平成２７年度

卒業論文

題目	Android 所持金管理アプリケーションの企画・開発と評価

担当教員 （自署）		印

学籍番号　201214053

氏　名　真田　成弘

広島経済大学

広島経済大学

要旨

第 1 章では、アプリケーションの開発の目的と経緯について、開発するアプリと関連のある既存のアプリの状況について述べている。

第 2 章ではアプリケーションの開発環境、開発するアプリケーションの概要とアプリケーションのプログラムの解説をしている。プログラム解説には実際に使用したコードを貼り、どのような処理を行ったか解説している。

第 3 章では、第 2 章で企画・開発したアプリケーションのプログラムの実際の動きを実機に実装した画面を加えて解説している。

第 4 章ではアンケート用紙を作成し、アプリを実装した端末をゼミナールのメンバーや家族に実際に使用してもらい、その後アンケートに答えてもらった集計結果でアプリを評価したものを述べている。

第 5 章では第 4 章で行ったアプリの評価から出てきたアプリに残された課題や追加・改善点を挙げ、それをどのように解決していくのかを述べている。

最後の第 2 節では結論として論文全体のまとめを述べている。

目次

第1章 はじめに

第1節 既存の所持金管理アプリケーションの状況と目的・開発経緯

アプリケーションの開発を行った目的は、アプリケーションとはどのように開発されているのかを実際にアプリケーションを開発する活動を通して理解を深めることである。

今までに多くのアプリケーションが配信され、同じ目的のアプリケーションでも何種類も　配信されている場合が多い。所持金管理アプリもその内の１つで、既に多数配信されている。それらのアプリはこれまでの数値を棒グラフ等に変換してくれるものまである〔１〕。

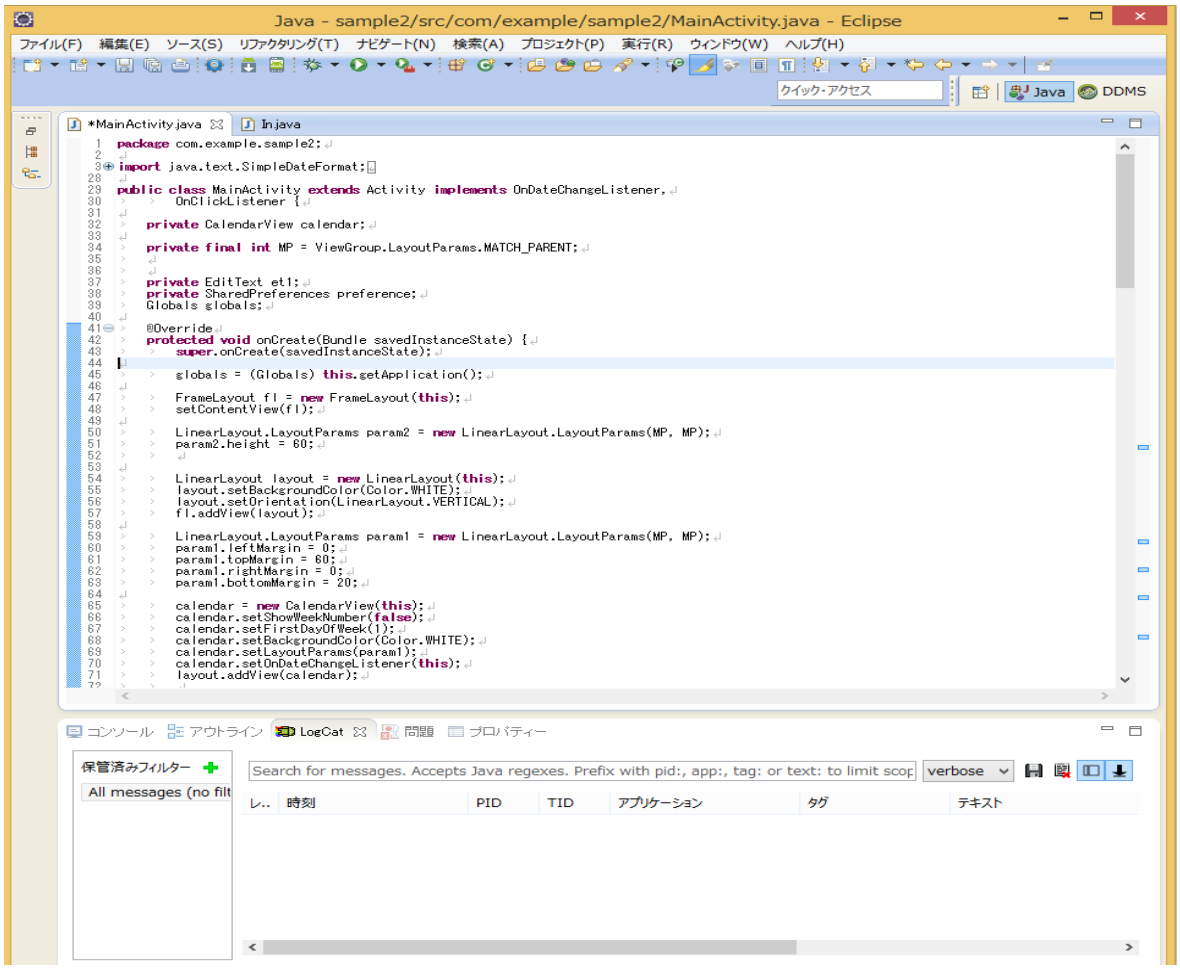
しかし、内容が多機能になるにつれて操作も簡単にはならなくなる、所持金管理アプリの様に利用していれば何度も使う事になるアプリケーションの場合は最小限の機能でより単純な操作のものの方が使いやすいのではないかと考え、企画・開発を行った

