を呼び出せるようにし、リスト 1.11 に示しました。

```
                if (v == bt2) {
                //１つ目の画面から２つ目の画面であるクラスSub1の呼び出し
                Intent intent = new Intent(this, Sub1.class);
                startActivity(intent);
                }
  }
```

リスト 1.11　次の画面の呼び出しを売るためのプログラム

ここから Sub1 の画面になりますが、コードの内容は縦横の長さが 2 から 3 への変更をリスト 1.12、new int によって爆弾の数をマスの数に合うように変更をリスト 1.13、Intent によって Sub2 から Sub3 に変更するリスト 1.14 以外は Main Activity と同じのため、説明は省きます。

```
   LinearLayout.LayoutParams params2 = new LinearLayout.LayoutParams(w / 3, w /
3);
```

リスト 1.12　画面のマスを増やすためのコード

```
   imbt = new ImageButton[9];
```

リスト 1.13　爆弾の数をマスの数に合うように設定

```
if (v == bt4) {
//一つ目の画面から２つ目の画面であるクラスSub1の呼び出し
Intent intent = new Intent(this, Sub2.class);
startActivity(intent);
   }
```

リスト 1.14　２つ目の画面から 3 つ目の画面へ移動するプログラム

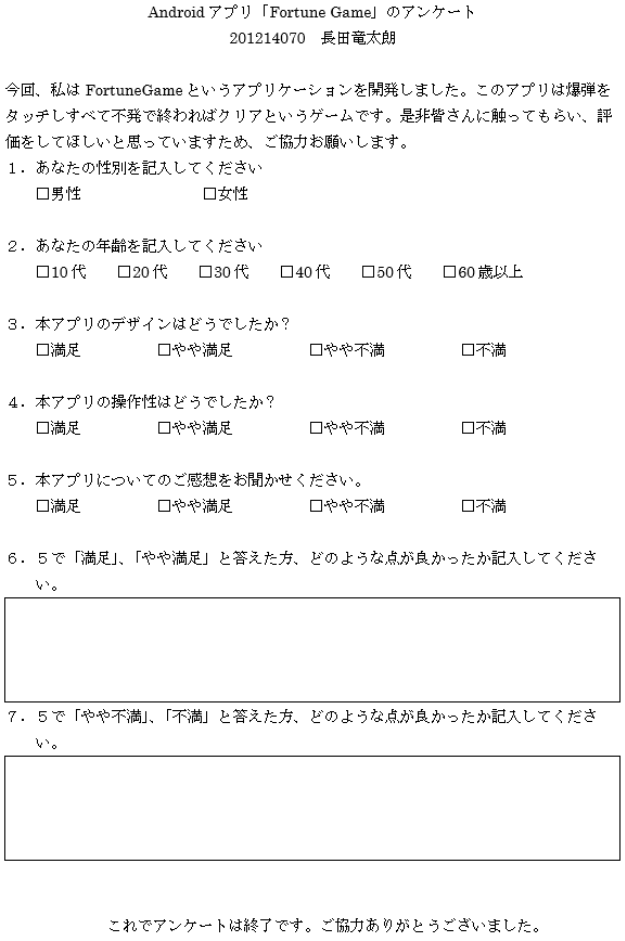
第5章 評価

第1節 アンケートの各種調査法

私が開発した Android アプリケーション「Fortune Game」のアンケートを説明する前に、まずアンケートについて説明します〔５〕。まず、アンケートとは質問に対して回答者が答えを記入やチェックをして最後にそのデータを集計するものです。回答方法には４種類あり一つ目は、質問に対して文章や数値で自由に答えてもらう「自由回答法」。２つ目はプリコードという、予想される回答内容を選択肢としてあらかじめ用意しておき、それぞれにコード No.をつけておき、該当するコード No.で回答してもらう「プリコード回答法」。３つ目は、２つの回答選択肢から選んでもらう「二項選択法」、３つ以上の回答選択肢から選んでもらう「他項選択法」、それらをまとめた「SA 回答法」。最後に、回答の個数に制限をつけないで選んでもらう方法の「無制限法」、回答の個数に制限をつけて選んでもらう「制限法」、それらをまとめた MA 回答法があります。今回アンケート調査に使用したものは SA 回答法の二項選択法と多項選択法、自由回答法を用いて作成しました。この 2 つの回答法のメリットは、まず自由回答法は対象者の意見を最も取り入れやすくなっているため、対象者がそのアプリケーションにどのように感じたのか詳細な情報を得ることができます。次に SA 回答法は、アンケート配布者にとって集計が容易なため、手間がかからないです。そのため、この回答法を選びました。

