Android の導入的な何か

鳥見晃平

2019 年 10 月 29 日

はじめに

本講習会では Android アプリ開発入門を行います。本講習会を履修している皆さんの中には Java を初めて触る方もいるということで、本書は Android プログラミングを行うために最低限必要な Java の解説を行います。

何かわからないところや、より Java の深いところまでやりたいと思ったらぜひ質問してみてください。

お詫び

早速で申し訳ないのですが、AIC のサイト上では「要 Java 経験」と書きましたが、プログラミングの初心者でもついて行けるような内容にすることになりました。ですので、最初は Java 経験者の方は退屈だと思います。申し訳ありません。復習だと思って読んでみてください。

目次

第 1 章

Android と Java について

1.1　Android について

Android とは Google が開発した OS です。Android の特徴の一つとして、誰もが無償で利用可能なオープンソースのモバイル端末向け OS であることがあげられます。これを各企業が独自にカスタマイズすることによって現在世に出回っている様々な Android 端末が生み出されています。

1.2　Android のプラットフォーム

Android は、完全にはゼロから作り上げられたプラットフォームではありません。オープンソースの UNIX 互換 OS である Linux をベースにした、様々なソフトウェア集合体です。図 1.1 に Android のアーキテクチャの階層構造を示します。今はこういうものがあるんだー程度に思ってください。