第2章 所持金管理アプリの企画・設計

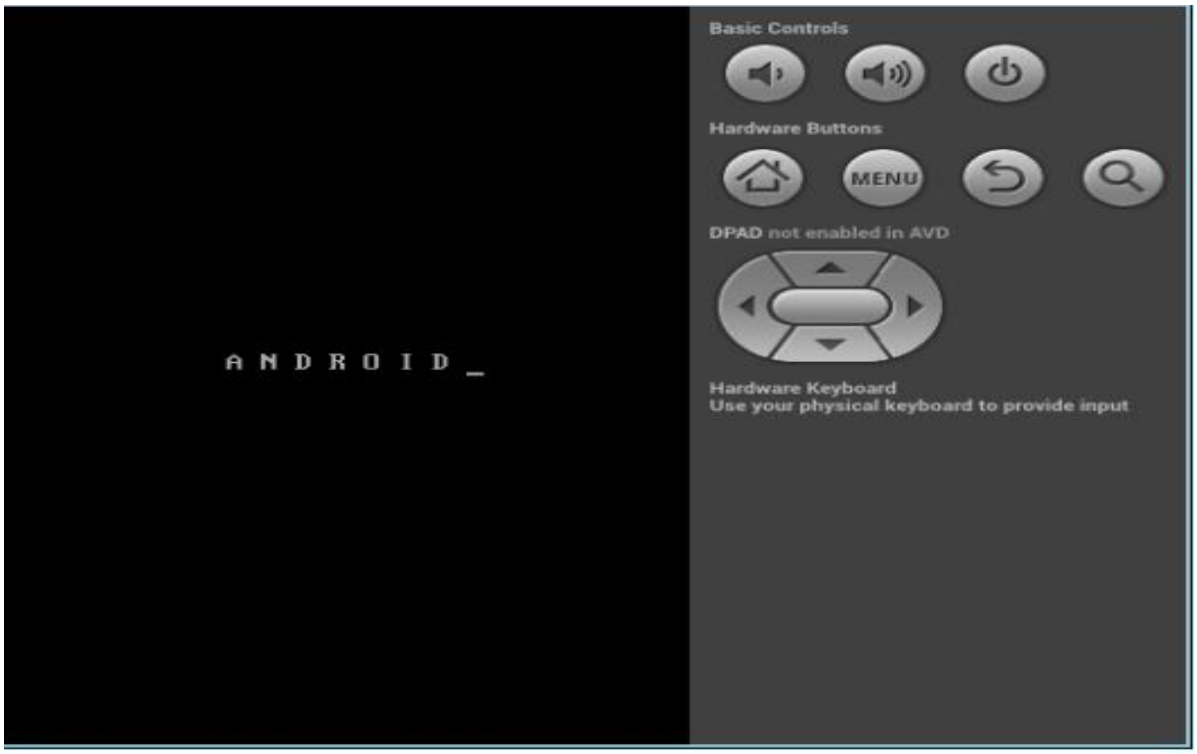
第1節 開発環境

開発環境は Eclipse を使用した。(図1)

Eclipse は、IBMによって開発されたオープンソースの統合ソフトウェア開発環境IDEの一つで、プラグインを使用すれば様々な言語に対応することができるが、 Eclipse 自体が Java で記述されていることもあり、 Java の開発環境として主に利用される。利点としてコードを数文字入力すると、予測変換で候補に挙がるコードを表示してくれる事や、コードにエラーが出た際には行番号に×印が表示されエラーを起こしている箇所を確認しやすい等のメリットがあり、初心者でも簡単にアプリケーションの開発を進めることができる。開発を進める段階で、プログラムを入力した後にそれが正常に動作するかテストする必要がある。その際には (図2)の様な仮想デバイスと、 Android 端末の実機に接続して確認した。



上（図1:開発環境 Eclipse)



下（図2:仮想デバイス）

第2節　アプリケーションの概要

本研究で作成したアプリケーションはグラフ化の様な必須ではない機能を省いて誰でも簡単に使える事を目的とした、月別の所持金を管理する所持金管理のアプリケーションとした。

