

Introduction

The Purpose of This Lecture

Object Oriented Programming
2024 First Semester
Shin-chi Tadaki (Saga University)

1 Object Oriented Programming

2 Today's tasks

3 MergeSort

4 Classes and instances

Object-Oriented / オブジェクト指向

- Schemes for constructing a computational model as a set of objects with their operations.
- OOP (Object Oriented Programming) schemes consist of the following features
 - Class inheritance / 継承
 - Polymorphism / 多形
 - Abstract classes / 抽象クラス
- Java supports OOP schemes

Procedures in programming

- Lectures on programming usually teach you grammatical structure
- Practical programming requires skills of
 - planning overall structure / 全体設計
 - arranging the targets / 問題整理
 - designing classes and modules / クラスとモジュールのデザイン
 - testing functions/ テスト
- Continuous skill improvement is necessary

This lecture covers the followings

- Skills in OOP through practical examples
 - Inheritance
 - Threads
 - Regular expressions
 - File IO
 - Events
 - GUI
- Coding styles of Java
- Effective skills for writing good codes

To be a proficient programmer

- Decomposing goals into smaller objectives
- Establishing a cohesive structure
 - Design and organize your codebase to ensure clarity, modularity, and maintainability.
- Segregating components
 - Divide your code into distinct sections, such as data and models, controls, and user interfaces.
- Employing suitable libraries
- Learning from good examples

Today's tasks

- Preparing your work platform
 - Java 21 is required
- Getting sample codes
- Reviewing fundamentals of Java through a 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.
<https://github.com/oop-mc-saga>

Getting sample codes

- NetBeans

team → 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

Merge Sort Algorithm

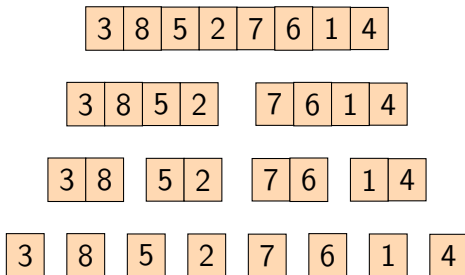
Algorithm 1 Merge Sort (recursive)

```
1: procedure SORT( $l, r$ )
2:   if  $r > l + 1$  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
```

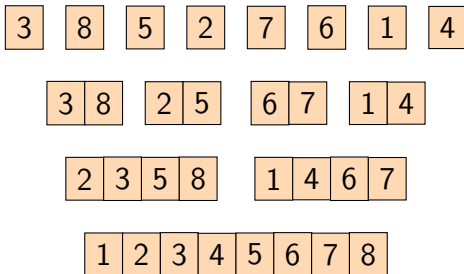
Idea of merge sort

- Merging two sorted list is an easy task: *linear order*
- Dividing the target into short lists of length 1.
- Merging short sorted lists repeatedly.: *logarithmic order*

Divide elements: 分割

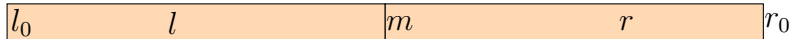


Merge: 結合



Merging two sorted lists

- Tips for implementation
 - Updating data of specifying range in the list
 - This requires 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_{dummy}

while $l < m \wedge r < r_0$ **do**

if $d_l < d_r$ **then**

 Append d_l to d_{dummy}

$l++$

else

 Append d_r to d_{dummy}

$r++$

end if

end while

if $l \geq m$ **then**

 Append right remainings to dummy

end if

if $r \geq r_0$ **then**

 Append left remainings to dummy

end if

Overwrite d_{dummy} on original list

end procedure

▷ Left part is completed

▷ Right part is completed

Actual processes : important!

3	8	5	2	7	6	1	4
3	8	5	2	7	6	1	4
3	8	5	2	7	6	1	4
3	8	5	2	7	6	1	4
3	8	5	2	7	6	1	4

3	8	5	2	7	6	1	4
---	---	---	---	---	---	---	---

3	8	2	5	7	6	1	4
---	---	---	---	---	---	---	---

2	3	5	8	7	6	1	4
---	---	---	---	---	---	---	---

Actual prodesesses

- Sorting the right part $[7, 6, 1, 4]$ is waiting until finishing of sorting the left part $[3, 8, 5, 2]$.
- Sorting the right part $[5, 2]$ is waiting until finishing of sorting the left part $[3, 8]$.
- Sorting the right part $[8]$ is waiting until finishing of sorting the left part $[3]$.
- Then $[3]$ and $[8]$ are merged to $[3, 8]$
- And so on.

Exercise

Read source codes.

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

Classes and instances

- A *Class* is a template of objects
 - Fields: stores properties of the object
 - values, class instances
 - Constructor: initializes class instances
 - The name of the constructor method is the class name in Java.
 - Methods: manipulates the fields
 - Special methods: setters and getters
- An *Instance* is a class realization
 - Class instances are created through class constructors
 - Instances store own values in fields

Modifiers

- Modifiers for access controls
 - `public`: available from any places
 - `protected`: available only from inherited classes
 - `private`: available only in the class
 - Without modifier: The object is public in the package where the class is included.
- `final`: constant, not modifiable
 - The value must be assigned at its definition or in the constructor of the class.

Static modifier

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

Examples of static methods and fields

- `main()`: JVM invokes this method for starting application without creating an instance
- Mathematical functions and constants in `Math` class
 - Any instances of `Math` class can not be created
 - Constructor are not allowed to use.
 - Examples: `Math.sin()`, `Math.PI`

References

- M. Loy, P. Niemeyer, D. Leuck, Learning Java 6th ed. (O'Reilly, 2023).
- D. Poo, D. Kiong and S. Ashok, Object-Oriented Programming and java (Springer, 2008).
- Richard Warburton, Java 8 Lambdas, (O'Reilly, 2014).