Newer Post Slider(Newer 20) Home (Archives) Random Post Slider(Random 20) Slider(Older 20)

Monday, July 27, 2015

The Fake Geek's blog: Design a Call Center

The Fake Geek's blog: Design a Call Center
Imagine you have a call center with three levels of employees: fresher, technical lead (TL), product manager (PM).There can be multiple employees, but only
PM.An incoming telephone call must be allocated to a fresher who is free.

If a fresher can't handle the call, he or she must escalate the call to technical lead.If the TL is not free or not able to handle it, then the call should be escalate
PM.Design the classes and data structures for this problem.Implement a method getCallHandler().

https://tianrunhe.wordpress.com/2012/03/18/design-the-classes-and-data-structures-for-a-call-center/
The way of handling abilities to handle the call is implemented by matching employee's level and call's level.
Escalation is implemented by increment of the level of the call.

```
public class CallHandler {
    static final int LEVELS = 3; // we have 3 levels of employees
    static final int NUM_FRESHERS = 5; // we have 5 freshers
    ArrayList<Employee>[] employeeLevels = new ArrayList[LEVELS];
    // queues for each call's rank
    Queue<Call>[] callQueues = new LinkedList[LEVELS];

    public CallHandler() {
        // constructor
    }

    Employee getCallHandler(Call call) {
        for (int level = call.rank; level < LEVELS - 1; level++) {
            ArrayList<Employee> employeeLevel = employeeLevels[level];
            for (Employee emp : employeeLevel) {
                if (emp.free) {
                    return emp;
                }
            }
        }
        return null;
    }

    // routes the call to an available employee, or adds to a queue
    void dispatchCall(Call call) {
        // try to route the call to an employee with minimal rank
        Employee emp = getCallHandler(call);
        if (emp != null) {
            emp.ReceiveCall(call);
        } else {
            // place the call into queue according to its rank
            callQueues[call.rank].add(call);
        }
    }

    void getNextCall(Employee e) {
        // look for call for e's rank
```

Popular Posts

Archives

Design a chess game using OO principles | Runhe Tian Coding Practice

thought-works: Object Oriented design for Elevator in a multi-storied apartment

How to design a tiny URL or URL shortener?

Implement a jigsaw puzzle ~ KodeKnight

Snake Game Design

thought-works: Object oriented design for a Restaurant

Design Key Value Store

Google Map Architecture

Orphan vs Zombie vs Daemon processes | gmarik.info

Labels

- Review (572)
- System Design (334)
- System Design - Review (198)
- Java (189)
- Coding (75)
- Interview-System Design (65)
- Interview (63)
- Book Notes (59)
- Coding - Review (59)
- to-do (45)
- Linux (43)
- Knowledge (39)
- Interview-Java (35)
- Knowledge - Review (32)
- Database (31)
- Design Patterns (31)
- Big Data (29)
- Product Architecture (28)
- MultiThread (27)
- Soft Skills (27)
- Concurrency (26)
- Cracking Code Interview (26)
- Miscs (25)
- Distributed (24)
- OOD Design (24)
- Google (23)
- Career (22)
- Interview - Review (21)

- Java - Code (21)
- Operating System (21)
- Interview Q&A (20)
- System Design - Practice (20)
- Tips (19)
- Algorithm (17)
- Company - Facebook (17)
- Security (17)
- How to Ace Interview (16)
- Brain Teaser (14)
- Linux - Shell (14)
- Redis (14)
- Testing (14)
- Tools (14)
- Code Quality (13)
- Search (13)
- Spark (13)
- Spring (13)
- Company - LinkedIn (12)
- How to (12)
- Interview-Database (12)
- Interview-Operating System (12)
- Solr (12)
- Architecture Principles (11)
- Resource (10)


```
    }
}

class Call {
    int rank = 0; // minimal rank of employee who can handle this call

    public void reply(String message) {
        // reply
    }

    public void disconnect() {
        // disconnect
    }
}

class Employee {
    CallHandler callHandler;
    int rank; // 0- fresher, 1 - technical lead, 2 - product manager
    boolean free;

    Employee(int rank) {
        this.rank = rank;
    }

    void ReceiveCall(Call call) {
        // receive call
    }

    void CallHandled(Call call) {
        // call is complete
    }

    void CannotHandle(Call call) { // escalate call
        call.rank = rank + 1;
        callHandler.dispatchCall(call);
        free = true;
        callHandler.getNextCall(this); // look for waiting call
    }
}

class Fresher extends Employee {
    public Fresher() {
        super(0);
    }
}

class TechLead extends Employee {
    public TechLead() {
        super(1);
    }
```

```
}


class ProductManager extends Employee {
    public ProductManager() {
        super(2);
    }
}
```

