Checklist review today:

Comparator vs Comparable

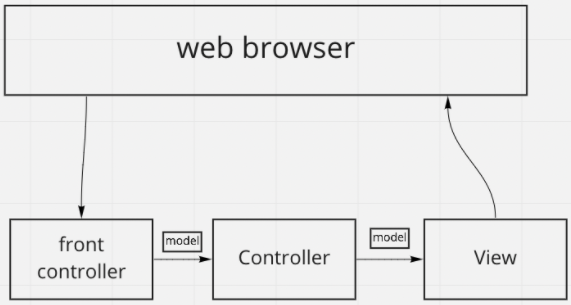
* maintain TreeSet (TreeMap)
* Comparable
  + class implements Comparable<class>
  + Override compareTo(Object o)
  + return this.attribute - o.attribute (ascending order)
* Comparator
  + create class implements Comparator<class>
  + Override compare(DataType o1, DataType o2)
  + return o1.attribute – o2.attribute

Multi-threading

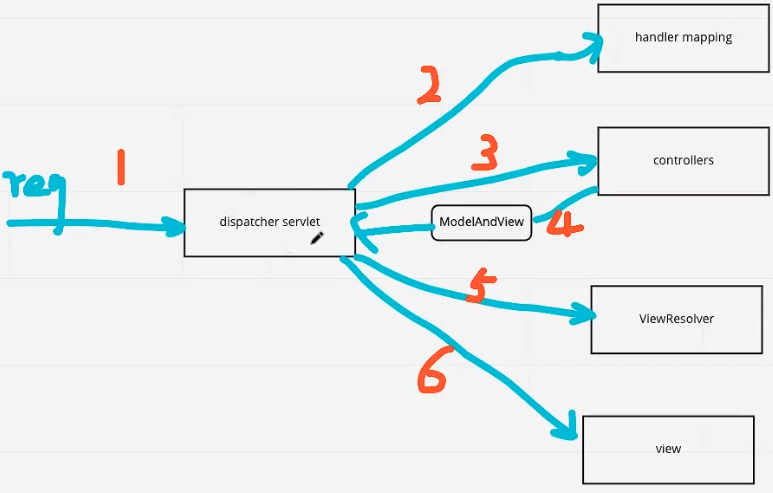
* process - independent memory space, heap, OS resources
* thread - shared memory space, private stack, program counter, register
* states (understands with pic)
  + new - create, not start
  + runnable - execute in JVM
  + blocked - wait for a monitor lock to enter synchronized block or method
  + waiting
    - Object.wait with no timeout
    - Thread.join() with no timeout
    - park()
  + time\_waiting
    - thread sleep
    - Object.wait() with timeout
    - Thread.join() with timeout
    - park
  + terminated – thread has completed
* thread creation
  + extends Thread
  + implements Runnable
    - no return
    - no exception
    - run();
  + implements Callable
    - has return
    - has exception
    - call();
* thread pool
  + corePoolSize
  + maximumPoolSize
  + KeepAliveTime
  + Time unit
  + work queue
  + thread factory
  + handler
    - occurred when # of thread > max + queue(#)
    - abortPolicy
    - callerRunPolicy
    - discardPolicy
    - discardOldestPolicy
  + in-built thread pool (not often use)
    - newFixedThreadPool(nThreads)
      * core = max = nThreads
    - newSingleThreadExecutor()
      * core = max = 1
    - newCachedThreadPool()
      * core = 0, max = MAX\_VALUE
    - SchedualThreadPoolExecutor(corePoolSize)
      * corePoolSize, max = MAX\_VALUE
    - Problem: OutOfMemoryError
      * LinkedBlockQueue
      * max = MAX\_VALUE

New things learned today:

Spring MVC



* front controller: in Spring MVC, dispatcher servlet
* controller: business logic of application. @Controller
* View: web pages, JSP
* model: contains the data of the application
* flow



* + all incoming request is intercepted by the dispatcherServlet that work as the front controller
  + the dispatcherServlet gets an entry of handler mapping form the xml file and forwards the request to the controller
  + the controller return an object of ModelAndView
  + the dispatcherServlet checks the entry of the view resolver in xml file and invokes the specified view component.

Spring Boot

* provide a flexible way to config the java beans, xml configuration and database transactions
* batch processing and manage rest endpoints
* everything is auto configured
  + @SpringBootApplicaiton annotation includes @EnableAutoConfiguration, @ComponentScan, @SpringBootConfiguration
* annotation based spring application
* eases dependency management
* include embedded servlet container -> Tomcat

Rest API Design

* http url design
* basic

一張含有 文字 的圖片

自動產生的描述

* filtering, sorting, searching, sort, search, limited fields, auto loading

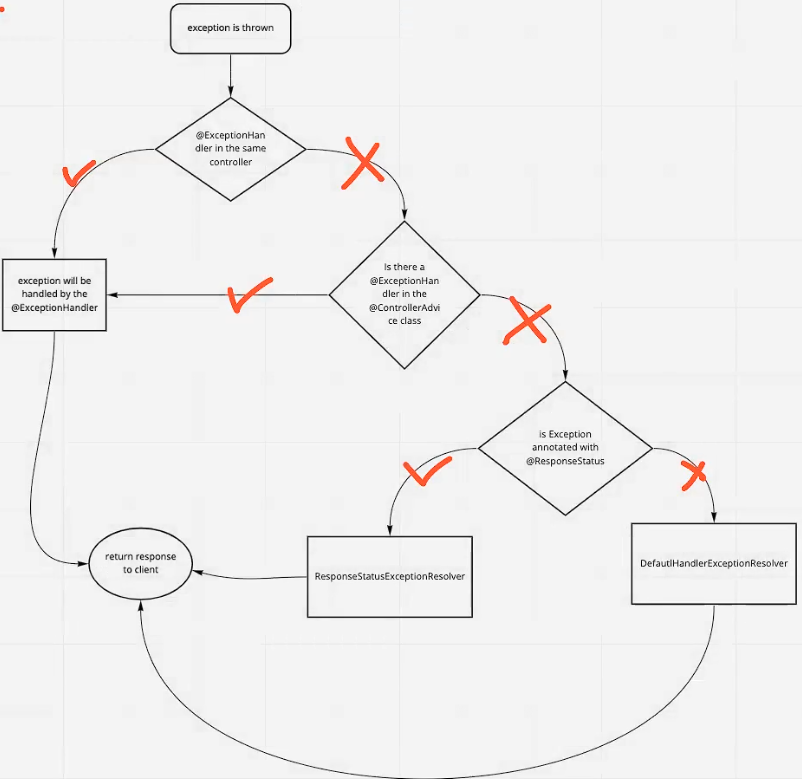
一張含有 文字 的圖片

自動產生的描述

Spring RESTful API

* @RestController, @Controller, @Service, @Repsoitory,
* @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @RequestMapping,
* @PathVariable, @RequestParam
* @RequestBody, @ResponseBody

Exception Handling



* @ExceptionHandler
  + local
* @ControllerAdvice
  + global
* @ReponseStatus

Validation

* @NotNull, @NotEmpty, @NotBlank, @Min, @Max, @Pattern, @Email …
* @Valid, @Validated

Learned Plan tomorrow:

* + Review Java 8 features
    - lambda
    - functional interface
    - Stram API
    - Optional