標準ライブラリ

2018/6/12(火)　プログラミングIV 第九回

福井大学　工学研究科　情報・メディア工学専攻

長谷川達人

復習
コレクション

複数オブジェクトを管理する**コレクション**は主に3つに分類できる.

- **リスト**（implements List）
  - 要素の並び順に意味のある場合に使用する（配列に近い）.
  - **ArrayList**，**LinkedList**等がある.
- **マップ**（implements Map）
  - キーと値（オブジェクト）をペアで管理する場合に使用する.
  - **HashMap**やLinkedHashMap，TreeMap等がある.
- **セット**（implements Set）
  - 要素の並び順に意味がなく，重複を許さない場合に使用する.
  - HashSetやTreeSet等がある.

復習
コレクション

コレクションを扱う際などに，後から決まる型のことを**ジェネリクス型（総称型）**と呼び，このような仕組み自体を**ジェネリクス**と呼ぶ．以下のように<>で型を記述する．

```
ArrayList<Monster> list = new ArrayList<Monster>();
```

コレクション
ラッパークラス

（復習）基本型：intやdouble等，値を直接格納する型
参照型：インスタンスや配列等，参照値を格納する型

ラッパークラス（Wrapper Class）は，基本型を参照型で包み込んだ（Wrapした）クラスである．色々と便利なメソッドを持つ．

基本型
int型やdouble型
値を直接持つ

↓ ラッパークラス

参照型
Integer型やDouble型
インスタンスになる

```
// 普通のint型（値がそのまま入っているだけ）
int i1 = 10;
System.out.println(i1);

// Integer型（クラスなのでメソッドが使える）
Integer i2 = new Integer(10);
System.out.println(i2.intValue());
System.out.println(i2.compareTo(9));

// String->intの変換は実はIntegerクラス
int i3 = Integer.parseInt("199");
```

int型の値を1つ持つだけのクラス

復習
イテレータ

イテレータとは何か？という
ことだけ知っておいてほしい.

拡張for文が使える＝Iterable<E>をimplementsしている

例えばArrayListはimplements **Iterable<E>**である.
＞したがって，iterator()メソッドが使える.
＞＞iterator()の戻り値の**Iteratorインスタンス**は，**next()**や
hasNext()メソッドが使える.
＞＞＞以下の様に要素を順番に取得できる（＝拡張for文）.

```
Iterator<Integer> it = list.iterator();
while(it.hasNext()){                    // 次の要素があるか確認
    System.out.println(it.next());  // 次の要素を取得
}
```

すなわち，イテレータとは次の要素にアクセスできる機能のこと

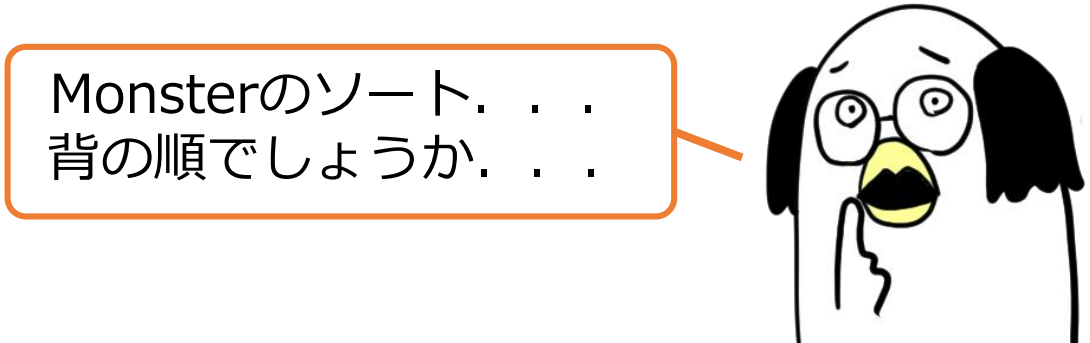
復習
ソート

リストの中身をソートするにはCollections.sort()を使う.

```
ArrayList<String> list = new ArrayList<String>();
Collections.sort(list);
```

数値や文字列の場合は標準で定義されたCompalatorが昇順を定義しているので昇順に並べてくれる.

では，Monster型をソートしたい場合どうすればよいだろうか？
文字列型を独自順序でソートしたい場合どうすればよいだろうか？



復習
ソート

独自のソートを行うには，**何を基準に大小関係を決定するのか**を定義して，**Collections.sort()に教えてあげる**必要がある.

パターン１：Collections.sort(list);

listの要素が**Comparableな場合**
＞list = new ArrayList<Integer>();の場合（IntegerはComparable）
＞list = new ArrayList<Monster>();の場合でも，
MonsterにComparableをimplementsさせればこれに該当する.