メインとなる画面は、月別、日別に管理しやすくするために、縦スクロールのカレンダー形式にした。起動するとカレンダーになっているメインとなる画面が表示される。

まず、所持金データ入力用のテキストボックスに所持金を入力し、 OK と書かれたボタンを押すと、入力されたデータを保存・テキストボックスに表示する。

カレンダーの日付をタップすると日別に消費した金額を入力する画面に移行し、テキストボックスに使用した金額を入力する。

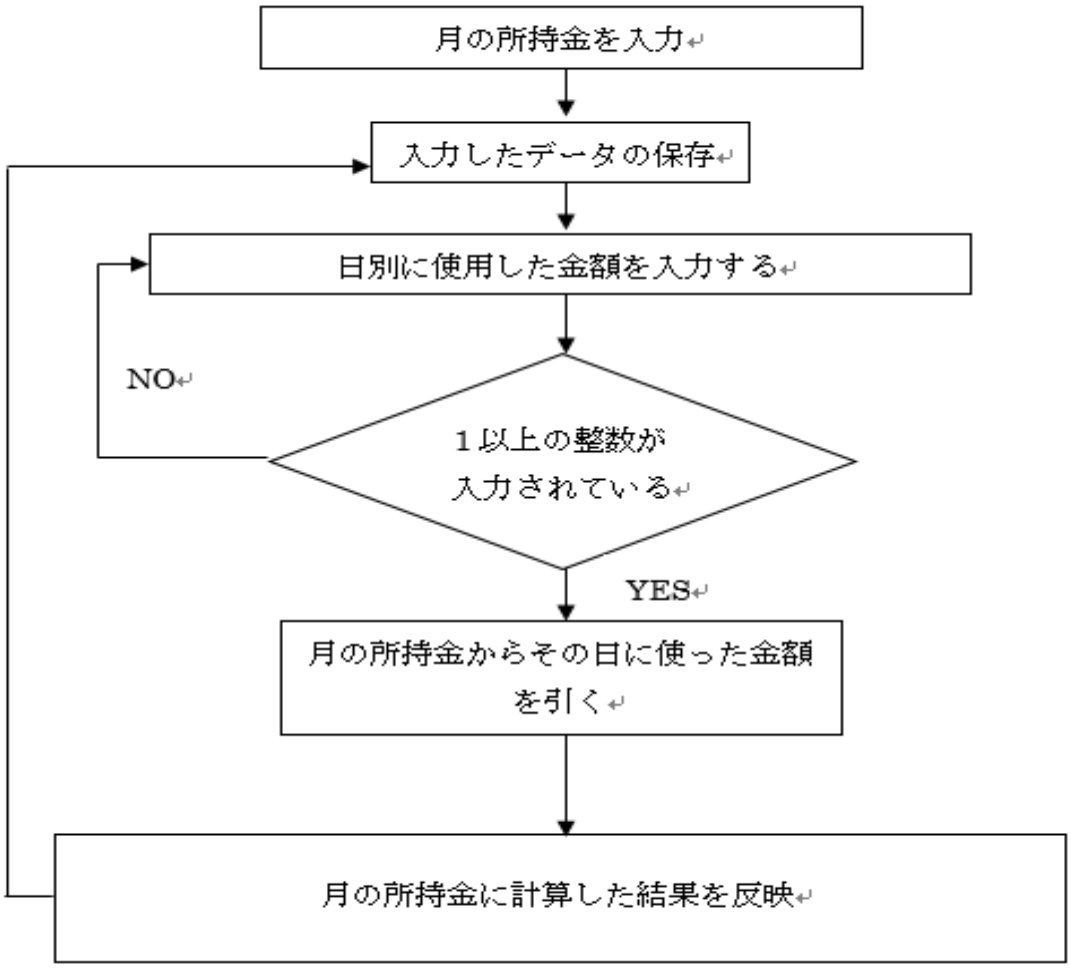
入力した後、テキストボックスの下に配置されているボタンを押すと、テキストボックスに入力されたデータを保存し、メイン画面に戻る。

所持金に入力された金額から、日別に消費した金額を引き、計算結果で所持金を上書きする。

終了操作は、メイン画面でバックボタンを押すと終了確認ウィンドウが表示され、本当に終了するか確認する。

第3節　コード概要

ここでは、所持金管理アプリにおいて主要となるプログラムについて説明していく、プログラムの処理手順は下記のフローチャートの様になっている。（図7:フローチャート）



（図7:フローチャート）

次に、コード部分の説明をしていく。

リスト１は、アプリケーション内全てのクラスに Globals を適用するものである。

Globals は、アプリケーション内の各クラスで打ち込んだプログラムをアプリケーション全体で利用できるようにするものである。

```
1 package com.example.sample2;
2 import android.app.Application;
3 public class Globals extends
4  Application {
        String day;
```