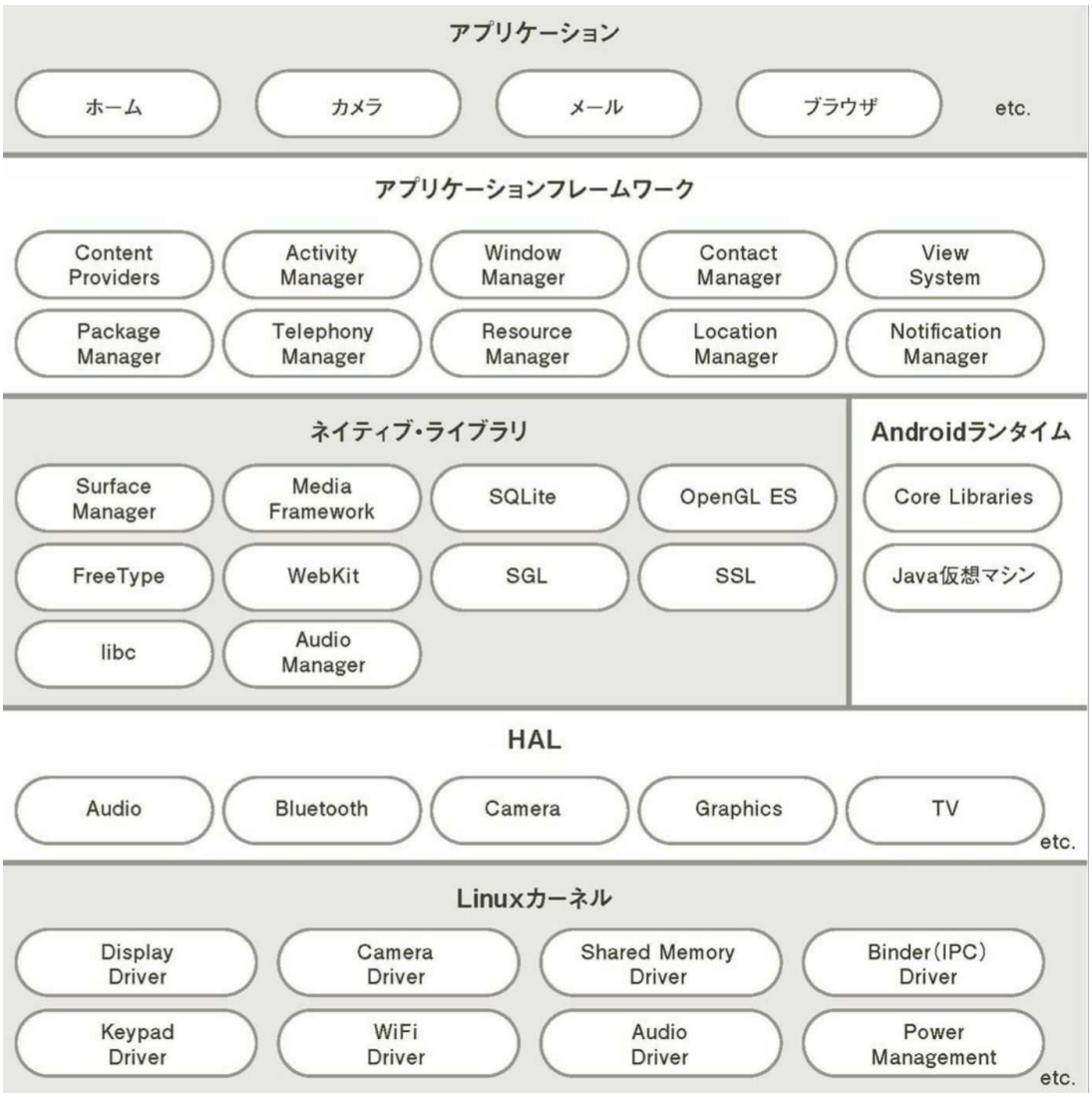

図 1.1　Android のアーキテクチャ

各層の内容についてだけ説明します。

・アプリケーション

最上位の階層には、ユーザーが実際に使用する Android アプリケーションが位置します。個々のアプリケーションには ”アクティビティ” と ”サービス” という二つの要素があります。アクティビティとは簡単に言えば”アプリの画面”です。それに対してサービスとは“バックグラウンドで動作可能なプログラム”です。例えば、Google Play でのアプリのダウンロードはサービスとして実行されるため、Google Play を画面に表示していなくても行われます。

・アプリケーションフレームワーク

アプリケーション構築のための様々な機能をまとめたものです。実際に Android アプリケーションを開発する場合、これらのフレームワークに用いられる API(Applicational Program Interface) を利用することになります（最終回で詳しく説明します)。例えば、GUI ツールキットである View System では、ボタンやテキストボックス、WebView などの部品が用意されています。

・ネイティブ・ライブラリ

ここには C や C++ で開発されたライブラリが存在します。例えば、3D グラフィックエンジンの「openGL ES」がこれにあたります（~~皆 open GL やろう~~)。

・Android ランタイム

この階層は Java で記述されたアプリケーションの実行環境です。次節で詳しく説明します。

・HAL（Hardware Abstraction Layer）

この階層はハードウェアに簡単にアクセスできるようにするためのものです。

・Linux カーネル

先ほど説明した Linux をモバイル向けにカスタマイズしたものです。

1.3　Android アプリ開発について

Android アプリ開発では Java というプログラミング言語を使用します（ほかの言語もありますが、ここでは省きます)。

Java のキャッチコピーに「Write Once Run Anywhere」というものがあります。これは、作成した Java プログラムが OS に依存せず Windows、Mac、Linux で同じように動くことを意味します。これは「Java VM(Java Virtual Machine)」と呼ばれる仮想的なコンピューター環境を実現するソフトウェアによって動作しています。

ここで、人が読めるテキスト形式のプログラムのことを**ソースファイル**、CPU が実行可能な形式のファイルのことを**オブジェクトファイル**、ソースコードをオブジェクトファイルに変換することを**コンパイル**といいます。Java のソースコードをコンパイルすると、「バイトコード」と呼ばれるオブジェクトファイルが生成されます。このバイトコードは Java VM が実行可能なオブジェクトファイルです。Java VM は、バイトコードを読み込むと、それを、逐一現実の CPU マシン語に変換しながら実行します。つまり、Java VM を OS ごとに用意することによって、同じプログラムが Windows でも Mac 上でも動作します。

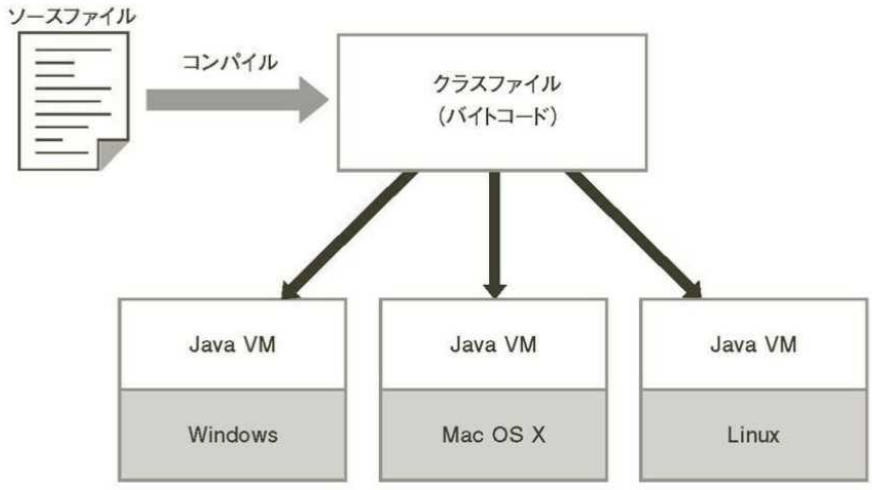

図 1.2　Java VM

ごちゃごちゃ書きましたが、とりあえず Java は Java VM を使ってワンクッション置くことで OS 毎の違いをうまいことやってると覚えてもらって大丈夫です。

