**به نام خدا**

|  |  |  |  |
| --- | --- | --- | --- |
| سری هفتم تمرینات درس برنامه‌نویسی جاوا | ترم دوم سال تحصیلی 97-96 | مدرس: مظفر بگ محمدی | دانشگاه ایلام |

1. **وراثت**: شرکت دنا الکترونیک می­خواهد که سیستم محاسبه­ی خودکار حقوق کارمندان را راه­اندازی کند. این شرکت دارای دو نوع کارمند است. مدیران می­خواهند برای ساعات ورزش نیز حقوق دریافت کنند. به علاوه، حقوق مدیر دارای یک ضریب 1.5 است که بعد از پایان محاسبات در حقوق ضرب می­شود. مهندسین نیز می­خواهند که ساعات معطلی در ترافیک جزء ساعات کار محاسبه شود. هم­چنین، مهندسین 200 دلار به عنوان حق سختی کار نیز دریافت می­کنند. لیست کارمندان شرکت مطابق لیست زیر است:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ورزش | ترافیک | ساعات کار در هفته | دستمزد هر ساعت | نوع | نام خانوادگی | نام |
| 10 | - | 45 | 120.00 | مدیر | هاشمی | علی |
| 5 | - | 55 | 255.50 | مدیر | موسوی | حسن |
| - | 8 | 7 | 220.00 | مهندس | حسینی | زهرا |
| - | 12 | 42 | 124.50 | مهندس | کریمی | فاطمه |
| - | 5 | 38 | 248.00 | مهندس | رضوی | حسین |
| - | 7 | 44 | 320.00 | مهندس | احمدی | محمد |

* 1. ابتدا، کلاس Employee را پیاده کنید. این کلاس شامل فیلدهای firstName, lastName, id , salary , basePay و workHours است. فیلد basePay دستمزد هر ساعت کار را مشخص می­کند. فیلد salary حقوق ماهانه­ی کارمند است و باید محاسبه شود. متد انتزاعی weeklyPay دستمزد هفتگی کارمند را محاسبه می­کند.
  2. حال، کلاس Manager را پیاده کنید که از کلاس Employee ارث­بری می­کند. این­ کلاس دارای فیلد اضافه­ی sportHours است. دستمزد هر ساعت ورزش نصف یک ساعت کار است. دقت کنید که weeklyPay باید بازنویسی شود.
  3. کلاس Engineer نیز از کلاس Employee ارث­بری می­کند. این­ کلاس دارای فیلد اضافه­ی trafficHours است. دستمزد هر ساعت معطلی در ترافیک 70% یک ساعت کار است. دقت کنید که weeklyPay باید بازنویسی شود.
  4. برای تمام کلاسها متدهای equals، toString، اجزاء سازنده­ی کامل، پیش فرض و کپی را پیاده کنید.
  5. در برنامه­ی اصلی یک آرایه از Employee را ایجاد کنید و آنرا مطابق لیست فوق مقداردهی کنید. سپس، در یک حلقه­ی for حقوق تمام کارمندان و میزان اضافه دریافت آنها را چاپ کنید.

1. **شیءگرایی:** کلاس‌های لازم را به‌گونه‌ای پیاده‌سازی کنید که متد main زیر اولاً بدون خطا کامپایل شده و ثانیاً خروجی اجرای آن دقیقاً به شکلی که گفته‌شده است باشد.

public static void main(String[] args) {

    A[] elements = {

        new D(), new A(), new C(), new B()

    };

    for (int i = 0; i < elements.length; i++) {

        System.out.println(elements[i].method1());

        System.out.println(elements[i].method2());

        System.out.println();

    }

}

**خروجی** مورد انتظار:

D1

D1B2

A1

A2

C1

C1B2

A1

A1B2

**در پیاده‌سازی خود، موارد زیر را باید رعایت کنید. رعایت نکردن هر یک از این موارد، موجب کسر نمره می‌شود:**

* به‌جز کلاس A، سایر کلاس‌ها تنها یک متد می‌توانند داشته باشند.
* هیچ کلاس و متد اضافه‌ای ایجاد نکنید.
* هیچ ویژگی (Field) ای در هیچ کلاسی ایجاد نکنید.