パターン２：Collections.sort(list, comparator);

listの要素に関わらず**独自の基準comparatorを明示する場合**
＞listの要素の型＝comparatorの要素の型として自作のcomparatorを定義し，インスタンス化したものを第二引数とする.
＞具体的な使用例は前回の演習２の解答例を参照する.

演習問題の解説

- HP上にアップロードした.
- 点数に**不服がある人**は個別に申し立ててください.
- **第4回**，**第7回**の採点結果に意図しないパターンが混在していたため，再採点しました．両者ともに点数が上がっている人が一部いると思いますのでご確認ください.
- 講義回の提出時に**第N回**を間違える人がちらほら
- 前回の解説を今から行います.
  - びっくりするほど**ケアレスミス**？が多かった.

質問への回答

- ArrayListにまとめてadd()できないのか.
  - できる．addAll(Collection c);でCollection型のcをまとめてリストに追加できる．ただしまとめての側もCollection型なので，そちらに代入しておかねばならない．２つのリストを結合したいときなどに使う.
- ArrayListは現場ではどのように使われるのか.
  - 配列の要素数を事前に決めづらい場合積極的に使われる.
  - ArrayListが**常識**かつ超便利なので，配列なんて使わないかも.
- ArrayListを早めに教えてくれなかったのはなぜか.
  - Javaで最重要なのは，継承，カプセル化，多態性である．その上でGUIを先に学ぶと学習意欲が向上すると判断したため.

質問への回答

- ジェネリクス型に配列を指定できるのか.
  - 配列は参照型なのでラッパークラスを使わずにできる.
  - ArrayList<int[]> array = new ArrayList<int[]>();
- HashMapはいつ使うのか．ハッシュ関数を通してハッシュ値を算出していて使いにくそう.
  - 使う分にはハッシュ関数～～といった内部動作は気にしなくて良い．連想配列のように文字列等をキーにして値にアクセスしたいときに使う．というか本日の演習課題でも使う.
- ガーベージコレクションとコレクションは何か関係が？
  - ない.

質問への回答

Key	Value
MacBookAir	89000
MacBookPro	129000
iPad	37000

- Mapだけiterableじゃないのがよくわからなかった.
  - Mapは右表のようなKey-Valueのペアを管理する．これがiterableで一つ一つの要素が取り出せるとするとKeyから出せばいいのかValueから出せばいいのか…等よくわからない.
  - したがって，**Keyのみ**や**Valueのみ**，もしくは**Key-Valueのペアー一つを示すEntry型インスタンス**であればiterableで出力できるが，Map自体はiterableではないということになっている.
- Mapに対して[]でアクセスできないのか？
  - map[“aaa”]みたいなことはできるのか？という意図だと思うが，残念ながらできない．PHPとかだとできるよね.

質問への回答

- String型はラッパークラスか？
  - String型だけは少々特殊でプリミティブ型のように使えるが，参照型クラス．ラッパークラスを兼ねてると考えても良い．
- intやdouble以外のラッパークラスってあるの？
  - プリミティブ型には全部ある．Integer以外は全て先頭を大文字にしただけのクラスである．LongやFloat，Boolean等．
- コレクションは配列の完全上位互換なのか．
  - なんとも言えないところだが，int型とInteger型の関係に近いものはある．便利な分，シンプルな配列に速度が劣る可能性も．

質問への回答

- 降順でソートするには？
  - Comaratorのcompare()実装時に＋と－を反転させれば良い.
- 普通の配列も拡張forできるのはiterableだから？
  - 普通の配列だけは例外（iterableではない）
- 複数のcompareTo()を任意に分岐させられないか.
  - たぶんできない.
- 要素数がわかれば無理に拡張for文使わなくても良い？
  - そのとおり．拡張for文を使えることはかっこいいけど，全てを拡張for文で書く人はよろしくない．使い分けが重要である.
  - ただ，MapやSetは拡張for文じゃないと書けない.

質問への回答

- 楽しい（お薦めの）プログラム言語があったら教えて.
  - 長谷川研でお馴染み今流行りの機械学習（人工知能技術の一つ）を使いたいならばPython
  - Webやりたいなら，HTML＋CSS＋JavaScript＋PHP（鯖側）
  - とりあえず実用的な何か作りたいならC#
  - 売りたいならAndroid（Java）
- 演習中に周囲と相談してもよいのか.
  - 荒れ過ぎなければOK．ただし，解法を直接教えるのはNG.
- 自由課題，何考えても既存とかぶりそう．．．
  - 流石に研究ではないので既存製品にあるのでは？といった観点で検閲しない．既存にとらわれず，コレがあったら便利そう！面白そう！を探求してほしい.

