7章
Java APIの利用

概要

- Java言語が提供するライブラリの集合および言語規約

- よく使う機能がひとまとめにされている

もくじ

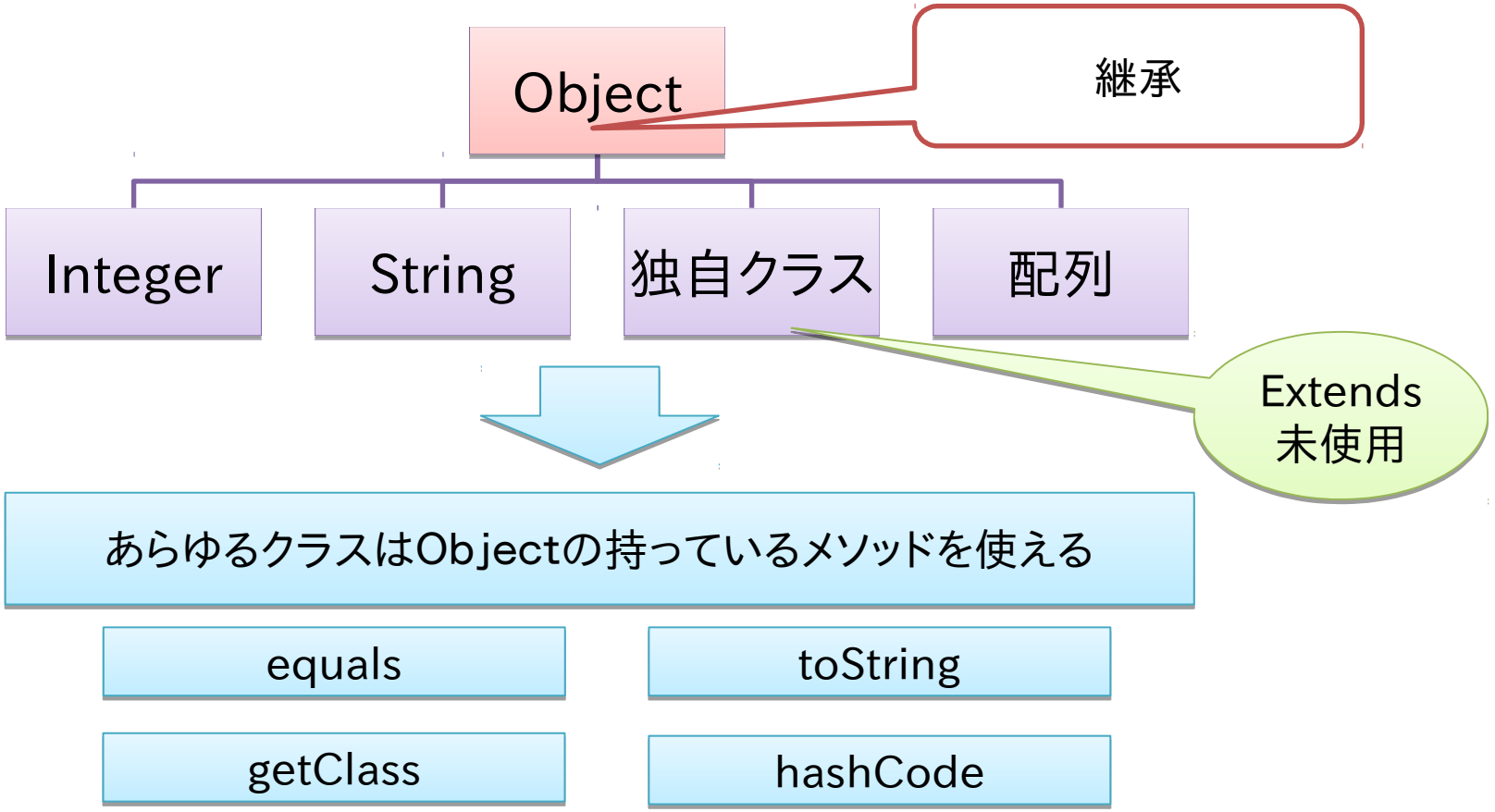
- Java.lang.Objectクラス
- Date and Time API
- コレクションとラムダ式