1. **فروشگاه لباس:** یک فروشگاه لباس برای نگه‌داری اطلاعات لباس‌های خود از یک برنامه جاوا استفاده می‌کند. این فروشگاه چهار نوع لباس (Clothing) دارد که برای هر یک، یک کلاس در نظر گرفته‌شده است: جوراب Socks، پیراهنShirt ، شلوارPants ، ژاکتJacket

هر لباس نیز دارای سه ویژگی مشترک، نام (name)، فصل مناسب برای پوشیدن لباس: season (بهار، تابستان، پاییز و زمستان) و قیمت پایه (basePrice) است.

یک رفتار مهم و تغییرپذیر در این برنامه، محاسبه قیمت هر لباس با توجه به تخفیف‌های مختلف (زمستانه، تابستانه، شب یلدا و…) است که هم‌اکنون با این منطق پیاده‌سازی شده است:

* **در تخفیف تابستانه:** لباس‌های بهاری: ۴۰٪ تخفیف، لباس‌های تابستانی: ۵۰٪ تخفیف و لباس‌های زمستانه ۳۰٪ تخفیف.
* **در تخفیف زمستانه:** لباس‌های پاییزی: ۴۰٪ تخفیف، لباس‌های زمستانی: ۵۰٪ تخفیف و لباس‌های تابستانی: ۲۵٪ تخفیف. به‌علاوه تمامی ژاکت‌ها از ۱۰٪ تخفیف دیگر نیز برخوردار هستند؛ یعنی یک ژاکت پاییزی ابتدا ۴۰٪ از قیمتش کسر می‌شود چون لباس پاییزی است و سپس ۱۰٪ دیگر از قیمتش کاسته می‌شود.
* **در تخفیف شب یلدا:** تمامی ژاکت‌ها: ۱۰٪ تخفیف، تمامی جوراب‌ها: ۲۰٪ تخفیف و بقیه ی لباس‌ها ۲۵٪ تخفیف.

برنامه‌نویس، در نسخه قدیمی این برنامه در کلاس Clothing یک متد به نام getPrice با ورودی نوع تخفیف (disountType) قرار داده بود که با توجه به نوع تخفیف، نوع لباس و ویژگی‌های لباس، قیمت نهایی لباس را محاسبه می‌کرد:

public long getPrice(DiscountType discountType) {

    if (discountType == DiscountType.YALDA) {

// return with YALDA discount

    } else if (discountType == DiscountType.SUMMER) {

// return with SUMMER discount

    } else if (discountType == DiscountType.WINTER) {

// return with WINTER discount

    }

    return price;

}

حال پس از چند ماه مسئول فروشگاه از او می‌خواهد چند نوع تخفیف جدید برای مناسبت‌های مختلف اضافه کرده و رفتار چند نوع تخفیف موجود را تغییر دهد. برنامه‌نویس متوجه مسئله‌ای می‌شود، با اضافه کردن این تغییرات به متد getPrice و تغییرات احتمالی آینده، این متد رفته‌رفته بزرگ‌تر و ناخواناتر می‌شود. ایده ی جدیدی به سراغ ذهن برنامه‌نویس می‌آید که نتیجه‌ی آن حذف کردن متد getPrice است ولی به دلیل تسلط کافی نداشتن از دانشجویان دانشگاه ایلام کمک گرفته تا ایده‌ی خود را پیاده‌سازی کند. برای مشاهده ی نسخه ی فعلی **بسته** [**com.example.shop**](https://www.dropbox.com/s/gcmnabbh8rmndqx/src.zip?dl=0) **را دانلود کنید.** او می‌خواهد وظیفه محاسبه‌ی هر تخفیف را به کلاس‌هایی مجزایی بسپارد. برای این کار ابتدا یک واسط به نام DiscountStrategy ایجاد کنید:

1. public interface DiscountStrategy {
2. long priceByDiscount(Clothing clothing);
3. }

سپس به ازای هر نوع تخفیف، یک کلاس ایجاد کنید که این واسط را پیاده‌سازی می‌کند. در متد priceByDiscount هر یک از این کلاس‌ها باید رفتار نوع تخفیف مربوطه را پیاده‌سازی کنید:

* برای تخفیف تابستانه: کلاس SummerDiscountStrategy
* برای تخفیف زمستانه: کلاس WinterDiscountStrategy
* برای تخفیف شب یلدا: کلاس YaldaDiscountStrategy

در کلاس Clothing متد زیر را برای دریافت و ثبت **نوع تخفیف فعلی** پیاده‌سازی کنید:

public void setDiscountStrategy(DiscountStrategy discountStrategy) {

// TODO set discountStrategy

}

حال در متد getPrice جدید (که دیگر پارامتر ورودی ندارد) با استفاده از discountStrategy مشخص‌شده، قیمت با تخفیف را محاسبه کرده و برگردانید:

public long getPrice() {

// TODO get price by considering discountStrategy

}

اگر هیچ discountStrategy ای تعیین نشده بود، باید قیمت پایه‌ی لباس برگردانده شود. **توجه:** متد getPrice از کلاس Clothing نباید از نحوه‌ی محاسبه‌ی تخفیف‌های مختلف خبر داشته باشد. به‌عبارت‌دیگر، تمام منطق محاسبه‌ی هر نوع تخفیف باید در کلاس مربوط به خود نوشته شود.

**مثال:**

Jacket linenCoat = new Jacket("Linen Coat", Season.FALL, 100000);

System.out.println("Linen Coat: " + linenCoat.getPrice());

linenCoat.setDiscountStrategy(new YaldaDiscountStrategy());

System.out.println("Linen Coat with Yalda discount: " + linenCoat.getPrice());

linenCoat.setDiscountStrategy(new WinterDiscountStrategy());

System.out.println("Linen Coat with Winter discount: " + linenCoat.getPrice());

Clothing woolenSocks = new Pants("Woolen Socks", Season.WINTER, 10000);

System.out.println("Woolen Socks: " + woolenSocks.getPrice());

woolenSocks.setDiscountStrategy(new SummerDiscountStrategy());

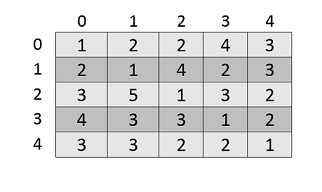
System.out.println("Woolen Socks with Summer discount: " + woolenSocks.getPrice());

woolenSocks.setDiscountStrategy(new WinterDiscountStrategy());

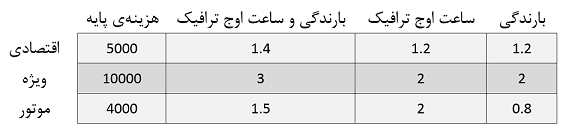
System.out.println("Woolen Socks with Winter discount: " + woolenSocks.getPrice());

**خروجی:**

1. Linen Coat: 100000
2. Linen Coat with Yalda discount: 90000
3. Linen Coat with Winter discount: 54000
4. Woolen Socks: 10000
5. Woolen Socks with Summer discount: 7000
6. Woolen Socks with Winter discount: 5000
7. **تاکسی آنلاین:** برای یک شهرک کوچک، قصد داریم با کمک دانشجویان دانشگاه ایلام بخشی از یک برنامه‌ی سفارش تاکسی آنلاین را پیاده‌سازی کنیم. ابتدا، بسته  [com.example.onlineTaxi](https://www.dropbox.com/s/9be79ew0p0oqq7d/srcOnlineTaxi.zip?dl=0) را دانلود کنید. کارشناسان ترافیک، این شهرک را به ۵ ناحیه‌ی اصلی با شماره‌های ۰ تا ۴ تقسیم‌بندی کرده‌اند و همان‌طور که در شکل زیر دیده می‌شود، برای رفتن از هر ناحیه به ناحیه دیگر، ضریبی تعیین کرده‌اند که در محاسبه‌ی هزینه‌ی نهایی مورداستفاده قرار می‌گیرد.