質問への回答

- 自由課題で授業の内容をたくさん使っていても面白くならなければ点数は低いのか？
  - 色々な観点で総合的に評価なので，総合的に高い方が良いかも.
- 自由課題を作り始めているが完成まで気が遠くなりそう.
  - あまり大きすぎる課題設定にしないことを推奨する.
  - 開発期間は2週分(4コマ)で完成できそうな規模で良い.
  - 一人あたり発表時間は1分厳守なので重すぎても紹介できない.
- 自由課題はソースコードの提出はするのか.
  - ソースコードは提出不要．当日発表用のアプリの動作がわかる動画を埋め込んだパワポファイルを提出してもらう予定.
  - 自由課題についてはこれから例示する.

質問への回答

- お薦めの本やサイトはあるか？
  - プロIIの時から参考図書の「スッキリわかるJava入門」は非常にわかりやすく書かれており，基本～多態性までであればコレを全ページ一読することを強く推奨する.
- 期末テストはどのような対策をしたらよいか.
  - ラスボスに対して弱点の道具は何ですか？って聞いてるようなものだと思うが．．．過去資料をベースにこれまで長谷川が重要だと言ってきた内容（Javaができる人を判別する上で必須な内容）を問う予定である.
- ３回欠席したのですが優は狙えるか？
  - 前回計算式を提示しているので自分で計算しなさい.

本講義の概要

前半		後半	
第1回	基本文法の復習	第9回	**標準ライブラリ**
第2回	クラス～カプセル化の復習	第10回	ファイル入出力
第3回	抽象クラス，インタフェース	第11回	デバッグ，インポート，高速化
第4回	ポリモーフィズム	第12回	オブジェクト指向
第5回	GUI 1	第13回	自由開発演習 1
第6回	GUI 2	第14回	自由開発演習 2
第7回	スレッド，例外処理	第15回	自由開発演習発表会
第8回	ジェネリクス，コレクション	第16回	期末試験

何を開発するか，少しずつ考えておくこと

本日の目標

概要

Java開発を行う際に知っておくと重宝する
標準ライブラリの使い方を学ぶ.

目標

こういうライブラリがあったなぁ，程度に記憶し，
使いたいときに思い出せるようにする.

本日の提出課題

講義パート

課題を意識しながら
講義を聞くと良い.

課題 1

本日の授業を聞いて,
よくわかったと思う内容を2点簡潔に述べよ.

課題 2

本日の授業を聞いて,
質問事項または**気になった点**を1点以上簡潔に述べよ.

課題 3

感想（あれば）

標準ライブラリ

Math：数学メソッド

Mathクラスは様々な**数学処理**を実装している.

全てのメソッドは**static**で定義されており,
引数を渡して戻り値を得るという使い方をする.

いくつか例を紹介する.

```
// abs(): 絶対値を取得する
System.out.println(Math.abs(-100.0));

// pow(): べき乗を計算する
System.out.println(Math.pow(10.0, 3));

// max(): 最大値を取得する
System.out.println(Math.max(10, -10));
```

出力

```
100.0
1000.0
10
```

標準ライブラリ

Math：数学メソッド

ここで重要なことは，Mathクラスのメソッドを
すべて暗記することでは**ない**.

APIリファレンスを参照し，必要な時に必要なメソッドを**調べて**，**使いこなせる**ことが重要である.

Java(tm) Platform, Standard Edition 8　API仕様
https://docs.oracle.com/javase/jp/8/docs/api/overview-summary.html

TOPページから意図した場所にたどり着くのは困難なので，「Java Math」等でググるとよい.

標準ライブラリ

Math：数学メソッド

MathクラスのAPIリファレンスを読んでみよう.

https://docs.oracle.com/javase/jp/8/docs/api/java/lang/Math.html

java.langパッケージに所属

java.lang

クラスMath

java.lang.Object

　　java.lang.Math

public final class **Math**
extends Object

Objectクラスを継承している

フィールドのサマリー

数学の定数が2種類定義されている.

フィールド

修飾子と型	フィールドと説明
static double	**E** 自然対数の底*e*にもっとも近いdouble値です。
static double	**PI** 円周とその直径の比*pi*にもっとも近いdouble値です。

標準ライブラリ

Math：数学メソッド

メソッドも書いてある通り．意外とわかりやすい．

メソッドのサマリー

すべてのメソッド | staticメソッド | 具象メソッド

修飾子と型	メソッドと説明
static double	**abs**(double a) double値の絶対値を返します。
static float	**abs**(float a) float値の絶対値を返します。
static int	**abs**(int a) int値の絶対値を返します。

標準ライブラリ

String：文字列処理

Stringクラスは文字列を保持するだけでなく様々な**文字列処理**を実装している.

主に**非static**メソッドであり，**保持している文字列の値**に対して処理を行うメソッドであることが多い.