Read full article from The Fake Geek's blog: Design a Call Center

Posted by Jeffery at 10:16 PM

Labels: Interview, Interview-System Design, System Design

Newer Post Slider(Newer 20) Home (Archives) Random Post Slider(Random 20) Slider(Older 20)

Labels

Review (572) System Design (334) System Design - Review (198) Java (189) Coding (75) Interview-System Design (65) Interview Notes (59) Coding - Review (59) to-do (45) Linux (43) Knowledge (39) Interview-Java (35) Knowledge - Review (32) Database (31) Design Patterns (31) Big Data Architecture (28) MultiThread (27) Soft Skills (27) Concurrency (26) Cracking Code Interview (26) Miscs (25) Distributed (24) OOD Design (24) Google (23) Career (2 Review (21) Java - Code (21) Operating System (21) Interview Q&A (20) System Design - Practice (20) Tips (19) Algorithm (17) Company - Facebook (17) Security (17 Interview (16) Brain Teaser (14) Linux - Shell (14) Redis (14) Testing (14) Tools (14) Code Quality (13) Search (13) Spark (13) Spring (13) Company - LinkedIn (12) Interview-Database (12) Interview-Operating System (12) Solr (12) Architecture Principles (11) Resource (10) Amazon (9) Cache (9) Git (9) Interview - MultiThread (9) Scalabil Shooting (9) Web Dev (9) Architecture Model (8) Better Programmer (8) Cassandra (8) Company - Uber (8) Java67 (8) Math (8) OO Design principles (8) SOLID (8) Design (7) Interview Corr Java Basics (7) Kafka (7) Mac (7) Machine Learning (7) NoSQL (7) C++ (6) Chrome (6) File System (6) Highscalability (6) How to Better (6) Network (6) Restful (6) CareerCup (5) Code Revie How to Interview (5) JDK Source Code (5) JavaScript (5) Leetcode (5) Must Known (5) Python (5)

Popular Posts

Implement a jigsaw puzzle ~ KodeKnight
Implement a jigsaw puzzle ~ KodeKnight Implement a jigsaw puzzle. Design the data structures and explain an algorithm to solve the puzzle....

Archives

How to design a tiny URL or URL shortener?
Related: http://massivetechinterview.blogspot.com/2015/06/n00tc0d3r.html https://puncsky.com/hacking-the-software-engineer-interview#desi...

Design a chess game using OO principles | Runhe Tian Coding Practice
http://k2code.blogspot.com/2014/03/design-chess-game-using-oo-principles.html http://swcodes.blogspot.in/2012/09/chess-game-design.html ...

thought-works: Object Oriented design for Elevator in a multi-storied apartment
thought-works: Object Oriented design for Elevator in a multi-storied apartment A typical lift has buttons(Elevator buttons) inside the ca...

Snake Game Design
2. 设计贪吃蛇 怎么定义蛇， 怎么移动， 怎么吃， 怎么判断时候活着， 怎么定义游戏版 Design a snake game function played in nokia mobiles Different score strategies

System Design for Big Data [tinyurl]
Related: http://massivetechinterview.blogspot.com/2015/06/how-to-design-tiny-url-or-url-shortener.html https://www.youtube.com/watch?v=fM...

Google Map Architecture
http://all-things-spatial.blogspot.com/2009/06/ingenuity-of-google-map-architecture.html The traditional way to publish maps over the Inte...

Design Key Value Store
https://puncsky.com/hacking-the-software-engineer-interview#design-a-key-value-store-with-external-storage Designing a KV store with exte...

Design Delayed Job Scheduler
Related: http://massivetechinterview.blogspot.com/2015/11/java-delayqueue.html 请教一道面试题 -- delay scheduler 经常在面经里看到要求implement a delay sc...