Android 上でも ART VM（Android Runtime VM）という、スマホのような低メモリの環境に最適化され、複数の仮想環境が効率的に動作することを目的とした Android 専用の VM が使用されています（Android5.0 以降）。そのため、通常の Java VM で動作するソースコードをそのまま持ってきても動作しません。

第 2 章

Java 入門

2.1 Hello World!

今回は、次回以降の講習会で必須になる、Java の文法の基本的な部分だけを解説します。ですので、Java 初心者にとっては難しい内容は触りません。

本講習会では、Java の入門のために `https://paiza.io/ja` というオンライン実行環境を使用します（ほかの実行環境を使ってもらっても大丈夫です）。ここで「Java」を選択すると以下のようなコードがあると思います。

Listing 2.1 ソースコード 1

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) throws Exception {
5         // Your code here!
6
7         System.out.println("XXXXXXXX");
8     }
9 }
```

このコードの「XXXXXXXX」と書かれている部分を「Hello World!」に書き換えて実行ボタンを押してみてください。

出力の欄に以下のような結果が出ると思います。

```
1 Hello World!
```

プログラムがコンパイルされました。次の節で詳しく解説します。

2.2　クラス、メソッド、コメント

先ほどのプログラムの中にあった

```
1 public class Main {
2     ...
3 }
```

以上のように記述するものを**クラス**と呼びます。

クラスとは、プログラムを実行するための処理をまとめたもので、一つの部品のようなものです。

クラスは難しいので、とりあえず名前だけ憶えてもらえれば大丈夫です。

また、

```
1     public static void main(String[] args) throws Exception {
2         ...
3     }
```

以上のように記述するものを**メソッド**と呼びます。

メソッドとは、クラス内に用意されるいくつかの処理をひとまとめにしたものです。

他の言語では「関数」と言ったりします。

CPU にやらせる命令は全てこのメソッドの中に書くことになります。

```
1     // Your code here!
2     /* un */
3     /** k0 */
```

以上のように記述するものを**コメント**と呼びます。コメントはソースコードに記述する注釈のことで、コメント内の内容はコンパイル時には無視されます。

上のいずれの書き方でもコメントとして機能します。

ソースコード 1 では main メソッドの中に書かれている「System.out.println」という () の中に書かれた文字を出力に表示する命令を実行しました。

2.3　基本データ型

プログラミングでは、数字や文字、配列など様々なデータを扱います。このうち、数字や文字のようなシンプルなデータを**基本データ型**といいます。そして、それらの値を入れておく箱のことを**変数**といいます。

Java には 8 種類の基本データ型が用意されています。

分類	データ型	保持できるデータ
整数型	byte	8ビット整数
	short	16ビット整数
	int	32ビット整数
	long	64ビット整数
浮動小数点数型	float	32ビット符合付き浮動小数点数
	double	64ビット符合付き浮動小数点数
文字型	char	文字（16ビットUnicode文字）
真偽値型	boolean	true（真）、false（偽）

図 2.1　基本データ型

byte、short、int、long の 4 つは整数を表すデータ型、float と double は小数点を表現できる浮動小数点数、char 型は 1 文字を表す文字型、boolean 型は、正しい正しくないをあらわす true、false という値を扱うためのデータ型です。

変数を使うためには、以下のように変数をあらかじめ宣言しておく必要があります。

```
1    データ型　変数名；
```

データ型は他のプログラミング言語と同様に扱うことができ、以下のように書けます。

Listing 2.2　ソースコード 2

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          int i;
7          i = 1;
8          float f = 1.0f;
9          char c = 'a';
10         boolean b = true;
11
12         System.out.println(i);
13         System.out.println(f);
14         System.out.println(c);
15         System.out.println(b);
16     }
17 }
```

結果は以下のようになります。

```
1 1
2 1.0
3 a
4 true
```

また、他の言語と同様に演算子を扱うことができます。

Listing 2.3　ソースコード 3

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          int a = 2 + 1; // 足し算
7          int b = 2 - 1; // 引き算
8          int c = 2 * 1; // 掛け算
9          int d = 2 / 1; // 割り算
10         int e = 2 % 1; // 余り
11
12         System.out.println(a);
13         System.out.println(b);
14         System.out.println(c);
15         System.out.println(d);
16         System.out.println(e);
17     }
18 }
```

結果は以下のようになります。