いくつか例を紹介する.

```
String str = "Mojiretsu";
// charAt(): 指定された位置のchar値文字を返す
System.out.println(str.charAt(4));
// concat(): 指定された文字列を自身の最後に結合する
System.out.println(str.concat("!!!"));
// indexOf(): 指定された文字が最初に出現する位置を返す
System.out.println(str.indexOf("ret"));
```

出力

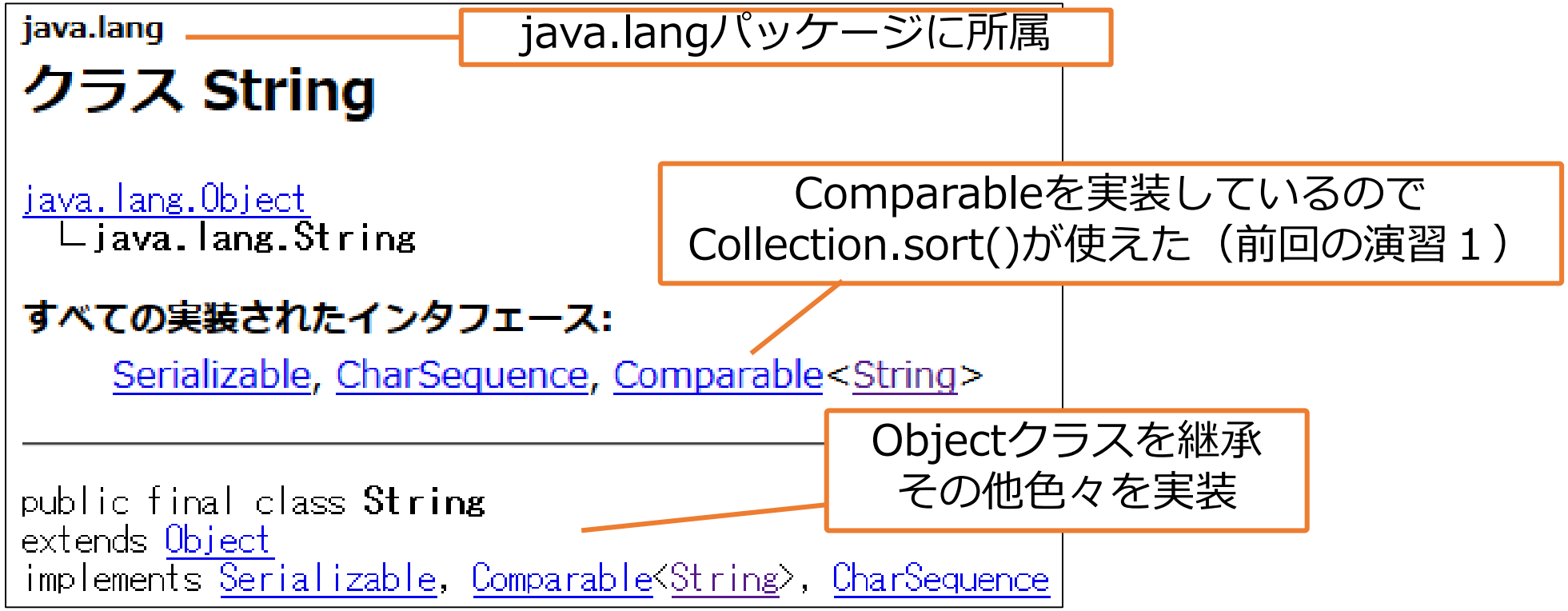
```
r
Mojiretsu!!!
4
```

標準ライブラリ

String：文字列処理

StringクラスのAPIリファレンスを読んでみよう.

https://docs.oracle.com/javase/jp/6/api/java/lang/String.html



標準ライブラリ

String：文字列処理

コンストラクタの使い方がたくさん！！

コンストラクタの概要
String() 新しく生成された String オブジェクトを初期化して、空の文字シーケンスを表すようにします。
String(byte[] bytes) プラットフォームのデフォルトの文字セットを使用して、指定されたバイト配列を復号化することによって、新しい String を構築します。
String(byte[] bytes, Charset charset) 指定された 文字セット を使用して、指定されたバイト配列を復号化することによって、新しい String を構築します。
String(byte[] ascii, int hibyte) **推奨されていません。** *このメソッドでは、バイトから文字への変換が正しく行われません。JDK 1.1 以降では、バイトから文字への変換には、引数として Charset、文字セットの名前を取る String コンストラクタ、またはプラットフォームのデフォルト文字セットを使用する String コンストラクタの使用が推奨されます。*
String(byte[] bytes, int offset, int length) プラットフォームのデフォルトの文字セットを使用して、指定されたバイト部分配列を復号化することによって、新しい String を構築します。
String(byte[] bytes, int offset, int length, Charset charset) 指定された charset を使用して、指定されたバイト部分配列を復号化することによって、新しい String を構築します。
String(byte[] ascii, int hibyte, int offset, int count) **推奨されていません。** *このメソッドでは、バイトから文字への変換が正しく行われません。JDK 1.1 以降では、バイトから文字への変換には、引数として Charset、文字セットの名前を取る String コンストラクタ、またはプラットフォームのデフォルト文字セットを使用する String コンストラクタの使用が推奨されます。*

標準ライブラリ

練習問題１：APIリファレンス

1. 引数に対して自然対数をとり，小数第一位を四捨五入した値を返すstaticメソッドintLn()を実装せよ.