（リスト 1:Globals.java コード）

```
01        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMM");
02         String selectedDate = sdf.format(new Date(calendar.getDate()));
03         preference = getSharedPreferences("kakeibo", MODE_PRIVATE);
04        String money = preference.getString(selectedDate, "");
05        et1.setText(money);
06        }
07        public boolean onCreateOptionsMenu(Menu menu) {
08                  getMenuInflater().inflate(R.menu.main, menu);
09                  return true;
10        }
11        public void onSelectedDayChange(CalendarView view, int year, int month,
12                  int dayOfMonth) {
13                   String yy = String.valueOf(year);
14                  String mm = String.valueOf(month + 1);
15                  if (mm.length() == 1) {
16                            mm = "0" + mm;
17                  }
18                  String money = preference.getString(yy + mm, "");
```

（リスト2:ボタン・日付を選択した時の処理）

リスト２は、現在の年月に予算金額が記録されている時の処理と、日付を選択した時の処理についてである。

02~04 行で現在表示している年月を取得して、もし現在の年月に予算金額が記録されていた場合 05 行目で変数 money として et1 という EditText に表示させている。

11行~23行目では、カレンダーの画面で別の日が選択された時の処理を行っている。

11~21行で変数yy、mm 、dd にそれぞれ変数year、month、 dayOfMonth から選択された年、月、日を取得する。この時、 14 行目の選択されている月を取得する時に数値的に0月から始まるため変数monthに +1 を加えて調整している。

```
01public void onClick(View v) {
02        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMM");
03        String selectedDate = sdf.format(new Date(calendar.getDate()));
04                SharedPreferences.Editor editor = preference.edit();
05                 editor.putString(selectedDate, et1.getText().toString());
06                editor.commit();
07        }
08         protected void onActivityResult(int requestCode, int resultCode, Intent
 data) {
09                super.onActivityResult(requestCode, resultCode, data);
10                if (requestCode == 1) {
11                        et1.setText(String.valueOf(globals.nokori));
12                }
```

（リスト3:所持金額の保存処理と選択された日から戻る時の処理）

リスト3は、 et1 に入力された予算金額のデータを、現在の年月をキーワードにして記録する処理である。 et1 の隣に設置したボタンがクリックされると処理を開始する。

3行目で selectedDate を使ってカレンダーで選択されている年月を取得し、それを数値化したものと et1 に入力した予算金額と関連付けて何年何月にいくらの予算を入力したのかを記録するようになっている。

```
01    public void onClick(View v) {
02     SharedPreferences.Editor editor = preference.edit();
03     editor.putString(globals.day, et1.getText().toString());
04     editor.commit();
05      String yosann = preference.getString(globals.day.substring(0, 6), "");
06     if (yosann.length() == 0) {
07       Toast.makeText(getApplicationContext(), globals.day.substring(0, 6)
08           + "の予算が入力されていません", Toast.LENGTH_LONG).show();
09      } else {
10       Log.d("VarDebug", "yosan=" + yosann);
11       Log.d("VarDebug", "length=" + yosann.length());
12       int i_yosann = Integer.parseInt(yosann);
13       for (int i = 1; i < 32; i++) {
14         String money = preference.getString(globals.day.substring(0, 6) + i,
```

（リスト 4.1 計算処理）

```
15          if (money.length() > 0) {
16            try {
17              i_money = Integer.parseInt(money);
18            } catch (NumberFormatException nfex) {
19              Log.d("VarDebug", "i=" + i + " money=" + money);
20        Toast.makeText(getApplicationContext(), globals.day.substring(0, 6) + i
21         + "の予算が" + money + "になっています ", Toast.LENGTH_LONG).show();
22            }
23            i_yosann = i_yosann - i_money;
24          }
25        }
26        globals.nokori = i_yosann;
```

（リスト 4.2 計算処理）

リスト 4 は、日別の支出金額を入力した後ボタンを押し、所持金から入力した支出金額を引き、残額を計算する処理を行っている。

02~04 行目は、SharedPreference という機能を使って、毎日の支出金額をアプリに記録する処理。SharedPreference という機能は、たとえば”20160118”のような数字 8 桁の文字列に対して、”850” などのように 2016 年 01 月 18 日に支出した金額 850 円を記録することができる。つまり、”20160118 ” のような文字列をキーとして、そのキーに ”850” などの文字列をデータとして記録することができる。この機能を使い、”20160118”のような日付を表す数字 8 桁の文字列に対して、毎日毎日の支出金額をそれぞれ記録して行く。

04 行目は、そのための SharedPreference という機能を作り出す処理であり、 03 行目では globals.day に入っている日付を表す数字 8 桁の文字列をキーとして、これに対して

et1 という名前の EditText の部品に入力された数字（今回の場合はその日に支出した金額）を et1.getText().toString() により文字列化したものを記録させるための対応付けをしている。

そして、 04 行目で実際に記録の処理をしている。なお、globals.day は、複数の画面でデータをやり取りするために用意した Globals.java の中に用意した day という変数の値を意味する。