```
1 3
2 1
3 2
4 2
5 0
```

・ 演習 1

好きな数字を半径とした円の面積、球の体積を求めてちょ

（ヒント）円周率は、「Math.PI」と記述することで使用できます。

（ヒント）型は double を使用してください

2.4　文字列

char では一文字しか格納できません。そこで、String を使うことで文字列を使うことができます。

ただし、String は先ほど述べた基本データ型ではない**オブジェクト型**というものです。オブジェクト型は扱いが少し難しいので、とりあえず今日は基本データ型と違うものだと思ってください。

Listing 2.4　ソースコード 4

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          String s = ”虚無”;
7
8          System.out.println(s);
9      }
10 }
```

虚無が出力されるはずです。

文字列同士の連結や文字列と数値の連結も可能です。

Listing 2.5　ソースコード 5

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          String s = "Monster";
7          String s2 = "Energy";
8          int num = 0;
9
10         System.out.println(s + s2);
11         System.out.println(s + num);
12     }
13 }
```

結果は以下のようになります。

```
1 MonsterEnergy
2 Monster0
```

2.5　条件判断

他の言語と同様に if 文を使用できます。

ただし注意が必要です。基本データ型で比較する際は「==」を使いますが、オブジェクト型を比較するときは「equals」を使いましょう。

Listing 2.6　ソースコード 6

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          int i1 = 1;
7          int i2 = 2;
8
9          if(i1 == i2){
10             System.out.println （”一緒だよ”）;
11         }
12         else{
13             System.out.println （”違うよ”）;
14         }
15
16         String s1 = "Monster";
17         String s2 = "Monster";
18
19         if(s1.equals(s2)){
20             System.out.println （”一緒だよ”）;
21         }
22         else{
23             System.out.println （”違うよ”）;
24         }
25     }
26 }
```

```
1 一緒だよ一緒だよ
```

switch 文も同様に扱えます。

Listing 2.7　ソースコード 7

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          int i = 2;
7
8          switch(i){
9              case 1:
10                 System.out.println （”月”）;
11                 break;
12             case 2:
13                 System.out.println （”火”）;
14                 break;
15             case 3:
16                 System.out.println （”水”）;
17                 break;
18             case 4:
19                 System.out.println （”木”）;
20                 break;
21             case 5:
22                 System.out.println （”金”）;
23                 break;
24             case 6:
25                 System.out.println （”土”）;
26                 break;
27             case 7:
28                 System.out.println （”日”）;
29                 break;
30
31         }
32     }
33 }
```

2.6　繰り返し構造

for 文、while 文も同様です。

Listing 2.8　ソースコード 8

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6          int i = 0;
7
8          for(i=0; i<=10; i++)
9          {
10             if((i % 3) == 0) // がの倍数なら出力 i3
11             {
12                 System.out.println(i);
13             }
14         }
15     }
16 }
```

Listing 2.9　ソースコード 9

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        // Your code here!
        int i = 0;

        while(i<=10)
        {
            if((i % 3) == 0) // がの倍数なら出力 i3
            {
                System.out.println(i);
            }

            i++;
        }
    }
}
```

・演習 2

好きな数字の階乗を繰り返し構造を用いて生成するプログラムを書いてください

・演習 3

以下のような出力を出すプログラムを書いてください

```
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz
     Buzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26,
    Fizz, 28, 29, Fizz Buzz, 31, 32, Fizz, 34, Buzz, Fizz, ...
```

・演習 4

1〜100 までの数字のうち素数のみを出力するプログラムを書いてください

・演習 5

好きな数字の平方根を計算するプログラムを書いてください

2.7　例外処理

プログラミングにおけるエラーはコンパイル前に防げないものもあります。**例外（Exception）**と呼ばれる、プログラミング実行時に起きるエラーもあります。

以下のコードを例に見てみましょう。

Listing 2.10　ソースコード 10

```
1  import java.util.*;
2
3  public class Main {
4
5      public static void main(String[] args) throws Exception {
6
7          String str = "A123";
8
9          int num = 0;
10
11         // Integer.parseInt は () の中身を int 型に変更する関数
12         num = Integer.parseInt(str);
13
14         System.out.println(num);
15     }
16 }
```

このコードを実行すると「Compile Error」ではなく、「Runtime Error」と表示されます。これはコンパイルは成功したが、実行時にエラーが起こったと言っています。このエラーは string 型を int 型に変換する際に int 型には変換できない「A」という文字が混ざっていたことが原因です。（A を消せば実行できるようになります）

このように例外が起こる可能性がある処理をするときは例外処理と呼ばれる try-catch 文を使います。

Listing 2.11　ソースコード 11