2. String型のAPIリファレンスで，indexOf()の項目を読み，3文字目以降で，”HasegawaTatsuhito”の中で”a”が出てくる一番最初のindex番号を取得する処理を実装せよ.

3. 引数の文字列を半角カンマで**分割**し最初のカンマまでの文字列を返すメソッドgetFirst()を実装せよ.
   ＞例：getFirst(“aa,ii,uu”) -> “aa”

4. 引数の文字列から半角カンマをすべて削除した文字列を返すメソッドremoveComma()を実装せよ.　（ヒント：**置換**）

標準ライブラリ

Date，Calendar：日時

日時を管理するには4つの方法がある.

現在時刻と協定世界時のUTC 1970年1月1日深夜零時との差 [ms]

1．long型の数値（System.currentTimeMillis()）
　＞機械としてはこれが一番効率的でわかりやすい
　＞メモリ的にも最も低コストなので，DB保存時などは採用されやすい.

2．Date型のインスタンス（java.**util**.Date）

java.**sql**.Dateもあるので注意する

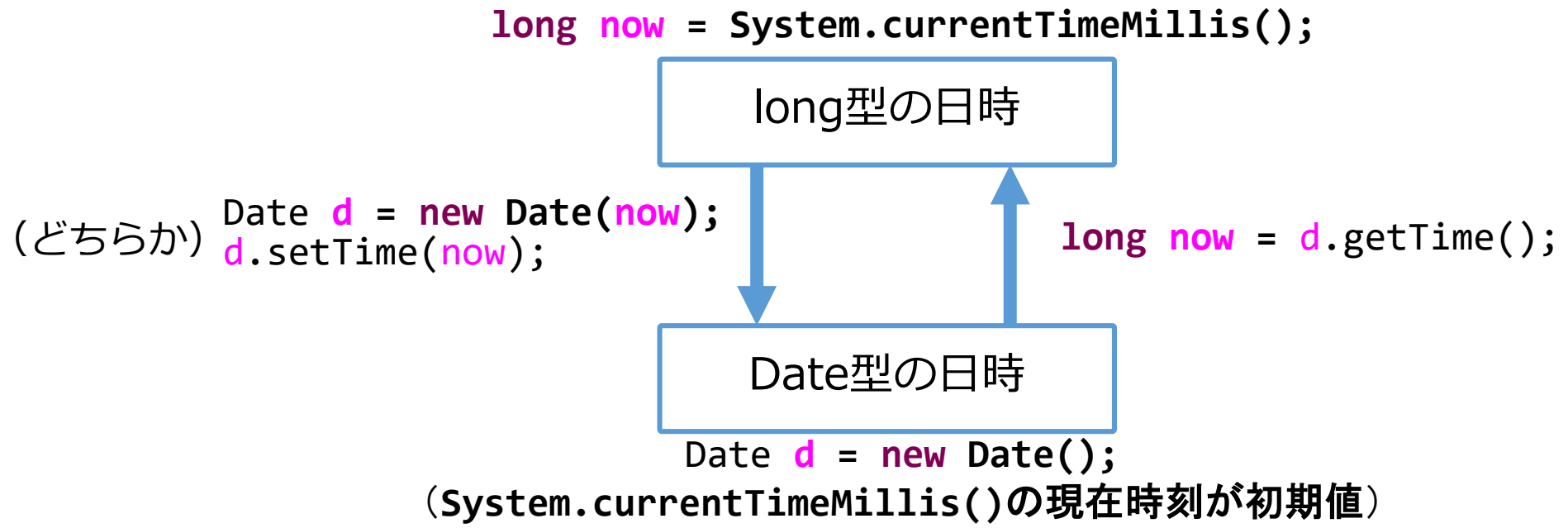
3．6つのint型の日時（2018, 4, 1, 12, 34, 56）
　＞人間としてはこれが一番計算しやすい

4．String型の文字列（”2018/04/01 12:34:56”）
　＞人間としてはこれが一番理解しやすい

標準ライブラリ

Date，Calendar：日時

一番基礎となるlong型とDate型の変換は以下の手順で実施する.



