import java.io.\*;

import java.util.\*;

import java.util.concurrent.\*;

class Product {

private String productID;

private String name;

private int quantity;

private Location location;

public Product(String productID, String name, int quantity, Location location) {

this.productID = productID;

this.name = name;

this.quantity = quantity;

this.location = location;

}

public String getProductID() { return productID; }

public String getName() { return name; }

public int getQuantity() { return quantity; }

public Location getLocation() { return location; }

public void setQuantity(int quantity) { this.quantity = quantity; }

}

class Location {

private int aisle;

private int shelf;

private int bin;

public Location(int aisle, int shelf, int bin) {

this.aisle = aisle;

this.shelf = shelf;

this.bin = bin;

}

}

class Order {

private String orderID;

private List<String> productIDs;

private Priority priority;

public enum Priority {

STANDARD, EXPEDITED

}

public Order(String orderID, List<String> productIDs, Priority priority) {

this.orderID = orderID;

this.productIDs = productIDs;

this.priority = priority;

}

public String getOrderID() { return orderID; }

public List<String> getProductIDs() { return productIDs; }

public Priority getPriority() { return priority; }

}

class OrderComparator implements Comparator<Order> {

@Override

public int compare(Order o1, Order o2) {

return o1.getPriority().compareTo(o2.getPriority());

}

}

class InventoryManager {

private Map<String, Product> products = new ConcurrentHashMap<>();

private PriorityQueue<Order> orderQueue = new PriorityQueue<>(new OrderComparator());

private ExecutorService executor = Executors.newFixedThreadPool(4);

public synchronized void addProduct(Product product) {

products.put(product.getProductID(), product);

}

public void placeOrder(Order order) {

orderQueue.offer(order);

executor.execute(this::processOrder);

}

private void processOrder() {

Order order = orderQueue.poll();

if (order != null) {

for (String productID : order.getProductIDs()) {

Product product = products.get(productID);

if (product != null && product.getQuantity() > 0) {

product.setQuantity(product.getQuantity() - 1);

System.out.println("Order " + order.getOrderID() + " processed for product " + product.getName());

} else {

System.out.println("Out of stock: " + productID);

}

}

}

}

}

public class Main {

public static void main(String[] args) {

InventoryManager inventoryManager = new InventoryManager();

inventoryManager.addProduct(new Product("P001", "Widget", 10, new Location(1, 2, 3)));

inventoryManager.addProduct(new Product("P002", "Gadget", 5, new Location(1, 3, 2)));

List<String> orderProducts = Arrays.asList("P001", "P002");

inventoryManager.placeOrder(new Order("O001", orderProducts, Order.Priority.EXPEDITE));

}

}