```
1  import java.util.*;
2
3  public class Main {
4
5      public static void main(String[] args) throws Exception {
6
7          String str = "A123";
8
9          int num = 0;
10
11         // try の中には例外が発生するかもしれない処理を書く
12         try
13         {
14             // Integer.parseInt は () の中身を int 型に変更する関数
15             num = Integer.parseInt(str);
16             System.out.println(num);
17         }
18         // catch の中には例外が発生した場合の処理を書く
19         catch(NumberFormatException e)
20         {
21             System.out.println(str + " はint 型に変換できないぞい");
22         }
23
24     }
25 }
```

2.8　配列

他のプログラミング言語と同様に配列を扱うことができます。

Listing 2.12　ソースコード 12

```
1  import java.util.*;
2
3  public class Main {
4
5      public static void main(String[] args) throws Exception {
6
7          // データ型 [] 配列名= new データ型 [ 要素数];
8          int[] a = new int[2];
9          a[0] = 0;
10         a[1] = 1;
11
12         // データ型配列名= { 配列の中身}
13         int[] b = {2, 3, 5};
14
15         int i = 0;
16
17         // 配列名 [ 番号で指定した番号のデータを扱える]
18         System.out.println(a[0]);
19
20         for(i=0; i < b.length; i++){
21             System.out.println(b[i]);
22         }
23     }
24 }
```

2.9　コレクション

Java では、複数のデータをまとめて扱うためのクラスがあります。これらのことを**コレクションクラス**といいます。コレクションクラスのベースは以下のようになっています。

クラス	インターフェース	説明
ArrayList	List	要素数を変更可能な配列。要素を番号で管理している
LinkedList	List	ArrayListと同じく要素数を変更可能な配列。要素を数珠つなぎのリストとして管理している
HashSet	Set	基本的な集合
TreeSet	Set	順序を持つ集合
HashMap	Map	連想配列
TreeMap	Map	キーをソートした状態で格納する連想配列
LinkedHashMap	Map	追加順を維持する連想配列

図 2.2　コレクションクラス

ここでは Andoid アプリを作る上で特につかうことの多い、ArrayList と HashMap について解説します。

・ArrayList

ArrayList は要素数が可変の配列のようなイメージです。

Listing 2.13　ソースコード 13

```
1  import java.util.*;
2
3  public class Main {
4
5      public static void main(String[] args) throws Exception {
6
7          // ArrayList< 型> 変数名= new ArrayList< 型>( 初期要素数)
8          // 初期要素数と後ろの型名は省略可
9          ArrayList<String> list = new ArrayList<>();
10
11         // 最後の要素に追加
12         list.add("nemui");
13         list.add("wakaru");
14
15         // 指定した番号の要素を削除
16         list.remove(0);
17
18         // 指定した番号の要素を使用する
19         System.out.println(list.get(0));
20
21         // 全要素削除
22         list.clear();
23     }
24 }
```

・HashMap

キーと値のペアでデータを管理するデータ構造に連想配列というものがあります。連想配列を扱ううえで最も基本的なのが HashMap です。

Listing 2.14　ソースコード 14

```
1  import java.util.*;
2
3  public class Main {
4
5      public static void main(String[] args) throws Exception {
6
7          // HashMap< 型 1, 型 2 > 変数名= new HashMap< 型 1, 型 2 >()
8          // 後ろの型名は省略可
9          HashMap<String, String> map = new HashMap<>();
10
11         // 要素を追加
12         map.put("keio", " 慶應");
13         map.put("waseda", " 早稲田");
14
15         // 指定したキーの要素を削除
16         map.remove("waseda");
17
18         // 指定したキーの要素を使用する
19         System.out.println(map.get("keio"));
20
21         // 全要素削除
22         map.clear();
23     }
24 }
```

・課題 6

作った配列に対して逆順にした配列を作るコードをを書いてください

・課題 7

作った配列に対して小さい順にソートした配列を作るコードを書いてください

2.10　クラスについて (興味のある人のみ)

クラスを変数のように扱うことができます。クラスに書き込んだ情報をもとにオブジェクト（コピー）を作ることを**インスタンス化**といいます。

以下のコードを見てみましょう。

Listing 2.15　ソースコード 15(Sub.java)

```
1  import java.util.*;
2
3  public class Sub{
4
5      public int num;
6
7      public void print(){
8          System.out.println(num);
9      }
10 }
```

Listing 2.16　ソースコード 15(Main.java)