標準ライブラリ

Date，Calendar：日時

6つのint型，文字列型はDate型から変換する．

```
Calendar cal = Calendar.getInstance();
```

6つのint型
(Calendar型)

Calendarクラスを仲介

```
Date d = cal.getTime();
```

```
cal.setTime(d);
int year = cal.get(Calendar.YEAR);
```

Date型の日時

```
SimpleDateFormat sdf  = new
 SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
String str = sdf.format(d);
```

```
Date d = sdf.parse(str);
```

文字列型の日時

SimpleDateFormatクラスを仲介

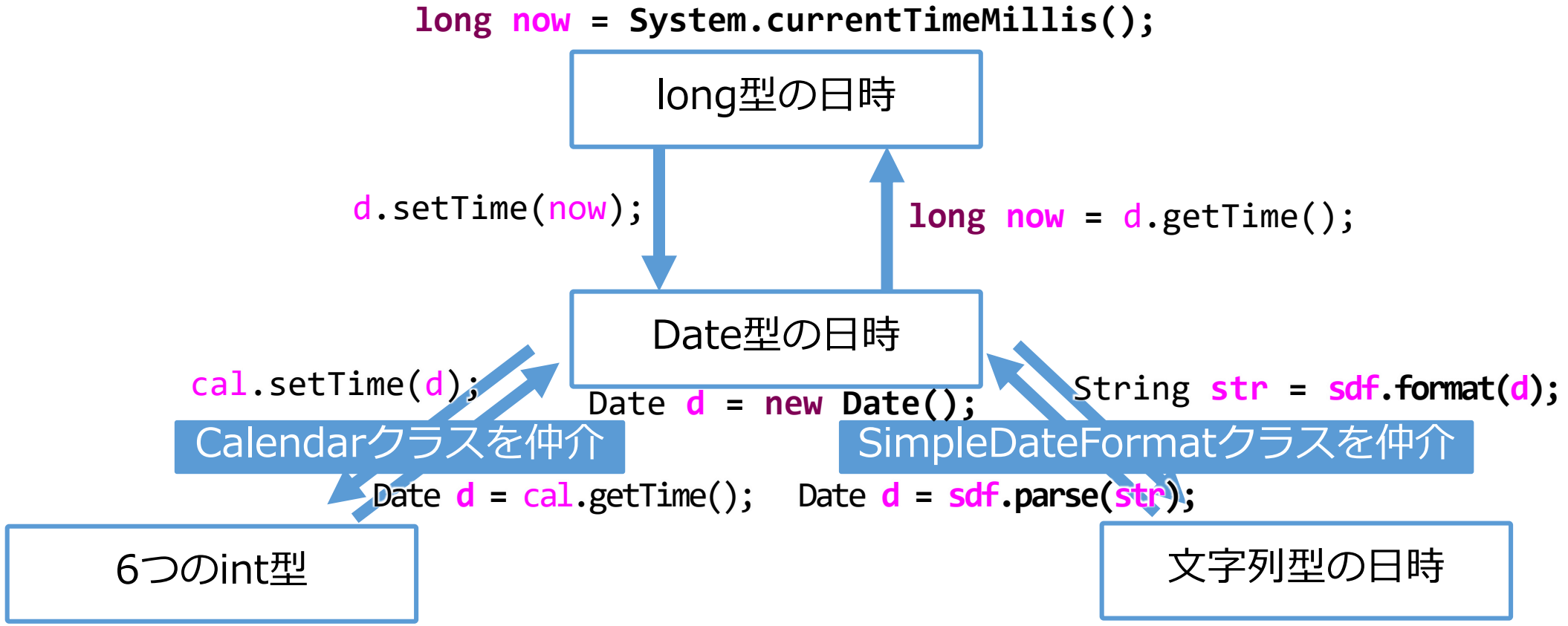
```
String str = "2018/01/01 11:11:11";
```

標準ライブラリ

Date，Calendar：日時

まとめると以下の通り．

日付の加減算はCalendarクラスでできる．詳細はググろう．



標準ライブラリ

Date，Calendar：日時　使ってみる

```
// long型の時刻を取得する
long now = System.currentTimeMillis();

// Date型に変換する
Date d = new Date(now);

// Calendar型に変換する
Calendar cal = Calendar.getInstance();
cal.setTime(d);

// 6つのint型に変換する(Monthだけは0-11なので+1する)
int year = cal.get(Calendar.YEAR);
int month = cal.get(Calendar.MONTH) + 1;
int date = cal.get(Calendar.DATE);
int hour = cal.get(Calendar.HOUR_OF_DAY);
int min = cal.get(Calendar.MINUTE);
int sec = cal.get(Calendar.SECOND);

// String型に変換する
SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
String str = sdf.format(d);
```