第2節 Android アプリケーション「Fortune Game」の評価方法

Android アプリケーション「Fortune Game」を評価してもらうためにアンケート用紙を作成しました。このアンケートではデザインや操作性が他人にとって満足できるものか調べ、満足した点、不満があった点を書いてもらいました。デザインは前述の SA 回答法と自由回答法を用いて、書く側にとってストレスを感じさせないようにしました。また、A４用紙１枚の表のみに収まるように質問数を６個にし、シンプルなものとしました。図の 1.9 が実際に配布したアンケート用紙となっています。



Android アプリ「Fortune Game」のアンケート
201214070　長田竜太朗

今回、私は FortuneGame というアプリケーションを開発しました。このアプリは爆弾をタッチしすべて不発で終わればクリアというゲームです。是非皆さんに触ってもらい、評価をしてほしいと思っていますため、ご協力お願いします。

1．あなたの性別を記入してください
□男性　　□女性

2．あなたの年齢を記入してください
□10 代　□20 代　□30 代　□40 代　□50 代　□60 歳以上

3．本アプリのデザインはどうでしたか？
□満足　□やや満足　□やや不満　□不満

4．本アプリの操作性はどうでしたか？
□満足　□やや満足　□やや不満　□不満

5．本アプリについてのご感想をお聞かせください。
□満足　□やや満足　□やや不満　□不満

6．５で「満足」、「やや満足」と答えた方、どのような点が良かったか記入してください。

7．５で「やや不満」、「不満」と答えた方、どのような点が良かったか記入してください。

これでアンケートは終了です。ご協力ありがとうございました。

図 1.9　実際のアンケート用紙

質問１から５までは「満足」「やや満足」「やや不満」「不満」の４択でチェック式としています。最後の「本アプリケーションについてのご感想をお聞かせください」という選択肢で「満足、やや満足」を選択した人は質問６で感想を記入してもらい、「やや不満、不満」を選択した人は質問７で不満点を記入してもらえるようにしました。チェック式の集計方法は「アンケートの集計方法を効率的に行う方法について〔６〕」を参考にし、

Excel を使い A 列に「性別」B 列に「年齢」C 列に「デザイン」D 列に「操作性」E 列に「総合評価」を入れました。図 1.10 に示しています。

	A	B	C	D	E
1	性別	年齢	デザイン	操作性	総合評価
2	男性	20代	やや満足	やや不満	やや不満
3	男性	20代	やや満足	やや満足	やや不満
4	男性	20代	やや満足	やや不満	やや満足
5	男性	20代	やや満足	やや不満	やや満足
6	男性	20代	やや満足	やや満足	やや満足
7	男性	20代	やや満足	満足	やや満足

図 1.10　集計結果の表

ここから性別は男性女性に分けるため Excel にある COUNTIF 関数を使い「=COUNTIF($A$2:$A$7,"男性")」という式を作りました。この式では先ほどの集計された A2 から E7 までのセルから「男性」という単語を抽出するために A2 から E7 までを範囲指定し、検索条件を「男性」にして出したものです。その際に「$A$2:$A$7」と$マークを使った理由は、男性の集計結果の下にある女性でも同じセル範囲のため、範囲指定を固定させるためです。次の年齢、デザイン、操作性も式はほぼ同じで年齢の場合は「=COUNTIF($B$2:$B$7,A15)」、デザインの場合は「=COUNTIF($C$2:$C$7,D10)」、操作性は「=COUNTIF($D$2:$D$7,D16)」、総合評価は「=COUNTIF($E$2:$E$7,A22)」と書きました。検索条件を“男性”などと単語ではなくセル名にしたのは、２０代や３０代などと書くことが多いため省きました。