05 行目は今選ばれている日の”201601”のような年月にすでに記録している予算金額を読み出している。 globals.day.substring(0, 6) は、globals.day に入っている”20160118”のような数字 8 桁の文字列の最初から 6 文字分を切り出す処理を意味し、”20160118”の場合は”201601”というように年と月を示す 6 文字が切り出される。この 6 文字をキーとして記憶しているデータを取り出すのが preference.getString の部分であり、もしこのキーに対して記憶しているデータがない場合は、”” という空の文字列を返すように指示している。このようにして取り出されたデータを変数 yosann に入れている。

06 行目は、この yosann に入っている文字列の長さを yosann.length() で計算して、その長さが 0 、つまり ”” という空の文字列の場合は、予算データが記録されていないということで、 09 ～ 10 行目でその旨のメッセージを表示している。

もし、 yosann に予算データが入っている場合の処理が 09 ～ 27 行目となる。

10 ～11行目は、 yosann に入っているデータと文字列の長さを LogCat に表示して確認するための処理であり、これはアプリ開発過程での確認用に使われる。

12 行目は、 yosann に文字列として入っているデータを数のデータに変換して、変数 i_yosann に入れるための処理を行う。

13 ～ 25 行目は、現在表示している年月の 1 日から 31 日までに毎日支出した金額を予算 i_yosann から順次引き算して、予算の残り金額を計算している。 14 行目では 13 行目の for 文の i に順次 1 から 31 まで入る数を年月の文字列のうしろに globals.day.substring(0, 6) + i で追加して、その文字列をキーに記録しているデータを取り出して、money という変数に入れている。 15 行目では、money という変数に入っている

文字列の長さを計算して、長さが 0 より大きい場合はデータが入っているため、 17 行目でmoneyに文字列として入っているデータを数のデータに変換して、変数 i_money に入れるための処理を行っている。 moneyには数字からなる文字列が入っているはずだが、もし数字以外の文字が入っている場合はエラーを表示するために、 17 行目の処理を 16 行目の try と 18 行目の catch で囲んでいる。数字以外の文字が入っている場合はエラーとして catch され、 19 ～ 21 行目でエラーメッセージを表示している。 23 行目では、予算の残りi_yosannから支出i_moneyを引き算している。このように for 文で 1 日から順次その日の支出を予算から引き算しながら、残りを計算している。

26 行目では、その月の残り金額が計算できたので、 globals.nokori にi_yosannの値を入れている。

最後にリスト 5 でアプリケーションを正常に終了するための処理を行っている。

01~02 行目でどのボタンが押された時に処理を行うか設定。今回は端末ごとに場所は異なるが、 Back ボタンを押すことで終了確認画面を出すようにした。

Back ボタンが押された時の処理内容が 03~22 行であり、誤操作による意図しない終了を防ぐために Back ボタンが押されると最初に本当に終了するかを尋ねるアラートダイアログボックスを表示し、終了したい場合は Yes を押すとアプリケーションが終了する。もし終了したくない場合は No を押すことでアプリケーションに戻るようになっている。

```
01        public boolean onKeyDown(int keyCode, KeyEvent event) {
02                if (keyCode == KeyEvent.KEYCODE_BACK) {
03                        new AlertDialog.Builder(this)
04                                .setTitle(" アプリケーションの終了")
05                                .setMessage("アプリケーションを終了してよ
ろしいですか？")
06        .setPositiveButton("Yes",
07                        new DialogInterface.OnClickListener() {
08                public void onClick(DialogInterface dialog,
09                int which) {
10                        MainActivity.this.finish();
11                }
12        })
13         .setNegativeButton("No",
14                new DialogInterface.OnClickListener() {
15                public void onClick(DialogInterface dialog,
16                int which) {
17                                        }
18                                .show();
19                return true;
20                }
21         return false;
22        }
```

（リスト5:終了処理）

第3章　アプリケーションの実装

図 3~9 は、実際にアプリケーションを実装した画像である。画像は実機で実行した画面をスクリーンショットで撮影したものを使用している。（図3)

まず画面上部のテキストボックスを押し、月別の所持金を入力する。入力後テキストボックスの右側のボタンを押すと。入力したデータが保存される。入力したデータは、保存後にテキストボックス内に所持金のデータとして表示される今回は 1000 と入力した。（図4)

所持金の入力が終わったところで、日付を押して日別に支出したデータを入力する画面に移行する。

（図3: メイン画面）

（図4: 金額入力後）

データを入力したい日付を押すと、（図5)の画面になる。ここに所持金の時と同様にし

て支出金額を入力し完了したら OK ボタンを押す。この OK ボタンを押した時に第 3 節で解説した処理計算が行われる。メイン画面の所持金が計算結果である残額に上書きされて、メイン画面に戻る。画像では支出金額に 500 と入力し、 OK ボタンを押した。

（図5: 日別支出金額入力画面）　　（図6: 支出金額計算処理後）

(図7)は誤操作による意図しないアプリケーション終了を確認する画面である。

アプリを終了するとき、つまり画面下に 3 つ横に並んでいる一番左の「戻る」ボタンがタッチされたら、本当に終了してよいかを Yes または No をタッチしてもらって確認するための処理を行っている。