CalendarはCalendar.getInstance()でインスタンス化する仕様
この時カレンダの初期時刻はgetInstance()実行時の時刻となる

Date d に保存されている時刻
nowをcalにセットする処理

Monthだけ注意

出力フォーマットを指定する

標準ライブラリ

練習問題２：日付

1. 現在の日時を「2018-01-01 10:00:00」の形式で出力するメソッドnow()を実装せよ.
2. 引数のlong値を上記書式で出力するメソッドprintDate()を実装せよ.
3. 現在の日時の時(hour of day)＋分(minute)を出力するメソッドhourPlusMinute()を実装せよ（特に意味はない）.
4. 引数のlong値に対して9時間前を上記書式で出力するメソッドprintUTC()を実装せよ. UTC=Coordinated Universal Time
5. 文字列”2018_10_10!20_00_00”をDate型に変換する処理を実装せよ.

Objectクラス
全てのクラスの祖先

以下のプログラムの実行結果を予測してみよう.

Empty.java

```
public class Empty {}
```

Main.java

```
public class Main {
    public static void main(String[] args){
        Empty e = new Empty();
        System.out.println(e.toString());
    }
}
```

1.例外で強制終了

2.何も起きない

3.”Empty****”と表示

Objectクラス
全てのクラスの祖先

前ページの結果の理由は以下の約束があるからである.

暗黙の継承
クラス定義時に何も継承しない場合java.lang.Objectを継承したものとみなされる.

すなわち，前ページのEmptyは実質は以下と同義である.

```
public class Empty extends java.lang.Object{}
```

ということは，**全てのクラスは元をたどるとObjectクラスにつながっている**のである.

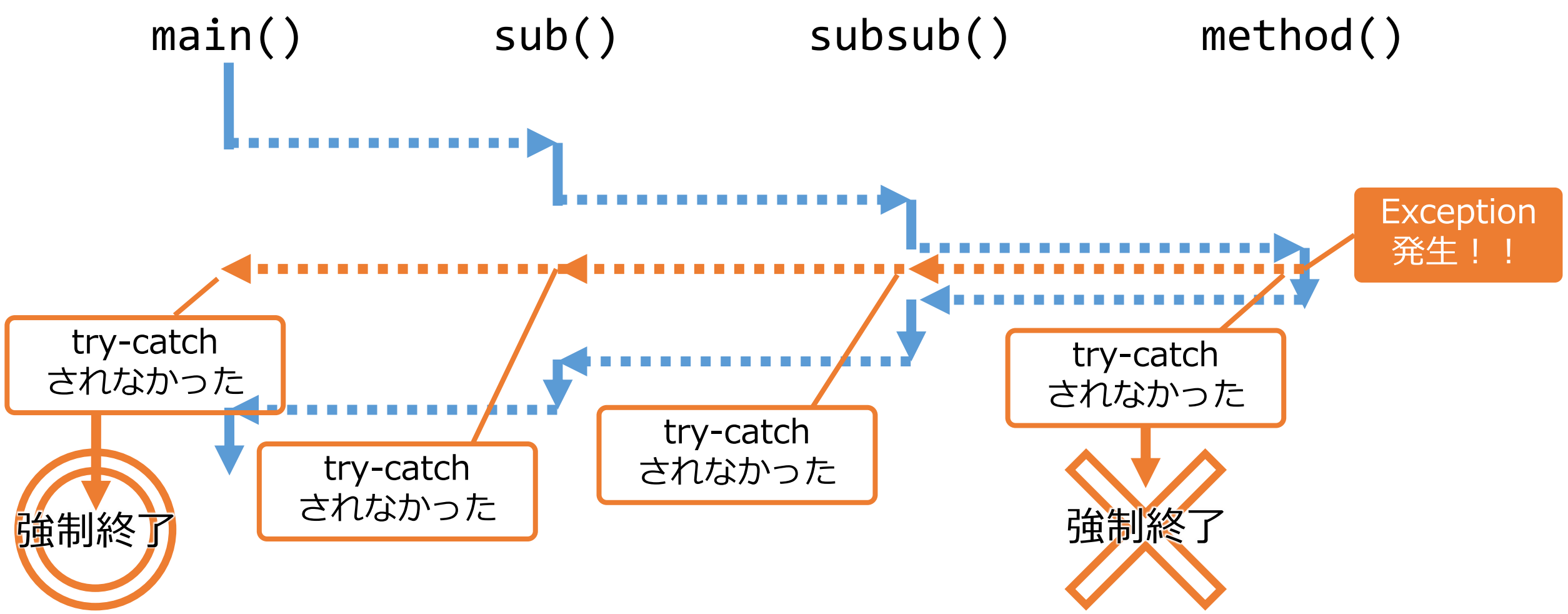
Objectクラス
全てのクラスの祖先

全てのクラスがObjectクラスを継承している事によって，次のようなメリットがある．

- **全クラスが最低限備えるべきメソッドを定義できる**．
  - toString()とequals()はObjectに記述されたメソッドなので，全クラスが必ず使えるメソッドである．
- **ポリモーフィズムが使える**．
  - 全クラス is a Objectなので，どんなクラスでもとりあえずObject型として扱える．
  - つまり，ArrayList<Object> listには何でも格納できる（危険だが）．