Java.lang.Objectクラス

- Objectクラスのメソッド
- ObjectクラスのEquals()メソッド
- ObjectクラスのtoString()メソッド
- Mathクラス
- 配列に関連するメソッド

Objectクラスのメソッド

- Java.lang.Objectクラス
  →全クラスのルートとなるクラス



ObjectクラスのEquals()メソッド

- 2つのオブジェクトを比較し、同じオブジェクトならtrueを返す

ポイント：内容ではなくオブジェクトが同じじゃないとfalse

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

class Foo {}

public class Main {
    public static void main(String[] args) throws Exception {
        Foo f1 = new Foo();  Foo f2 = new Foo();
        System.out.println("f1=f2:"+ (f1.equals(f2)));

        Foo f3 = new Foo();  Foo f4 = f3;
        System.out.println("f3=f4:"+ (f3.equals(f4)));
    }
}
```

==と同じ動作

```
f1=f2:false
f3=f4:true
```

StringクラスのEquals()メソッド

- 多くのクラスは自身の特性に合わせてEqualsをオーバーライドしている

同じオブジェクトならtrue

Stringバージョン

オブジェクトが保持する文字列が同じならtrue

```
public class Main {
    public static void main(String[] args) throws Exception {
        String s1 = "tanaka";
        String s2 = new String("tanaka");

        System.out.println("f1==f2        :"+ (s1==s2));
        System.out.println("f1.equals(f2):"+ (s1.equals(s2)));

    }
}
```

```
s1==s2        :false
s1.equals(s2):true
```

大文字小文字を無視
equalsIgnoreCase()

ObjectクラスのtoString()メソッド

- [オブジェクトのクラス名]@[オブジェクトのハッシュコード]
  を返す

overRide: Stringクラス・StringBuilderクラスでは
オブジェクト内の文字列を返す

overRide: 独自クラスでオーバーライドすることもできる

toString()メソッド：実行例

```
class Foo {}
class Bar {
    public String toString(){
        return "overRideされたtoStringです";
    }
}

public class Main {
    public static void main(String[] args) throws Exception {
        String obj1 = "tanaka";
        Foo obj2 = new Foo();
        Foo[] obj3 = {new Foo(),new Foo(),new Foo()};
        Bar obj4 = new Bar();