برای سادگی، این اطلاعات در قالب ماتریس دوبعدی distance، در کلاس DistanceMap در اختیار شما قرارگرفته است. برای سفارش تاکسی از این برنامه، سه روش مختلف (TripMethod) وجود دارد و با توجه روش انتخاب‌شده توسط مسافر، هزینه‌ی سفر وی محاسبه‌شده و نمایش داده می‌شود.



محاسبه‌ی قیمت برای هر یک از این روش‌ها، با توجه به جدول ضرایب بالا و مقادیر موجود در ماتریس فاصله‌ها، به این صورت محاسبه می‌شود: اگر مقدار (هزینه‌ی پایه \* مقدار استخراج‌شده از ماتریس فاصله‌ها) برابر با X باشد، هزینه‌ی نهایی برابر است با

X \* ratio

که مقدار ratio با توجه به ورودی‌های مسئله از جدول بالا انتخاب می‌شود. به‌طور مثال، اگر کاربر قصد سفارش تاکسی اقتصادی برای رفتن از ناحیه ۲ به ۳ در هوای بارانی و ساعت غیر اوج ترافیک را داشته باشد، هزینه‌ی سفارش برابر است با:

X = 5 \* 3 = 15

15 \* 1.2 = 18

**نکات قابل‌توجه:**

* کلاس TripHandler باید singleton باشد و تنها نمونه‌ی آن، از طریق متد getInstance قابل‌دسترس باشد.
* در کلاس TripHandler متد calcPrice را باید پیاده‌سازی کنید که دو ورودی دارد:
* type: نوع سفارش را تعیین می‌کند. مقادیر مجاز برای این ورودی برابر با bike یا economic و یا vip است. با توجه به مقدار این ورودی، باید نوع مناسب از TripMethod انتخاب‌شده و در محاسبه‌ی هزینه استفاده شود.
* :params از نوع TripParam است که شامل ویژگی‌های لازم برای محاسبه‌ی هزینه‌ی سفر (شامل شماره ناحیه مبدأ و مقصد، زمان اوج ترافیک بودن یا نبودن و بارانی بودن یا نبودن هوا) است. به سازنده‌ی این کلاس دقت کنید.
* به ازای هر روش سفارش تاکسی، یک پیاده‌سازی برای واسط TripMethod باید انجام دهید:
* EconomicTripMethod، VipTripMethod و BikeTripMethod
* متد calcPrice در ورودی یک شیء از نوع TripParam می‌گیرد و هزینه‌ی سفر را با توجه به ورودی و روش سفارش تاکسی، محاسبه می‌کند. به‌عبارت‌دیگر منطق محاسبه‌ی هزینه برای هر نوع سفارش در این متد پیاده‌سازی می‌شود و در متد calcPrice از TripHandler، این وظیفه به نمونه‌ی مناسب از یکی از پیاده‌سازی‌های TripMethod سپرده می‌شود.

**مثال:**

TripHandler taxi = TripHandler.getInstance();

//سفر ویژه از ناحیه 1 به 1 در ساعت اوج ترافیک و هوای غیر بارانی

int price = taxi.calcPrice("vip", new TripParam(1, 1, true, false));

int price2 = new VIPTripMethod().calcPrice(new TripParam(1, 1, true, false));

System.out.println(price + " = " + price2);

//سفر با موتور از ناحیه 2 به 4 در ساعت اوج ترافیک و در هوای بارانی

price = taxi.calcPrice("bike", new TripParam(2, 4, true, true));

price2 = new BikeTripMethod().calcPrice(new TripParam(2, 4, true, true));

System.out.println(price + " = " + price2);

**خروجی:**

20000 = 20000

12000 = 12000