```
1  import java.util.*;
2
3  public class Main {
4
5      public static void main(String[] args) throws Exception {
6
7          // クラス名　オブジェクト名　= new クラス名 ()　でそのクラスのイ
               ンスタンスを生成
8          Sub s = new Sub();
9
10         // オブジェクト名変数名.
11         s.num = 2;
12
13         // オブジェクト名メソッド名.
14         s.print();
15
16     }
17 }
```

このコードでは Main.java のなかで Sub.Java で定義した情報をもとにしたオブジェクトを作成して、そのなかの変数とメソッドを使っています。このように「 クラス名　オブジェクト名　= new クラス名 ()」とすることでクラスで定義した型の値やメソッドを呼び出すことができます。最初は難しいと思うのでこういうものだとおもってください。講義では扱いませんが、よりちゃんと理解したい人向けに詳しい説明を書いておきます。

同じクラスをもとにしたオブジェクトを複数作成することも可能です。

```
1 Sub s1 = new Sub();
2 Sub s2 = new Sub();
```

Java にはユーザーが自由に使える便利なクラスである**クラスライブラリ**というものが用意されています。例えば Math クラスの中には、平方根を求めるメソッドである「Math.sqrt()」というものがあります。

第 3 章

Android Studio について

（さっさと Android Studio 触らせろだと思うので）簡単な用語についてだけ解説します。

3.1　用語について

・JDK(Java Development Kit)

java アプリケーション開発に必要な実行環境や開発ツール群が同梱されています。

・Android SDK(Android SoftWare Development Kit)

Google にっよって提供されている Android アプリケーション開発のための開発キットです。コマンドラインツールやライブラリといったツール群、ドキュメント、サンプルプログラムなどが含まれます。また、AVD(Android Virtual Device) というエミュレーターが用意されています。

SDK は、Android の進化に伴ってバージョンが上がっています。バージョンごとに利用可能な API を一意に識別する **API レベル**（後述）というものがあります。

・AVD(Android Virtual Device)

Android 端末のエミュレータ

3.2　API レベルについて

先ほど説明したとおり、Android バージョンに対応した数字のことを API レベルといいます。

この講座では扱いませんが、API レベルごとに処理を分岐させることで様々なバージョンの端末に対応することが可能です。

本講座では、Android8.0 に対応している API レベル 26 で開発していきます。

Platform Version	API Level	VERSION_CODE	Release Date
Android 10.0(9.1?)未定	29	P	2019予定
Android 9.0	28	Pie	2018/08/06
Android 8.1	27	Oreo	2017/12/05
Android 8.0	26	Oreo	2017/08/21
Android 7.1	25	Nougat	2016/10/04
Android 7.0	24	Nougat	2016/08/23
Android 6.0	23	Marshmallow	2015/10/06
Android 5.1	22	LOLLIPOP_MR1	2015/03/09
Android 5.0	21	LOLLIPOP	2014/12/02
Android 4.4W	20	KITKAT_WATCH	2014/07/21(Wearables Only)
Android 4.4	19	KITKAT	2013/10/31
Android 4.3	18	JELLY_BEAN_MR2	2013/07/24
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1	2012/12/13
Android 4.1, 4.1.1	16	JELLY_BEAN	2012/07/09
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1	2011/12/16
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH	2011/10/19
Android 3.2	13	HONEYCOMB_MR2	2011/07/15
Android 3.1.x	12	HONEYCOMB_MR1	2011/05/10
Android 3.0.x	11	HONEYCOMB	2011/02/22

図 3.1　API レベル

第 4 章

ほんへ

4.1 Hello World

プロジェクトを立ち上げると「activity __ main.xml」と「MainActivity.java」というものがあると思います。

冒頭で説明したアクティビティが「activity __ main.xml」、サービスが（ここでは）「MainAcitivity.java」になります。

Run すると Hello World ができると思います。

MainActivity.java 内のコードを説明します。

