LAMBDA

- Functional Interface表示該介面只定義了一個抽象方法。
  - 介面中會加上@FunctionalInterface
  - 例如：Comparator
- Lamdba用來簡化實作Functional Interface的Anonymous class
  - 不會有$1.class產生
- list.forEach(s -> System.out.print(s));
- map.forEach((k, v) -> System.out.print(k + ":" + v));

忽略方法名稱
參數跟回傳重要

- **Runnable = () -> {}**
- **Comparator = (Object o1,Object o2) -> { return o1 > o2 ? -1 : 1 ;}**
- **Comparator = (o1,o2) -> {} 型態可以省略**
- **Comparator = (o1,o2) -> 1; return可以省略 //只有一行時，return 大刮號也可以省略**
- **Set<File> fileSet = new TreeSet<>() 練習排序。**

常遇見的Functional Interface

- **java.util.function.Consumer**
  - **void accept(T t)　拿到一個物件，沒有回傳**
  - **例如：list.forEach(Consumer)，接受一個物件，無回傳值。**
- **java.util.function.Function**
  - **R apply(T t);　拿到一個物件，有回傳**
  - **map.computIfAbscent(Function)**
- **java.util.function.Predicate**
  - **boolean test(T t);　拿到一個物件，回傳boolean值**
  - **set.removeIf(Predicate)**
- **java.util.function.Supplier**
  - **T get();　提供一個物件**

Method Reference

- **Method Reference,只管method定義是否相同，不管名稱。**
  - **list.forEach(System.out::println);**
- **傳入的物件，只使用一個方法，可以改寫成Method Reference**
  - **mapToInt(String::length) // s-> s.length()**
- **傳入的物件，被當做參數傳入另一個物件方法**
  - **forEach(System.out::println) // s-> System.out.println(s)**
- **傳入的物件，被當做參數傳入另一個static方法，跟第一種型式很像**
  - **mapToDouble(Double::parseDouble) // s-> Double::parseDouble(s)**
- **new**
  - **mapToObject(String::new) // s -> new String(s);**

Method Reference

- **x -> x.y()**
  - **Class Of X::f**
- **x -> f(x)**
  - **this::f**
- **x-> new A(x)**
  - **A::new**
- **x -> B.f(x)**
  - **B::f**
- **x,y -> x.z(y)**
  - **X::z**