（図7:終了確認画面）

（図7)は誤操作による意図しないアプリケーション終了を確認する画面である。

アプリを終了するとき、つまり画面したに３つ横に並んでいる一番左の「戻る」ボタンがタッチされたら、本当に終了してよいかを Yes または No をタッチしてもらって確認するための処理を行っている。

第4章 開発したアプリの評価

評価方法は実際に実機でアプリケーションを使用してもらい、アンケート用紙を作成しその答案で評価を集める方法をとった。

家族や同じゼミに所属するメンバーを中心に協力して貰い、評価を集めた。（図８アンケート用紙）

質問１　年齢を教えてください。

□20 歳以下　□21~30 歳　□31~40 歳　□41~50 歳　□51 歳以上

質問２　普段スマートフォンで所持金管理アプリを使用していますか？

□はい　　□いいえ

質問３　アプリケーションのボタンの大きさや配置のレイアウト

□とても良い　□まあまあ良い　□余り良くない　□悪い

質問４　アプリケーションの操作性はどうでしたか。

□良い　　□悪い

質問５　今回のアプリケーションは使いやすかったですか？

□使いやすかった　□どちらでもない　□使いにくかった

質問６　アプリケーションを選ぶ時に重視する事を教えてください。

□機能　　□見た目　　□話題性があるか　□無料・有料

□ユーザーのレビュー

質問７　今回のアプリは使いやすいと感じましたか？

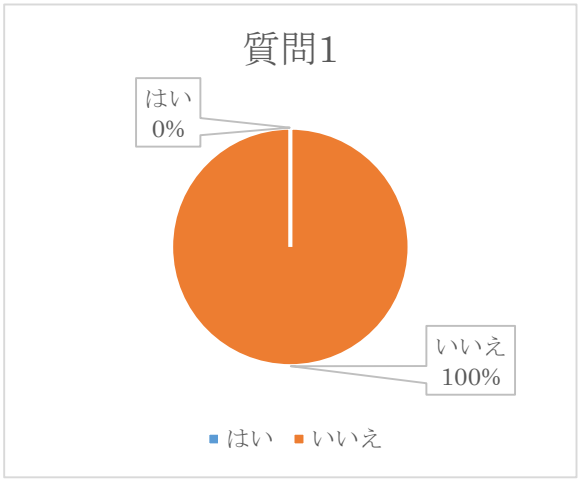
質問８　機能に物足りないと感じた所や、追加したほうがいいと思う機能があれば教えてください。

以上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８アンケート用紙）

回答の集計結果を質問ごとにまとめ、グラフ化した。この結果をもとにアプリケーションに残った今後の課題と改善点を見つけていく。

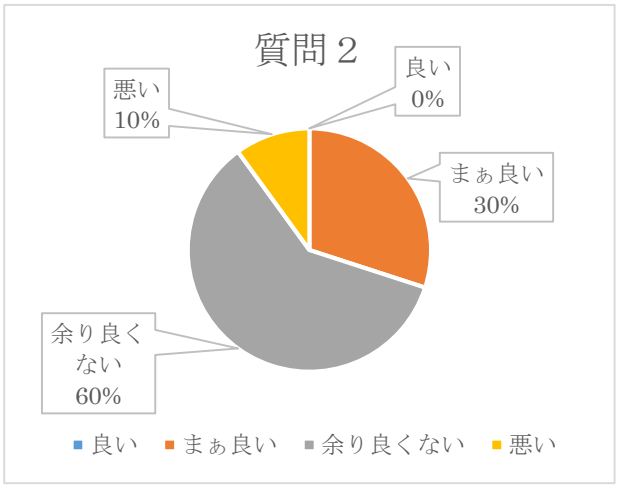
質問１　普段スマートフォンで所持金管理アプリケーションを使用しているか。



（図9）

図９は質問１の「普段スマートフォンで所持金管理アプリケーションを利用しているか」を表している。 10 人に集計を取り、質問１の結果は、「はい」が０人、「いいえ」が 10 人という結果となった。