例外

例外は伝播する


main()
sub()
subsub()
method()
Exception
発生！！
try-catch
されなかった
try-catch
されなかった
try-catch
されなかった
try-catch
されなかった
強制終了
強制終了

例外

例外は伝播する

実際にやってみよう.

```
6  public class Main {
7    public static void main(String[] args){
8      sub();
9    }
10   public static void sub(){
11     subsub();
12   }
13   public static void subsub(){
14     method();
15   }
16   public static void method(){
17     System.out.println(100/0);
18   }
19 }
```

Exception
発生！！

出力（StackTrace）

```
Exception in thread "main"
java.lang.ArithmeticException:
/ by zero
at Main.method(Main.java:17)
at Main.subsub(Main.java:14)
at Main.sub(Main.java:11)
at Main.main(Main.java:8)
```

StackTraceでは伝播の過程を確認できるのでデバッグに使える.

例外

例外が起こりうることを定義する

メソッド自身の中で例外が発生しうることを明記することができる.

通常のメソッド定義

発生しうる例外を明記しておく

```
public static void method() throws IOException{
    // 中身は何でも良いが, IOExceptionをthrowする
    // 可能性があるのでthrows宣言をメソッドに付記する.
}
```

throwsされる例外が**チェック例外であれば**, method()を呼び出す側は必ずtry-catchしなければならなくなる.

例外
チェック例外メソッドを定義する

スロー宣言を行うと便利なことがある.

```
public static void method() throws IOException{
    // (1) ファイルを開く
    FileWriter fw = new FileWriter("c:¥¥data.txt");
    // (2) ファイルに文字を書き込む
    fw.write("hello!");
    // (3) ファイルを閉じる
    fw.close();
}
```

上記のように本来であれば**try-catch(IOException e)**しなければならないメソッドを，**try-catch**せずに使用することができる．例外が発生したら呼び出し元に対応してもらうことを**throws**宣言しているためである.

要するに，自分で対応せず**上に丸投げする**のである.

例外
例外を生成する

throwsと混同しないように注意

例外を自分自身で生成することもできる.

```
public static void setData(int data){
  if(data <= 0){
    throw new IllegalArgumentException
              ("引数dataは自然数である必要があります. data=" + data);
  }
}
```

特に強制終了するような処理ではないが，例外を意図的に発生させる.

main()でsetData(-10)を実行した結果

```
Exception in thread "main" java.lang.IllegalArgumentException: 引数
dataは自然数である必要があります. data=-10
at Main.setData(Main.java:24)
at Main.main(Main.java:10)
```

一般的に例外が発生したときのようにStackTraceが表示される

例外

例外を発生させる

Q：例外を意図的に発生させるメリットってなくない？

A：いくつかメリットはある.

1. 呼び出し元でtry-catchでまとめて対応させることができる（複数まとめてtry-catchできるので）.
2. 「正常終了したかどうか」という無駄な戻り値を返さなくて良くなる.
（＝他のことに戻り値を有効活用できる）

まとめ

- よく使う標準APIとして，Mathクラス，Stringクラス，日付関連クラスについて使い方を学習した.
- APIリファレンスを読む練習を行った.
- 例外に関する発展事項の学習を行った.
  - 例外の伝播
  - throws宣言で上に例外対応を丸投げ
  - throwで意図的に例外を生成

次週予告

※次週以降も計算機室

前半

ファイル入出力の使い方を学ぶ

後半

講義内容に関するプログラミング演習課題に取り組む.

本日の提出課題

講義パート

課題 1

本日の授業を聞いて，
よくわかったと思う内容を
2点簡潔に述べよ.

課題 2

本日の授業を聞いて，
質問事項または**気になった点**
を1点以上簡潔に述べよ.

課題 3

感想（あれば）

課題 4

なし

演習

- 昼休み，いつものWebページに演習問題をPDFで演習問題をアップロードする．各自実施してプロII同様のWebページから提出すること.
- 質問は３人体制で受け付けるので遠慮なく申し出る．質問の際は，どこまでわかっていて何がわからないのかを申し出ること.
- （ないとは思うが）コピペは発覚次第両成敗する.
  - ✓ {コピペ，カンニング} ∈不正行為
- つまらないミスも今回は問答無用で×とするので，最終チェックを怠らないこと．（昨年は目視で甘めに採点していたが，自動採点を開発している意味がないので．．．）