        System.out.println(obj1);
        System.out.println(obj2);
        System.out.println(obj3);
        System.out.println(obj4);
```

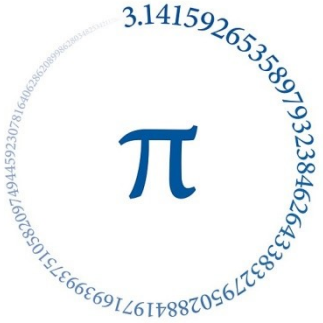
```
tanaka
Foo@77468bd9
[LFoo;@12bb4df8
overRideされたtoStringです
```

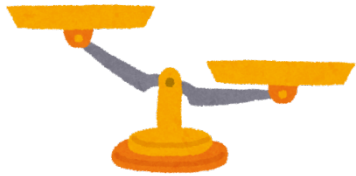
Println()メソッドに参照変数名を指定するとtoStringが呼び出される

Mathクラス

- 数値を扱うのに便利なライブラリ

Static final double E	自然対数の底eを表す
Static final double PI	円周率を表す
Static int max(int a, int b)	大きいほうの数字を返す
Static double random()	0.0以上1.0未満の 正のdouble値を返す





配列に関するメソッド

- JavaAPIでは配列を操作するメソッドが多数用意されている

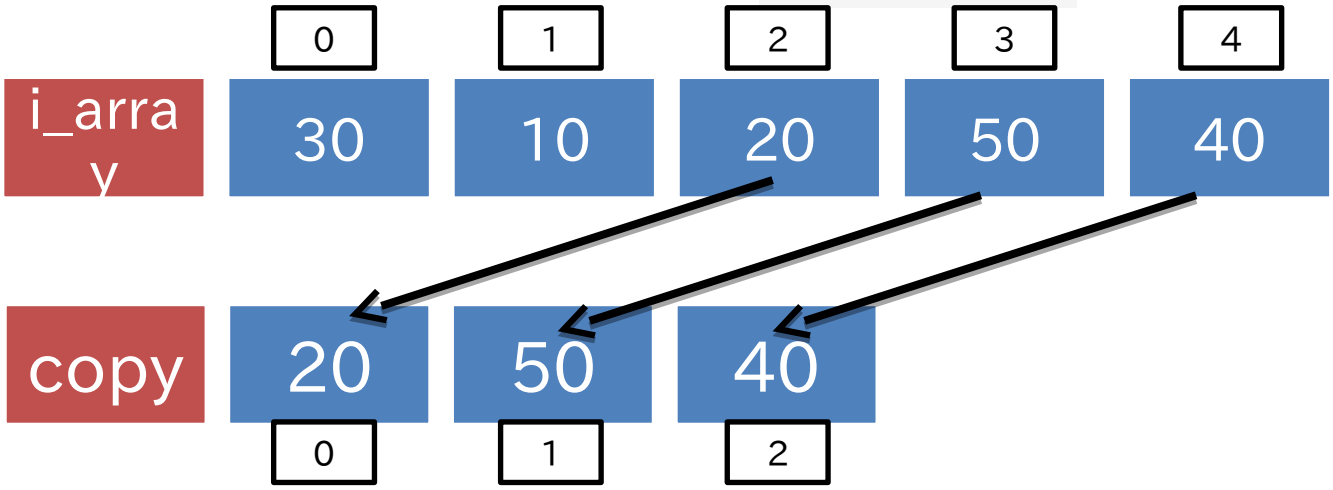
Systemクラス	Static void arraycopy()	配列を別の配列にコピーする
Classクラス	Boolean isArray()	オブジェクトが配列かを判定する
ArrayListクラス	Object [] toArray()	リスト内のすべての要素を配列に返す
Arrayクラス	Static <T> List<T> asList()	引数で指定した配列を固定サイズのリストで返す
Arrayクラス	Static void sort()	配列を昇順にソートする

arraycopy()

```
public class Main {
    public static void main(String[] args) throws Exception {
        int[] i_array = {30,10,20,50,40};
        int[] copy = new int[3];
        System.arraycopy(i_array,2,copy,0,3);
        for(int val:copy){
            System.out.print(val + " ");
        }
    }
}
```

```
20 50 40
```

i_array:コピー元
2:コピー開始位置
copy:貼り付け先
0:貼り付け開始位置
3:張り付ける配列数



sort()

```java
public class Main {
    public static void main(String[] args) throws Exception {
        int[] i_array = {30,10,20,50,40};
        String[] s_array = {"banana","orange","apple"};
        String[] s2_array = {"バナナ","オレンジ","アップル"};

        Arrays.sort(i_array);
        Arrays.sort(s_array);
        Arrays.sort(s2_array);

        for(int val:i_array){
            System.out.print(val + " ");
        }System.out.println();
        for(String val:s_array){
            System.out.print(val + " ");
        }System.out.println();
         for(String val:s2_array){
            System.out.print(val + " ");
        }
    }
}
```

```
10 20 30 40 50
apple banana orange
アップル オレンジ バナナ
```

asList()

```java
import java.util.*;


public class Main {
    public static void main(String[] args) throws Exception {
        String[] s_array = {"banana","orange","apple"};
        List<String> list = Arrays.asList(s_array);