第3節　アンケートの集計結果

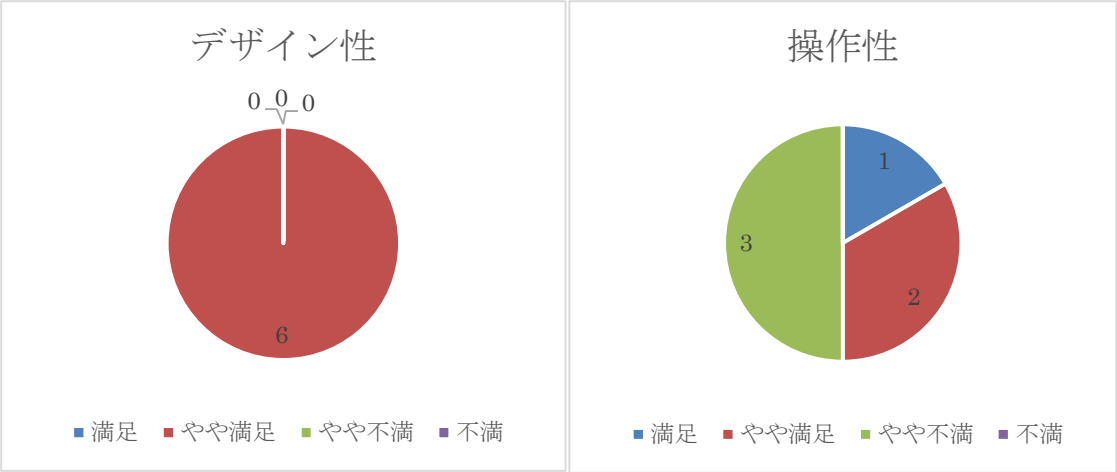
今回、アンケート調査は時間的都合により同じゼミナールメンバーの６名だけとなりました。そのため性別と年齢の集計結果は男性６名、女性０名、年齢は２０代６名、３０代以上は０名となっています（図 1.11、1.12 参照）。

性別	
男性	6
女性	0

年齢	
20代	6
30代	0
40代	0
50代	0
60代以上	0

図 1.11　1.12　性別と年齢の表

次にデザインでは満足が０名、やや満足が６名、やや不満が０名、不満が０名となり（グラフ 1.1、図 1.13 参照）、操作性は満足が１名、やや満足が２名、やや不満が３名、不満が０名となりました（グラフ 1.2、図 1.14 参照）。



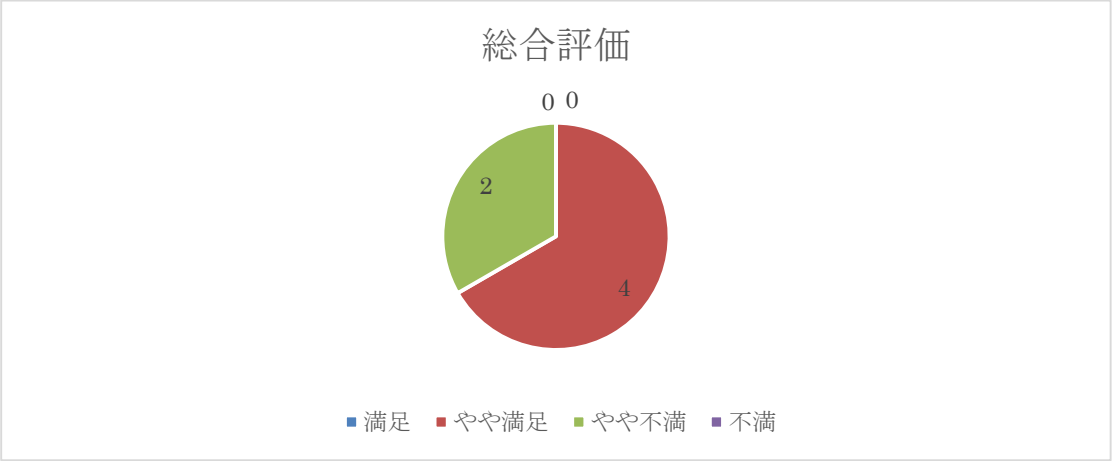
グラフ 1.1、1.2　デザイン性の円グラフが左、操作性の円グラフが右

デザイン	
満足	0
やや満足	6
やや不満	0
不満	0

操作性	
満足	1
やや満足	2
やや不満	3
不満	0

図 1.13　1.14　デザインと操作性の表

総合評価では、満足が０名、やや満足が４名、やや不満が２名、不満が０名となりました。総合評価でやや満足以上の評価をした人の感想を見ると４名中４名全員が「シンプルで分かりやすかった」「大きくて見やすい」という評価をもらえました（グラフ 1.3、図 1.15 参照）。



