

Time limit	1 s
Memory limit	64 MB

Nama File: MediaPlayer.zip

Implementasikan kelas dalam file [berikut](#).

Upload jawaban sebagai berkas **MediaPlayer.zip** yang berisi tepat 5 file: LiveStream.java, Podcast.java, Movie.java, Downloadable.java, dan Playable.java

Gunakan main file [berikut](#) untuk melakukan pengujian hasil implementasi kalian.

Catatan: Pastikan setiap output diakhiri oleh newline ("\n") atau menggunakan `println`.

Java 8 ▾

 [MediaPlayer.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.06 sec, 27.85 MB
2	10	Accepted	0.06 sec, 26.86 MB
3	10	Accepted	0.06 sec, 27.94 MB
4	10	Accepted	0.06 sec, 28.99 MB
5	10	Accepted	0.06 sec, 29.01 MB
6	10	Accepted	0.06 sec, 28.00 MB
7	10	Accepted	0.06 sec, 28.73 MB
8	10	Accepted	0.06 sec, 28.67 MB
9	10	Accepted	0.06 sec, 28.93 MB
10	10	Accepted	0.06 sec, 28.90 MB

Time limit	1 s
Memory limit	64 MB

Nama File: Nimonsgram.zip

Implementasikan kelas dalam file [berikut](#).

Upload jawaban sebagai berkas **Nimonsgram.zip** yang berisi tepat 6 file: Engageable.java, Sharable.java, Post.java, PhotoPost.java, VideoPost.java dan PostUtil.java

Gunakan main file [berikut](#) untuk melakukan pengujian hasil implementasi kalian.
Berikut [output](#) yang diharapkan

Catatan: Pastikan setiap output diakhiri oleh newline ("\\n") atau menggunakan `println`.

Java 8 ▾

 [Nimonsgram.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.14 sec, 28.46 MB
2	10	Accepted	0.47 sec, 33.48 MB
3	10	Accepted	0.29 sec, 28.25 MB
4	10	Accepted	0.31 sec, 28.17 MB
5	10	Accepted	0.43 sec, 32.25 MB
6	10	Accepted	0.30 sec, 26.33 MB
7	10	Accepted	0.28 sec, 27.98 MB
8	10	Accepted	0.12 sec, 28.25 MB
9	10	Accepted	0.11 sec, 28.05 MB
10	10	Accepted	0.21 sec, 26.27 MB

Time limit	1 s
Memory limit	64 MB

Nama File: Gadgetin.zip

Implementasikan kelas dalam file [berikut](#).

Upload jawaban sebagai berkas **Gadgetin.zip** yang berisi file yang sama dengan file zip soal (5 class file dan 4 interface file).

Gunakan kode berikut untuk menguji salah satu kegunaan interface:

```
IGadget[] gadgets = {
new Smartphone("Google", "Pixel 7", 6999999),
new Laptop("ASUS", "ZenBook", 8999999, "Windows 11"),
new GamingConsole("Microsoft", "Xbox Series X", 4999999)
};
```

Catatan: Pastikan setiap output diakhiri oleh newline ("\n") atau menggunakan `println`.

Java 8

 [Gadgetin.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.07 sec, 28.52 MB
2	12	Accepted	0.06 sec, 28.95 MB
3	12	Accepted	0.06 sec, 28.93 MB
4	12	Accepted	0.07 sec, 26.38 MB
5	12	Accepted	0.06 sec, 28.98 MB
6	12	Accepted	0.06 sec, 28.08 MB
7	12	Accepted	0.06 sec, 30.20 MB
8	16	Accepted	0.06 sec, 28.68 MB

Time limit	1 s
Memory limit	64 MB

Kumpulkan dalam **Pembayaran.zip**

Bayangkan anda sedang merencanakan suatu payment gateway. Jika sebelumnya kalian telah mengimplementasikan cashier, kali ini kita akan membuatnya dengan lebih advance dan clean. Interface dapat membuat kode menjadi lebih clean dan scalable. Alih alih langsung override dari Pembayaran, kita akan buat interface (kontrak) bagi kelas yang akan mengimplementasikannya. Selain itu, Interface juga memberikan kemudahan kepada kita untuk membuat implementasi kelas dari multiple interface (kontrak).

Untuk memahaminya buatlah:

- 1. Kelas Pembayaran
 - Memiliki atribut privat balance (int)
 - Memiliki setter dan getter (setBalance()-->int, getBalance()-->int)

- 2. Interface Payable
 - Memiliki method void pay(int amount);

- 3. Interface Refundable
 - Memiliki method void refund(int amount);

- 4. Kelas EWallet
 - Kelas ini mengimplementasikan Payable dan Refundable
 - Kelas ini merupakan turunan dari kelas Pembayaran

- 5. Kelas COD
 - Kelas ini mengimplementasikan Payable
 - Kelas ini merupakan turunan dari kelas Pembayaran

Signature method:


- 1. void pay(int amount)
 - Jika amount valid (lebih besar dari nol dan balance nya cukup), keluarkan output ("[<Tipe Pembayaran>] Paid: {amount}. Remaining balance: {balance}")
 - Jika tidak valid, keluarkan output ("[<Tipe Pembayaran>] Payment failed. Insufficient balance or invalid amount.")
- 2. void refund(int amount)
 - Jika amount valid (lebih besar dari nol), keluarkan output ("[<Tipe Pembayaran>] Refunded: {amount}. New balance: {balance}")
 - Jika tidak valid, ("[<Tipe Pembayaran>] Refund failed. Invalid amount.")

Sekarang, tiap kali anda ingin menambah metode pembayaran, tinggal membuat kelas baru dan memilih ingin bisa payable dan refundable, atau hanya salah satu saja. Dengan design seperti ini, code anda menjadi lebih loosly coupling dan bisa mengimplementasikan lebih dari satu interface (kontrak).

Note:

- <Tipe Pembayaran> -> Sesuaikan dengan tipe pembayaran ("COD" atau "EWallet")
- Jangan lupa endline setiap statement
- Kumpulkan file **Pembayaran.java Payable.java, Refundable.java, EWallet.java, COD.java** dalam **Pembayaran.zip**

Java 8

 [Pembayaran.zip](#)