        //list.add("beef");

        for(String val: list){
            System.out.print(val + " ");
        }
    }
}
```

```
banana orange apple
```

引数は
オブジェクト型
配列のみ

コメントアウトを
外すとエラー

引数で指定した配列を
固定サイズのリストで返す

```
Exception in thread "main" java.lang.UnsupportedOperationException
        at java.base/java.util.AbstractList.add(AbstractList.java:153)
        at java.base/java.util.AbstractList.add(AbstractList.java:111)
        at Main.main(Main.java:11)
```

Add
できない！

Date and Time API

- APIの概要
- オブジェクト生成用のメソッド
- DateTimeFormatter
- 日付と時間の加算・減算

APIの概要

- 日付・時間を扱うライブラリ
- 各クラスが不変オブジェクト
- 日時演算機能が充実

Java.time	
LocalDate	日付を表すオブジェクト(年-月-日)
LocalTime	時間を表すオブジェクト(時-分-秒)
LocalDateTime	日付と時間を表すオブジェクト(年-月-日-時-分-秒)
Period	日付ベースの期間を表す
Java.time.format	
DataTImeFormatter	日付/時間の出力・フォーマットを行う

オブジェクト生成用のメソッド

- 各クラスが不変オブジェクトのため、メソッドを介して生成する

	LocalTime (int h, int mi, int s)	LocalDate (int y, int m,int d)	LocalDateTime (int y, int m, int d, int h, int mi, int s)
now	現在の時刻を返す	現在の日付を返す	現在の日付/時刻を返す
of	時、分、秒からLocalTimeオブジェクトを返す	年、月、日からLocalDateオブジェクトを返す	年、月、日、時、分、秒からLocalDateTimeオブジェクトを返す
Parse	17:04:03などのテキストからLocalTimeオブジェクトを取得する	2018-04-03などのテキストからLocalDateオブジェクトを取得する	2018-04-03T17:04:03などのテキストからLocalTimeオブジェクトを取得する

引数が不適切(2022,8,32)だ
と
実行時例外

DateTimeFormatter

- 日付・時間のオブジェクトをフォーマットするパターンを提供
- 以下の3つに加え、独自フォーマットを作成するofPattern()メソッドが提供されている。

ISO_DATE	2011-12-03 2011-12-03+01:00
ISO_TIME	10:15 10:15:30 10:15:30+01:00
ISO_DATE_TIME	2011-12-03T10:15:30 2011-12-03T10:15:30+01:00

実行例

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;


public class Main {
    public static void main(String[] args) throws Exception {
      LocalDateTime dateTime1 = LocalDateTime.now();
      DateTimeFormatter fmt1 = DateTimeFormatter.ISO_DATE;
      DateTimeFormatter fmt2 = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

      System.out.println(fmt1.format(dateTime1));
      System.out.println(fmt2.format(dateTime1));
    }
}
```

```
2018-04-03
2018/04/03 09:57:36
```

コレクションとラムダ式

- 関数型インタフェース
- ラムダ式
- コレクションAPIの利用

関数型インタフェース

- 単一のメソッドを持つインタフェース群
- 後述のラムダ式で利用するのに都合がよい

インタフェース名	抽象メソッド	
Function<T, R>	R apply(T t)	T型の引数を受け取りR型の結果を返す
Consumer<T>	void accept(T t)	T型の引数を受け取り何も返さない
Predicate<T>	Boolean Test(T t)	T型の引数を受け取りTRUEかFALSEを返す
Supplier<T>	T get()	引数無しでT型の値を返す
UnaryOperator<T>	R apply(T t)	引数と同じ型Tの結果を返す

ラムダ式

- 使い捨てのクラスを実装する際に使われる

（実装するメソッドの引数）　ー＞　（処理）

```
1  import java.io.BufferedReader;
2  import java.io.InputStreamReader;
3  import java.util.function.Function;
4
5  public class Main {
6      public static void main(String[] args) throws Exception {
7          Function<String, String> obj = (String str) -> {
8              return "Hello " + str;
9          };
10
11         String str = obj.apply("tanaka");
12         System.out.println(str);
13     }
14 }
```