グラフ 1.3　総合評価のグラフ

総合評価	
満足	0
やや満足	4
やや不満	2
不満	0

図 1.15　総合評価の表

反対にやや不満以上と答えた方の感想では「クリアに成功したらファンファーレなどのサウンドエフェクトがないと虚無感に襲われる」「ランダム要素が強いので、何個以上〇になればクリアにしたほうがわかりやすい」「最初説明がなければ親切設計だと分からず、進み方がわからなかった」などの意見をもらいました。

第6章 考察と今後の課題

第1節 考察

今回アプリケーションの評価を知るためにアンケートを作成し、自分のアプリケーションが周りにどう捉えられているかを調べました。結果は前述の通り良い評価では無く、無難な評価が多かったです。この評価に対し自分なりに考察した結果、中身が無いと思いました。集計結果のデザインを見れば６名全員が「やや満足」と回答しており、デザインに関しては特に問題がないと思われていることがわかります。しかし、操作性の集計結果を見ると「満足」と回答された方が１名いますが、３名が「やや不満」と回答されています。これは、そのまま総合評価に繋がっており、感想にも前述のとおり「ファンファーレのサウンドエフェクトがほしい」や「説明がないと次に進めない」などと書かれていました。そのためまず、最初に気付いたことは内容が不足していることだと思いました。私の中ではこれで完成だと思っていても、周りの人から見ると「あれが足りない」「これがない」など、不足事項を挙げてくれました。

今後の課題として、まず設計書の作成を緻密に行うことだと思いました。今回の評価を見る限り、設計書の製作を手早く終わらせたことによってゲームのアプリケーションに必要な説明が不足していることや、クリアしたあとやゲームオーバーになった後、どうすればいいかわからなかったと思います。アンケートを集計してまとめた後、ファンファーレのサウンドエフェクトはすぐに書けそうだったためリスト 2.1 のコードを追加し、ゲームがクリアした後、ファンファーレが聞こえるようにリスト 2.2 のようにしました。ファンファーレの種類は２種類用意し、ステージ１とステージ２では短めのもの、最終ステージのステージ３では、難しいステージをクリアしたということで少し長めの BGM を使用しました。BGM はこのアプリケーション「Ex100」の「res」フォルダの中にある「raw」フォルダに爆発した時のサウンドエフェクトと一緒に入れました。音源は MP3 拡張子をそのまま使用しました。

```
sp2 = new SoundPool(1, AudioManager.STREAM_MUSIC,0);
sound2 = sp2.load(this, R.raw.muci_fan_08,1);
```

リスト 2.1　ファンファーレの BGM をアプリケーションの中に追加するプログラム

```
Toast.makeText(this, "Conglatulations!", Toast.LENGTH_LONG).show();
sp2.play(sound, 50.0f, 50.0f, 0, 0, 1.0f);
```

リスト 2.2　リスト 2.1 の BGM をクリアした際に鳴らすプログラム

第2節 おわりに

今回アプリケーションを開発する前にサンプルとして、いくつかアプリケーションを開発した後に実際に考えたものを開発し始めました。開発期間は約 1 年で、内容もそれほど悪くはなかったと思いましたが、アンケートの結果ではそれほど良くはなかったため「開発期間が長い＝良いものが出来た」というわけではないことが分かりました。また、ウォーターフォールモデルのような開発モデルで、途中で他人に見てもらいながら開発を進めないと独りよがりなアプリケーションができてしまうことが分かりました。今回のアプリケーション開発は Java プログラミングを覚えるだけでなく、周囲の意見を取り入れることや開発途中にテストを入れることが重要だと思いました。

謝辞

このプログラミング開発を進めるにあたり、２年次から付きっきりでご指導していただいた伊藤則之教授や４年次からご指導していただいた田中章司郎教授、同じゼミナールに所属した皆様に感謝の意を表します。ありがとうございました。

なお、本論文や開発する際に作成したプログラムなど、全ての知的財産権を伊藤則之教授および田中章司郎教授に譲渡いたします。

参考文献

〔１〕http://news.mynavi.jp/news/2015/05/20/171/
〔２〕http://sega-net.com/release/150428_11132.html
〔３〕小野真樹 Eclipse パーフェクトマニュアルベストセレクション　「前書き」　３ページ
〔４〕http://e-words.jp/
〔５〕http://sjc-p.obx21.com/word/jk/fullyqualifiedname.html
〔６〕らくらく図解　アンケート分析教室
〔７〕http://global-wing.com/activity/excel_shukei.html