質問２　アプリケーションのボタンの大きさや配置等のレイアウトについて



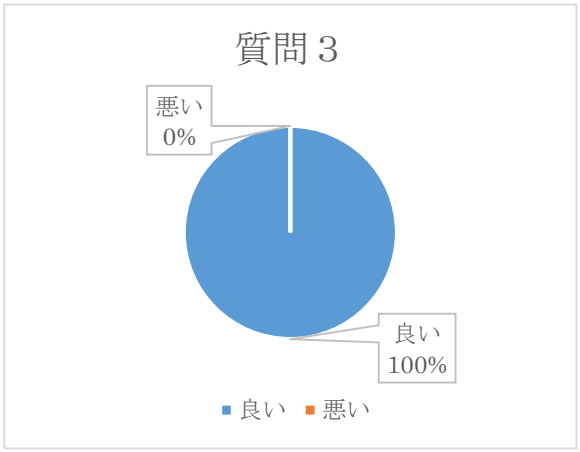
（図 10）

質問２「アプリケーションのボタンの大きさや配置等のレイアウトについて」の回答は、多い順に「余り良くない」６票、「まぁ良い」３票、「悪い」が１票となった。

「まぁ良い」「余り良くない」という意見には、「カレンダー式なのは使いやすかったが、

横スクロールの方が使いやすい」「ボタンが小さい」等の意見があった。

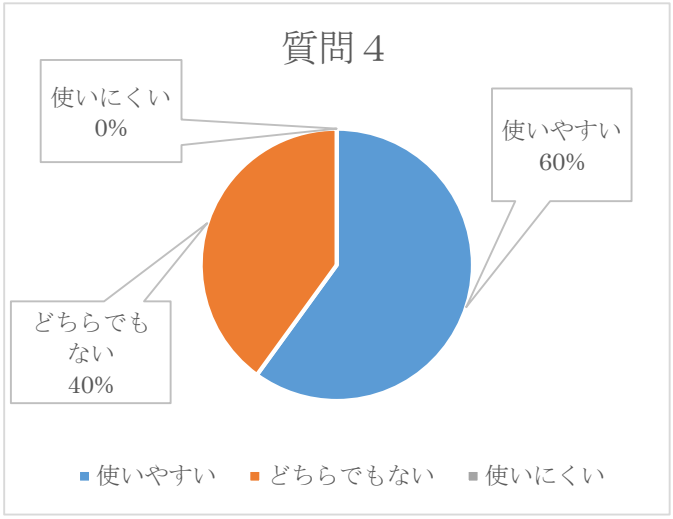
質問３　アプリケーションの操作性はどうだったか。



（図 11）

（図 11）「アプリケーションの操作性はどうだったか」についての回答は、「良い」 10 票、「悪い」 0 票となり、これについては高評価であった。

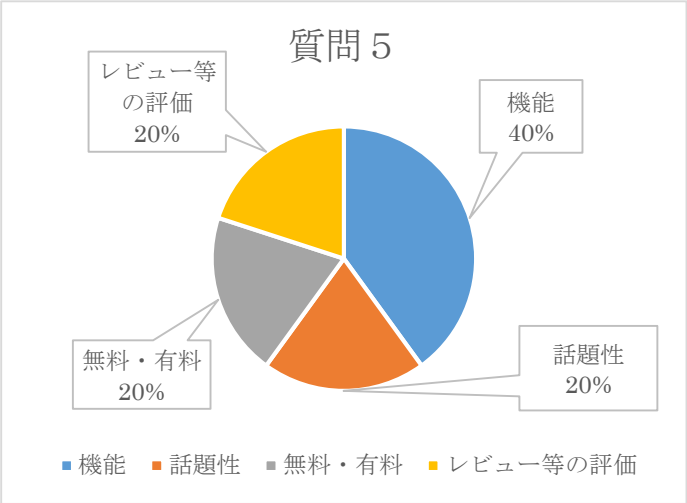
質問４　今回のアプリケーションは使いやすいと感じたか。



（図 12）

（図 12）「今回のアプリケーションは使いやすいと感じたか」についての回答は「使いやすい」6 票、「どちらでも無い」4 票、「使いにくい」0 票となった。

質問５　普段アプリケーションを選ぶ際に重視する点を教えてください。



（図 13）

（図 13）のアプリケーションを選ぶ際に重視する点についての回答は、アプリケーション選びには「機能重視」が 4 票、「話題性」が 2 票、「無料・有料」が 2 票、「レビュー等の評価」2 票となった。

質問 6 では、記述欄を設けてそこに自由にアプリケーションに対する意見を書いてもらった。

質問 6　物足りないと感じた点や追加した方が良い等の意見があれば教えてください

データの管理に関する意見として、「複数回入力したデータの中からピンポイントで削除できる機能が欲しい」「データを入力してある日付に色が付く等、一目で見やすくなるような機能があれば良い」「日付にデータを入力した後、一目で分かるようにカレンダーの日ごとに金額が表示されるとわかりやすくて良い」「使い続けた場合に、前月に残った所持金を次月の所持金に繰り越せる機能があった方が継続して使用する人には使いやすいと思う」「保存されたデータのリセットや変更する機能が欲しい」という意見があった。レイアウトや、細かい機能に関する意見は、「カレンダー式なのは使いやすかったが、縦スクロールのものよりも、横スクロールの方が使いやすい」「ボタンやテキストボックスのサイズを自由に変えられると良い」「金額の後ろに金額の単位があると良いと思う」「別ウィンドウで各月の収入と支出がわかると長期的に使うアプリとしては良いものになると思う」という意見があった。