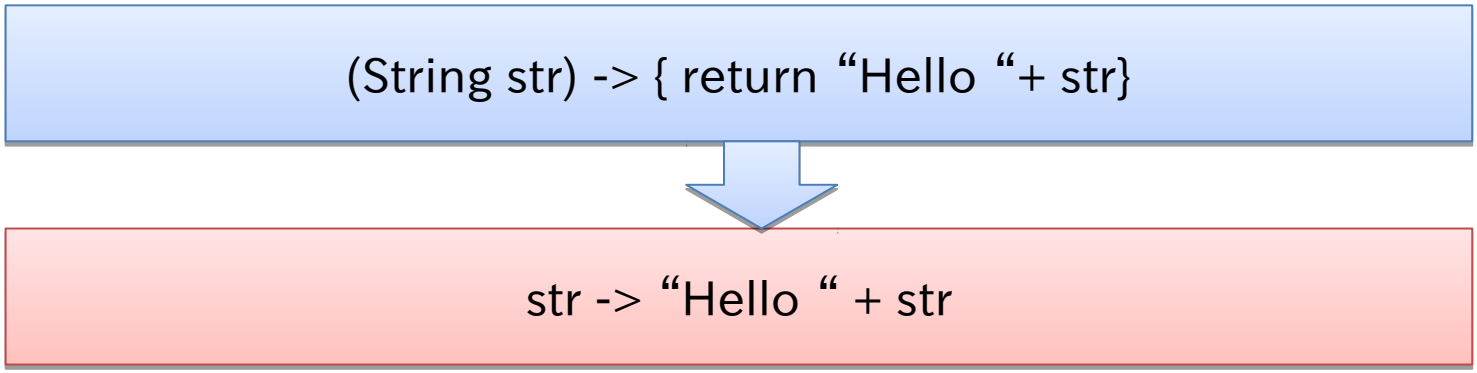
```
Hello tanaka
```

String型の引数を受け取り
String型の結果を返す

ラムダ式の省略

- (String str) -> { return “Hello “+ str}
- 左辺（String str)
  - 型の省略: (str) インタフェース宣言時に型が決まっている
  - ()の省略:str　引数が1つなら可能。　2つ以上OR0個の場合NG

- 右辺（{ return “Hello “+ str}）
  - {}とreturnの省略　処理が1行なら可能



コレクションAPIの利用

- デフォルトメソッドとして以下のようなメソッドがある

Default boolean removeif(Predicate<? Super E> filter)

指定された処理を満たすコレクションの要素を全て削除

Default void replaceAll(UnaryOperator<T> Operator)

指定された処理を行い、要素を置き換える

Predicate<T>

T型の引数を受け取りTRUEかFALSEを返す

UnaryOperator<T>

引数と同じ型Tの結果を返す

実行例

```
1  import java.io.BufferedReader;
2  import java.io.InputStreamReader;
3  import java.util.*;
4  import java.util.function.Function;
5
6  public class Main {
7      public static void main(String[] args) throws Exceptio
8        List<String> words = Arrays.asList("Tanaka", "Sato");
9
10       words.replaceAll( (String str) -> {return str.toUpperCase(); });
11     System.out.println(words);
12     }
13 }
```

①大文字化

②置き換え

```
[TANAKA, SATO]
```

実行例

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;
import java.util.function.*;

public class Main {
    public static void main(String[] args) throws Exception {
      List<Integer> data = new ArrayList<Integer>(Arrays.asList(1,2,3,4,5));

      data.removeIf((Integer i) -> {return i % 2 != 0;} );
      System.out.println(data);
    }
}
```

①判定

②true判定の
要素を削除

1, 3, 5がtrue

```
[2, 4]
```