```
1     @Override
2     protected void onCreate(Bundle savedInstanceState) {
3         super.onCreate(savedInstanceState);
4         setContentView(R.layout.activity_main);
5     }
```

後述しますが、onCreate というメソッドはアクティビティが生成されたときに呼ばれるものです。

一行目の「super.onCreate(savedInstanceState);」はとりあえず気にしないでください。

二行目のメソッドで「activity __ main.xml」を使用してアクティビティの画面を設定しています。

4.2　プロジェクト・ツールウィンドウの階層構造

Andoroid Studio では、一つのプロジェクト内での複数のアプリケーションやライブラリを管理できます。個々の要素のことを**モジュール**といいます。デフォルトでは「app」というモジュールが用意されています。ここでは app 内の主な要素について解説します。

・manifests

「AndroidManifest.xml」には、アプリケーション名やアクティビティ名など様々な情報が設定されています。

・Java フォルダ

アプリケーション内で使用する java ファイルが保存されています。

・Res フォルダ

様々なリソースが保存されています。

・drawable フォルダ
画面に表示するイメージファイルなどを保存するフォルダです。

・layout フォルダ
Android では、アクティビティのレイアウトを XML という形式のレイアウトファイルとして管理しています。このフォルダではそのレイアウトファイルを保存するフォルダです。

・menu フォルダ（デフォルトではないかもしれません）
メニュー画面を構成する XML ファイルを保存するフォルダです。

・minmap フォルダ
drawable と同じくイメージファイル保存用のフォルダです。イメージのサイズを変更するアニメーションなどの際に使用されます。

4.3　アクティビティのライフサイクル

アクティビティの状態が線にするにつれて、Activity クラスに用意されている対応するメソッドが呼び出されます。

メソッド	呼ばれるタイミング
onCreate()	アクティビティが生成されたとき
onStart()	アクティビティが表示される直前
onResume()	アクティビティがユーザーとやりとりする直前
onPause()	別のアクティビティが開始されるなどして一時停止状態になる直前
onStop()	アクティビティがバックグラウンドに移行し停止状態になる直前
onDestroy()	アクティビティが破棄される直前
onRestart()	アクティビティが再開される直前

図 4.1　メソッド一覧

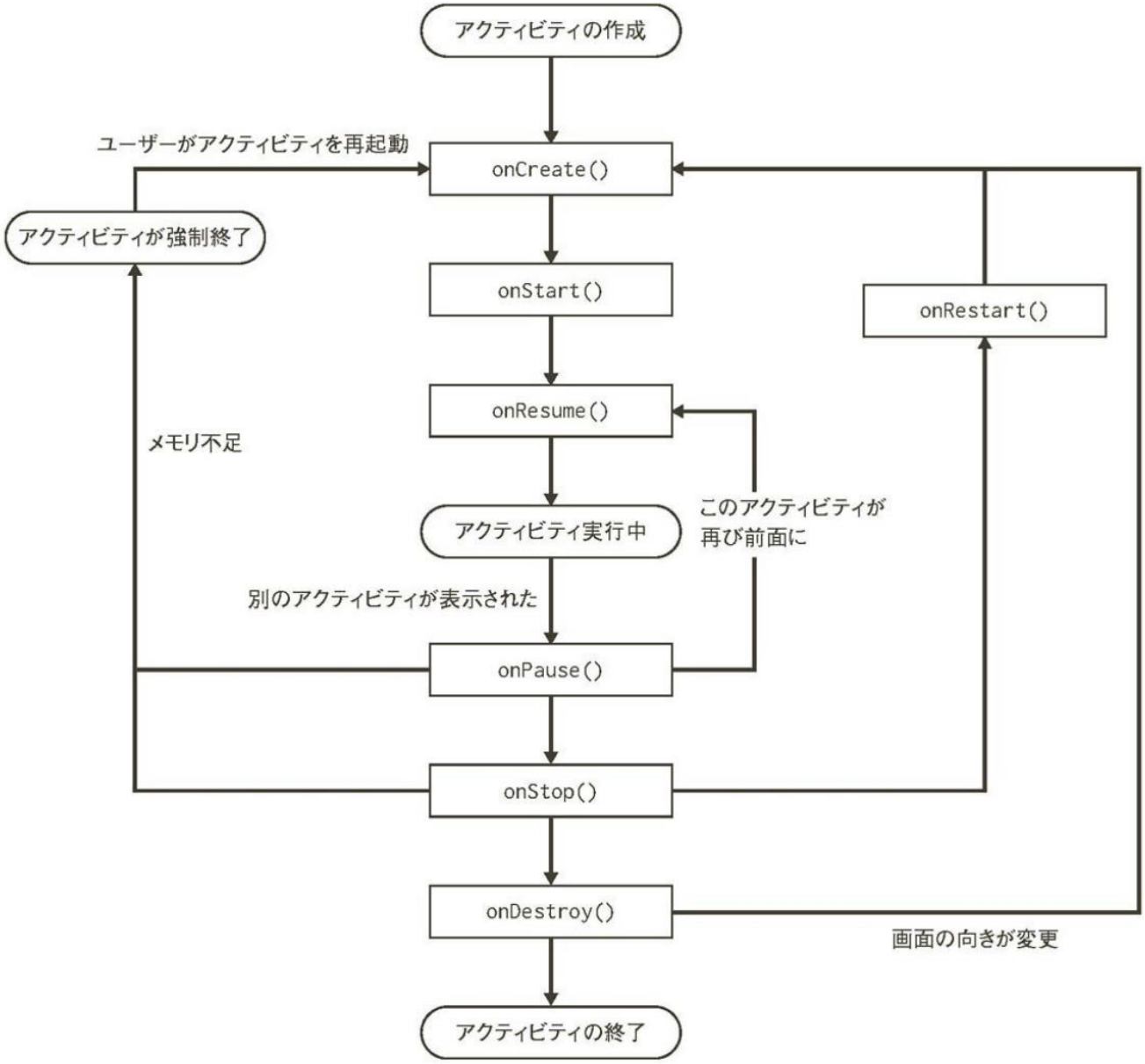

図 4.2　アクティビティのライフサイクル