第5章 おわりに

第1節 評価の意見から得た課題・改善点と考察

アンケートを集計した結果、元々所持金管理アプリケーションを利用している人は一人もいなかったが多くの意見を得ることができ、改善点・問題点を見つけることができた。まず、アプリケーションのボタン配置等のレイアウトについてだ。レイアウトについて「テキストボックスやボタンのサイズが小さい」、「カレンダーの形式は縦スクロールのものよりも横スクロールのものの方が使いやすい」等の意見があった。これは、アプリケーションではテキストボックスやボタンのサイズを数値指定していたため、機種によっては大きすぎたり小さすぎたりする。そこで画面サイズに比例してサイズが変更される様に変更する等して対応しなければならない。

次に、アプリケーションを選ぶ際に重視する要素について、アンケートの集計結果では機能を重視する意見が最も多かった。そのため、物足りないと感じた点や、追加すると良いと思う機能について多数指摘された点があった。まずは「複数回入力したデータの中からピンポイントで削除できる機能が欲しい。」、「入力されたデータをリセットする機能が欲しい。」といったデータの保存・消去に関する意見だ。現状の機能では処理できないので、これについては月に入力したデータ、日に入力したデータを管理するデータ管理画面といった追加を加える必要がある。もう一つ、「カレンダーの画面で、日別に入力した金額が表示されると良い」という意見についてだが、この問題は数字の桁数にもよるがレイアウトがカレンダー形式である以上、1日に割ける枠の大きさに限界があるため、入力した数値をそのまま日付ごとに表示することは難しい。よって、その日の増減によってカレンダーのその日付の色が変わる様な変更を加えれば分かりやすくなるだろう。

本アプリは、所持金のデータは月別で保存されるが、その月別で保存して管理する機能よりも、「前の月に残った金額を次月に繰り越せる機能にして、所持金の残額を次月に繰り越していった方が良い」という意見もあった。所持金のデータを繰り越すか、月別で管理するかのどちらがいいかについては個人の好み次第な部分なので、この二つを自由に選べ

るような機能に追加、変更した方が良さそうだ。

改善点をまとめると、テキストボックスやボタンのサイズについて、アプリを実行する機種によっては大きすぎたり小さすぎたりする場合があることの改善策は、アプリを実行する機種の画面サイズに比例するように変更する。また、エラーが出た際のエラーを知らせるメッセージ文が小さく、すぐ消えてしまうので気づきにくく、もう少しエラーメッセージが目立つ様な変更を加える必要がある。縦スクロール式のカレンダーを使用したが、これよりも横スクロールの方が人気であることが分かったため、変更する。

機能の追加点は、データ管理に関する問題を解決するには、現状の機能では解決できないので月、日に入力したデータの履歴を閲覧・変更・消去できるようなデータ管理画面と機能を追加する必要がある。また、カレンダーの表示形式についても縦横選べる様な機能があると良いかもしれない。

アンケートを実施することで、開発を行った自分以外のユーザーに実際にアプリケーションを使用してもらい、貴重な意見を集めることができた。人によって必要最低限の機能の線引きが違うため、アプリケーションをより良いものとするためには、どの程度の機能があればシンプルで使いやすい機能のアプリケーションとなるのかより多くのユーザーから意見を集めることが重要となる。加えて現状で分かっている問題点を改善していく必要がある。

第2節　結論

第１章で述べたように、アプリケーション開発を行った目的は、アプリケーションに対する理解を深めるために実際に企画し、開発することだ。

企画は既存する多機能のアプリケーションから、可能な限りの必要最低限の機能のみを実装したアプリケーションを開発し、多機能のものより使いやすくなるのかを調査する形をとった。

開発の課題としては、アンケートの集計から、意見として「使いやすさ」は高評価を得たが、機能面ついての意見で「日付にデータを入力した後、一目で分かるようにカレンダーの日ごとに金額が表示されるとわかりやすくて良い」等、現状のままでは機能が不足しているという結果となった。

Google Play 内にある類似のアプリにはアンケート内の意見にもあった、「カレンダーの画面で、日別に入力した金額が表示されると良い」という作成したアプリケーションに残された課題を解決する機能がある。

「かけ～ぼ（家計簿）」というアプリ〔2〕は、その課題を解決している。

しかし、このアプリには、その他にもデータをグラフ化する様な機能が搭載されている。今回作成したアプリケーションは必要最低限の機能で使いやすさを求めたものなので、シンプルさを残しながら機能を追加していくことでより良いアプリケーションになる。

謝辞

最後に、この場を借りて2年次、3年次でお世話になった伊藤則之教授、4年次より卒業論文作成の指導をしてくださった田中章司郎教授、アンケートに協力してくださった方々に深く感謝致します。

本論文、本研究で作成したプログラム及び、データ等全ての知的財産権を本研究の指導教員の田中章司郎教授に譲渡致します。

引用文献

〔1〕https://play.google.com/store/apps/details?id=com.donapon.pisces.cashbook

〔2〕https://play.google.com/store/apps/details?id=com.donapon.pisces.cashbook