Introduction
The purpose of this lecture

Object Oriented Programming
2022 First Semester
Shin-chi Tadaki (Saga University)

Object-Oriented / オブジェクト指向

- Construct a model as an object and its operations.
- OOP contains the following concepts
  - Class inheritance /継承
  - Polymorphism / 多形
  - Abstract classes / 抽象クラス
- Java supports OOP

Processes in programming

- Lectures usually give you grammatical structure
- Practical programming requires skills of
  - overall planning / 全体設計
  - arranging the targets / 問題整理
  - class design / クラスデザイン
  - testing / テスト
  - improvement / 改善

The purpose of this lecture

- Skills in OOP through practical samples
- Coding styles of Java
- Smart programming schemes
- Effective skills for writing good codes
- Improving programming skills

To be a good programmer

- Dividing targets into modules
- Arranging overall structure
- Separating data / model, flow, and UI
- Using appropriate libraries
- Learning through good examples
- Learning with good text books

Today's tasks

- Preparing your work platform
- Getting sample codes
- Reviewing fundamentals of Java through `MergeSort` example
    - Recursive `MergeSort` algorithm
    - Investigating the actual processes
    - Understanding tips for implementation
- Reviewing fundamentals of classes and instances

Preparation

- Updating your JDK and NetBeans if necessary
    - `https://aws.amazon.com/jp/corretto/`
    - `https://netbeans.apache.org/download/index.html`
- JDK API manuals
    - `https://www.oracle.com/jp/java/technologies/documentation.html`
- Introducing Git clients if necessary
    - `https://git-scm.com/`
    - All examples of this lecture are provided through `GitHub`.

Getting sample codes

- NetBeans
  `team` $\rightarrow$ `Git`
  - Specify repository : no need to input user name and passwords
  - Specify the destination folder
- Command line
  - Move to the destination folder
  - `git clone` repository

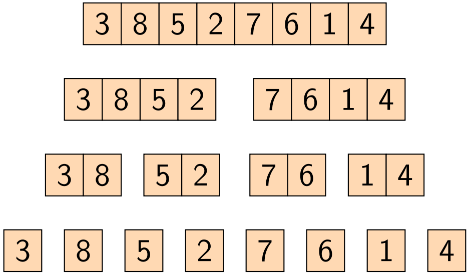
Today's sample codes

- `https://github.com/oop-mc-saga/Sort`
- `example0` package

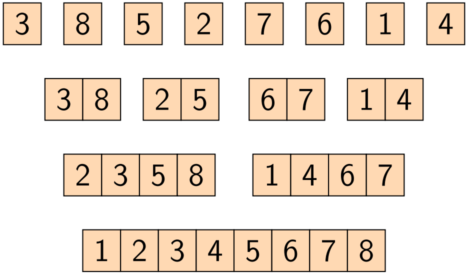
Algorithm 1 Merge Sort (recursive)

```
1: procedure SORT($l, r$)
2:     if $r > l$ then
3:         $m = (l + r)/2$                  ▷ Divide a target into two.
4:         sort($l, m$)
5:         sort($m, r$)
6:         merge($l, m, r$)                 ▷ Merge two sorted lists.
7:     end if
8: end procedure
```

Divide elements: 分割

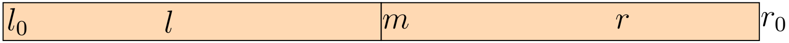


Merge: 結合



Merging two sorted lists

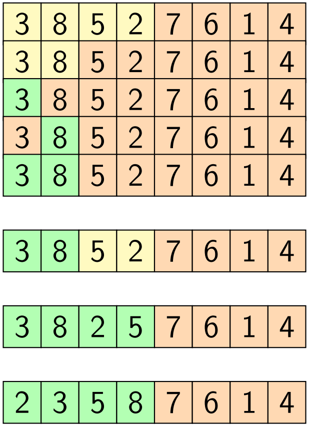
- Tips for implementation
  - Update data by specifying range in the list
  - Need work space (dummy list)



Algorithm 2 merge two sorted lists

```
procedure MERGELIST($l_0, m, r_0$)
    $l = l_0, r = m$
    Prepare dummy list $d_{\text{dummy}}$
    while $l < m \wedge r < r_0$ do
        if $l \geq m$ then                                ▷ Left part is completed
            Append right remainings to dummy
        end if
        if $r \geq r_0$ then                              ▷ Right part is completed
            Append left remainings to dummy
        end if
        if $d_l < d_r$ then
            Append $d_l$ to $d_{\text{dummy}}$
            $l++$
        else
            Append $d_r$ to $d_{\text{dummy}}$
            $r++$
        end if
    end while
    Overwrite $d_{\text{dummy}}$ on original list
end procedure
```

Actual processes : important!



Exercise

Read source codes.

- `sortSub()` method
- `mergeList()` method

Classes and instances

- *Class* is a template of objects
    - Fields: properties
        - values, class instances
    - Methods: for manipulating fields
        - Setters and Getters
- *Instance* is a class realization
    - Keeping values in fields

Modifiers

- public: available from any places
- protected: available only from inherited classes
- private: available only in the class
- final: constant, can not modify

Static modifier

- Methods are usually bound with an instance
  - You can not use methods without creating instances
- Some methods such as mathematical functions should not be bound with an instance.
- `static` methods and fields are bound with a class
  - Available without creating instances

Examples of static method

- `main()`: JVM invokes this before constructing instances for starting application
- Mathematical functions in `Math` class
    - Any instances of `Math` class can not be created
    - Constructor are not allowed to use.

References

- Patrick Niemeyer, Jonathan Knudsen, Learning Java 5th ed. (O' Reilly, 2020).
- D. Poo, D. Kiong and S. Ashok, Object-Oriented Programming and java (Springer, 2008).
- Richard Warburton, Java 8 Lambdas, (O' Reilly, 2014).