День 11

Темы:

Урок 26: Интерфейсы

Задачи:

1. На складе происходит сборка и упаковка интернет-заказов.
Создайте классы:

- “Склад” (англ. `Warehouse`) с полями `countPickedOrders` (количество собранных заказов), `countDeliveredOrders` (количество доставленных заказов), `get` методами для обоих полей и методом `toString()` для получения информации о значениях полей склада.

- “Сборщик” (англ. `Picker`) с полями `salary` (заработная плата) и `isPayed` (был выплачен бонус или нет), `get` методами для обоих полей, методом `toString()` и конструктором.

- “Курьер” (англ. `Courier`) с полями `salary` (заработная плата) и `isPayed` (был выплачен бонус или нет), `get` методами для обоих полей, методом `toString()` и конструктором.

В классах “Сборщик” и “Курьер” могут понадобиться и другие поля на ваше усмотрение (чтобы эти классы соответствовали условиям задачи).

Каждый класс-сотрудник должен реализовывать интерфейс `Worker`, в котором необходимо объявить сигнатуры `doWork()` и `bonus()`.
Каждый раз, когда сотрудник выполняет свою работу (вызов метода `doWork()`), ему выплачивается заработная плата (сокр. ЗП) (80 — сборщику, 100 — курьеру).
Также, при вызове `doWork()` у Сборщика, происходит увеличение значения поля `countPickedOrders` в объекте класса `Warehouse` на 1. А при вызове `doWork()` у Курьера, происходит увеличение значения поля `countDeliveredOrders` в объекте класса `Warehouse` на 1. Подумайте о том, как связать объекты работников с объектом склада (один из возможных вариантов - передавать объект склада в качестве аргумента при создании объектов-работников и хранить его в поле).
Сотрудникам полагается бонус, в зависимости от персональных показателей: когда на складе собрано 10.000 заказов, Сборщику выплачивается бонус в размере 70.000. Когда клиентам доставлено 10.000 заказов, Курьеру выплачивается бонус в размере 50.000.
Если на складе несколько сотрудников одной категории, то оценивается их коллективная работа, т.е. если 10 курьеров доставят каждый по 1000 заказов, то каждый курьер получит бонус.
Бонус сотрудникам должен выдаваться при вызове метода `bonus()`. Причем, если на складе не достигнуты необходимые показатели (10.000 собранных или доставленных

заказов), вызов метода `bonus()` не должен начислять денежную премию, а должен выводить в консоль сообщение "`Бонус пока не доступен`". Бонус может быть выплачен только один раз. При попытке повторной выплаты бонуса (повторный вызов метода `bonus()` на работнике) в консоль должно выводиться сообщение "`Бонус уже был выплачен`".

Реализуйте в классе `Task1` статический метод:
`static void businessProcess(Worker worker)`

Этот метод в качестве аргумента принимает объект класса, реализующего интерфейс `Worker`. В теле этого метода на объекте `worker` должен 10.000 раз вызываться метод `doWork()`, и после этого должен быть один раз вызван метод `bonus()`.

Для демонстрации и тестирования работы программы, в методе `main()` создайте склад и по 1 рабочему. Свяжите этих двух рабочих со складом. После этого, вызовите метод `businessProcess(Worker worker)`, передавая в качестве аргумента сотрудника. Вызовите метод сначала для сборщика, а потом для курьера. Выведите в консоль количество собранных и доставленных заказов на складе и ЗП каждого из сотрудников.

Создайте второй склад, на который также "примите" по 1 новому сотруднику. Вызовите один раз метод `doWork()` у сотрудников второго склада. Проконтролируйте, что у склада 1 и его сотрудников при этом значения не меняются.

2. В соответствии с таблицей ниже реализовать классы персонажей и необходимые интерфейсы.

Герой `Hero`	Здоровье `Health`	Тип атаки и величина урона	Защита Ф/М `physDef / magicDef`	Лечение
Воин (англ. `Warrior`)	100	Ф 30	80% / 0%	нет
Паладин (англ. `Paladin`)		Ф 15	50% / 20%	`healHimself` 25 `healTeammate` 10
Маг (англ. `Magician`)		Ф 5 М 20	0% / 80%	нет
Шаман (англ. `Shaman`)		Ф 10 М 15	20% / 20%	`healHimself` 50 `healTeammate` 30

Пояснения к таблице:

- В колонке “Герой” - названия классов героев, которые должны быть реализованы. Все классы наследуются от абстрактного класса `Hero`.

- Тип атаки, которую может наносить герой: “Ф” - физическая атака (`physAtt`), “М” - магическая атака (`magicAtt`).

- Защита героя от физических и магических атак: Ф 80% - поглощение 80% физического урона, например при Ф атаке на 100 единиц, героем будет получен урон лишь 20 единиц. Для М атаки - аналогично.

- Лечение: `healHimself()` - персонаж лечит себя на указанное количество единиц, `healTeammate(Hero hero)` - персонаж лечит союзника на указанное количество единиц, “нет” - персонаж не имеет способности лечить кого-либо.

- Максимальное здоровье любого Героя - 100, минимальное - 0.

Обратите внимание!
Значения защиты в процентах, а другие параметры в единицах, т.е. атаковав воина типом Ф 10, он получит урон не 10, а 2 ед. урона (`10 - 10*0.8 = 2`)

Необходимые интерфейсы и их сигнатуры:

- Лечение - `interface Healer`.
  Сигнатуры: `healHimself(), healTeammate(Hero hero)`

- Физическая атака - `interface PhysAttack`.
  Сигнатура: `physicalAttack(Hero hero)`

- Магическая атака - `interface MagicAttack`.
  Сигнатура: `magicalAttack(Hero hero)`

Каждый класс-герой должен иметь:

- Поля
  - `health` (здоровье)
  - `physDef` (процент поглощения физического урона)
  - `magicDef` (процент поглощения магического урона)
  - `physAtt` (величина физической атаки), по необходимости
  - `magicAtt` (величина магической атаки), по необходимости
- Реализацию необходимых интерфейсов
- Переопределенный `toString()`

Каждый герой должен обладать только теми способностями, которые ему доступны. Например, Воин не может лечить, значит в классе Воин **не** реализуется интерфейс `Healer`, соответственно запись `warrior.healHimself()` является недопустимой.

Параметры для героя задаются внутри конструктора, при этом сам конструктор не должен принимать аргументов.

После того, как все классы будут реализованы, в методе `main()` класса `Task2` последовательно выполните следующие действия, проверяя показатель здоровья у персонажа, на которого направлено действие:

1. Воин атакует Паладина
2. Паладин атакует Мага
3. Шаман лечит Мага
4. Маг атакует Паладина, тип атаки М
5. Шаман атакует Воина, тип атаки Ф
6. Паладин лечит себя
7. Воин атакует Мага 5 раз

Результат в консоли:

```
Paladin{health=85}
Magician{health=85}
Magician{health=100}
Paladin{health=69}
Warrior{health=98}
Paladin{health=94}

Magician{health=70}
Magician{health=40}
Magician{health=10}
Magician{health=0}
Magician{health=0}
```