OOPS

package com.gs.corejava.oopsModeling;

/\*\*

\* Imagine you have a call center with three levels of employees: respondent,

\* manager and director. An incoming telephone call must be first allocated to a

\* respondent who is free. If the respondent can't handle the call, he or she

\* must escalate the call to a manager. If the manager is not free or not able

\* to handle it, then the call should be escalated to a director. Design the

\* classes and data structures for this problem. Implement a method

\* dispatchCaLL() which assigns a call to the first available employee.

\*

\* @author momalhotra

\*

\*/

public class CallCenterModellingTest {

public static void main(String[] args) {

CallManagement callManagement = new CallManagement();

callManagement.callCustomerCare();

}

}

enum EmployeeType {

RESPONDENT, MANAGER, DIRECTOR,

}

abstract class Employee {

private long empId;

private String name;

private boolean isFree;

private EmployeeType employeeType;

/\*\*

\* @param empId

\* @param name

\* @param employeeType

\*/

public Employee(long empId, String name, EmployeeType employeeType) {

super();

this.empId = empId;

this.name = name;

this.employeeType = employeeType;

}

/\*\*

\* @param empId

\* @param name

\* @param isFree

\* @param employeeType

\*/

public Employee(long empId, String name, boolean isFree,

EmployeeType employeeType) {

super();

this.empId = empId;

this.name = name;

this.isFree = isFree;

this.employeeType = employeeType;

}

/\*\*

\* @return the empId

\*/

public long getEmpId() {

return empId;

}

/\*\*

\* @return the name

\*/

public String getName() {

return name;

}

/\*\*

\* @param name

\* the name to set

\*/

public void setName(String name) {

this.name = name;

}

/\*\*

\* @return the employeeType

\*/

public EmployeeType getEmployeeType() {

return employeeType;

}

/\*\*

\* @param employeeType

\* the employeeType to set

\*/

public void setEmployeeType(EmployeeType employeeType) {

this.employeeType = employeeType;

}

/\*\*

\* @return the isFree

\*/

public boolean isFree() {

return isFree;

}

/\*\*

\* @param isFree

\* the isFree to set

\*/

public void setFree(boolean isFree) {

this.isFree = isFree;

}

}

interface CallDispatcher {

public void dispatchCaLL();

}

class Respondent extends Employee implements CallDispatcher {

public Respondent(long empId, String name, boolean isFree) {

super(empId, name, isFree, EmployeeType.RESPONDENT);

}

@Override

public void dispatchCaLL() {

System.out.println("Call responded by " + getEmployeeType() + " "

+ getEmpId());

}

}

class Manager extends Employee implements CallDispatcher {

public Manager(long empId, String name, boolean isFree) {

super(empId, name, isFree, EmployeeType.MANAGER);

}

@Override

public void dispatchCaLL() {

System.out.println("Call responded by " + getEmployeeType() + " "

+ getEmpId());

}

}

class Director extends Employee implements CallDispatcher {

public Director(long empId, String name, boolean isFree) {

super(empId, name, isFree, EmployeeType.DIRECTOR);

}

@Override

public void dispatchCaLL() {

System.out.println("Call responded by " + getEmployeeType() + " "

+ getEmpId());

}

}

class CallManagement {

private Employee[] repondents;

private Employee[] managers;

private Employee[] directors;

public CallManagement() {

Employee respondent1 = new Respondent(1, "respondent1", false);

Employee respondent2 = new Respondent(2, "respondent2", false);

Employee respondent3 = new Respondent(3, "respondent3", false);

Employee respondent4 = new Respondent(4, "respondent4", false);

repondents = new Employee[4];

repondents[0] = respondent1;

repondents[1] = respondent2;

repondents[2] = respondent3;

repondents[3] = respondent4;

Employee manager1 = new Manager(5, "manager1", false);

Employee manager2 = new Manager(6, "manager2", false);

managers = new Employee[2];

managers[0] = manager1;

managers[1] = manager2;

Employee director1 = new Director(7, "director1", true);

directors = new Employee[1];

directors[0] = director1;

}

public void callCustomerCare() {

boolean callDispatched = false;

for (Employee emp : repondents) {

Respondent respondent = (Respondent) emp;

if (respondent.isFree()) {

respondent.dispatchCaLL();

callDispatched = true;

break;

}

}

if (!callDispatched) {

for (Employee emp : managers) {

Manager manager = (Manager) emp;

if (manager.isFree()) {

manager.dispatchCaLL();

callDispatched = true;

break;

}

}

}

if (!callDispatched) {

for (Employee emp : directors) {

Director director = (Director) emp;

if (director.isFree()) {

director.dispatchCaLL();

callDispatched = true;

break;

}

}

}

}

}