```
1     @Override
2     protected void onCreate(Bundle savedInstanceState) {
3         super.onCreate(savedInstanceState);
4         setContentView(R.layout.activity_main);
5     }
6
7     @Override
8     protected void onDestroy() {
9         super.onDestroy();
10    }
11
12    @Override
13    protected void onPause() {
14        super.onPause();
15    }
16
17    @Override
18    protected void onRestart() {
19        super.onRestart();
20    }
21
22    @Override
23    protected void onRestoreInstanceState(Bundle savedInstanceState) {
24        super.onRestoreInstanceState(savedInstanceState);
25    }
26
27    @Override
28    protected void onResume() {
29        super.onResume();
30    }
```

4.4　GUi 部品

4.4.1　・TextView

Hello world は TextView という画面に文字を表示する GUI 部品を使っています。TextView の内容をコードで変更してみましょう。

```
1     @Override
2     protected void onCreate(Bundle savedInstanceState) {
3         super.onCreate(savedInstanceState);
4         setContentView(R.layout.activity_main);
5
6         // 指定したをキャストする TextView
7         TextView textview = (TextView) findViewById(R.id.textview);
8         // に TextView()内の内容をセットする
9         textview.setTextわたてん ("");
10    }
```

4.4.2　・Button

ボタンという GUI 部品を使ってみましょう。
ボタンが押されたことを検知するメソッドは以下のようになっています。
setOnclicklistener のところはかなり難しいのでこういう呪文だと思ってください。

```
1     TextView textview;
2
3     @Override
4     protected void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.activity_main);
7
8         // 指定したをキャストする TextView
9         textview = (TextView) findViewById(R.id.textview);
10        // に TextView()内の内容をセットする
11        textview.setText(" わたてん");
12
13                // 指定したをキャストする Button
14        Button button = (Button) findViewById(R.id.button);
15
16        // が押されたときに呼ばれる関数 button
17        button.setOnClickListener(new View.OnClickListener(){
18            public void onClick(View v){
19                textview.setText(" 神");
20            }
21        });
```

・課題 7

ボタンを押すたびにカウントダウンするアプリを作ってください。

・課題 8

キーボードを作ってください。

4.4.3　・エラーメッセージ

エラーメッセージは以下のように出します。

```
1 Toast.makeText(this, "example", Toast.LENGTH_LONG).show();
```

4.4.4　・Edittext

EditText という GUI を使ってみましょう。
これを使うことでユーザーが欄内に書いた内容を取得することができます。

```
1     TextView textview;
2     EditText editText;
3     @Override
4     protected void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.activity_main);
7
8         // 指定したをキャストする TextView
9         textview = (TextView) findViewById(R.id.textview);
10        // に TextView()内の内容をセットする
11        textview.setTextわたてん ("");
12
13        editText = (EditText) findViewById(R.id.editText);
14
15        // 指定したをキャストする Button
16        Button button = (Button) findViewById(R.id.button);
17
18        // が押されたときに呼ばれる関数 button
19        button.setOnClickListener(new View.OnClickListener(){
20            public void onClick(View v){
21                textview.setText(editText.getText());
22            }
23        });
24    }
```

これを使えば、簡単な電卓等作れるようになります。

・課題 9

身長から標準体重を計算するアプリを作ってみましょう

（ヒント）　取得した文字列は数字以外が含まれる場合もあります

・課題 10

身長体重から BMI を計算するアプリを作ってみましょう

4.4.5　・回転に関して

前に作ったアプリを回転させてみましょう。

書き込んでいた内容などがリセットされると思います。

これは Android の使用上、画面の向きが変わると再起動するという仕様があるからです。

回転不可の設定にするか、回転する前の状態を保存して再起動後にを読み込むことで、これを回避できます。

今回は前の状態を保存して、再起動時にそれを読み込んでみましょう。

```
1     @Override
2     protected void onRestoreInstanceState(Bundle savedInstanceState) {
3         super.onRestoreInstanceState(savedInstanceState);
4         String s = savedInstanceState.getString("Key");
5         textview.setText(s);
6     }
7
8     @Override
9     public void onSaveInstanceState(Bundle outState) {
10        super.onSaveInstanceState(outState);
11        outState.putString("Key", editText.getText().toString());
12    }
```

・課題 11

ToDo リストを作ってみましょう

参考文献

・大津 真 (2015) 「Java からはじめよう Android プログラミングー Android Studio 対応版」