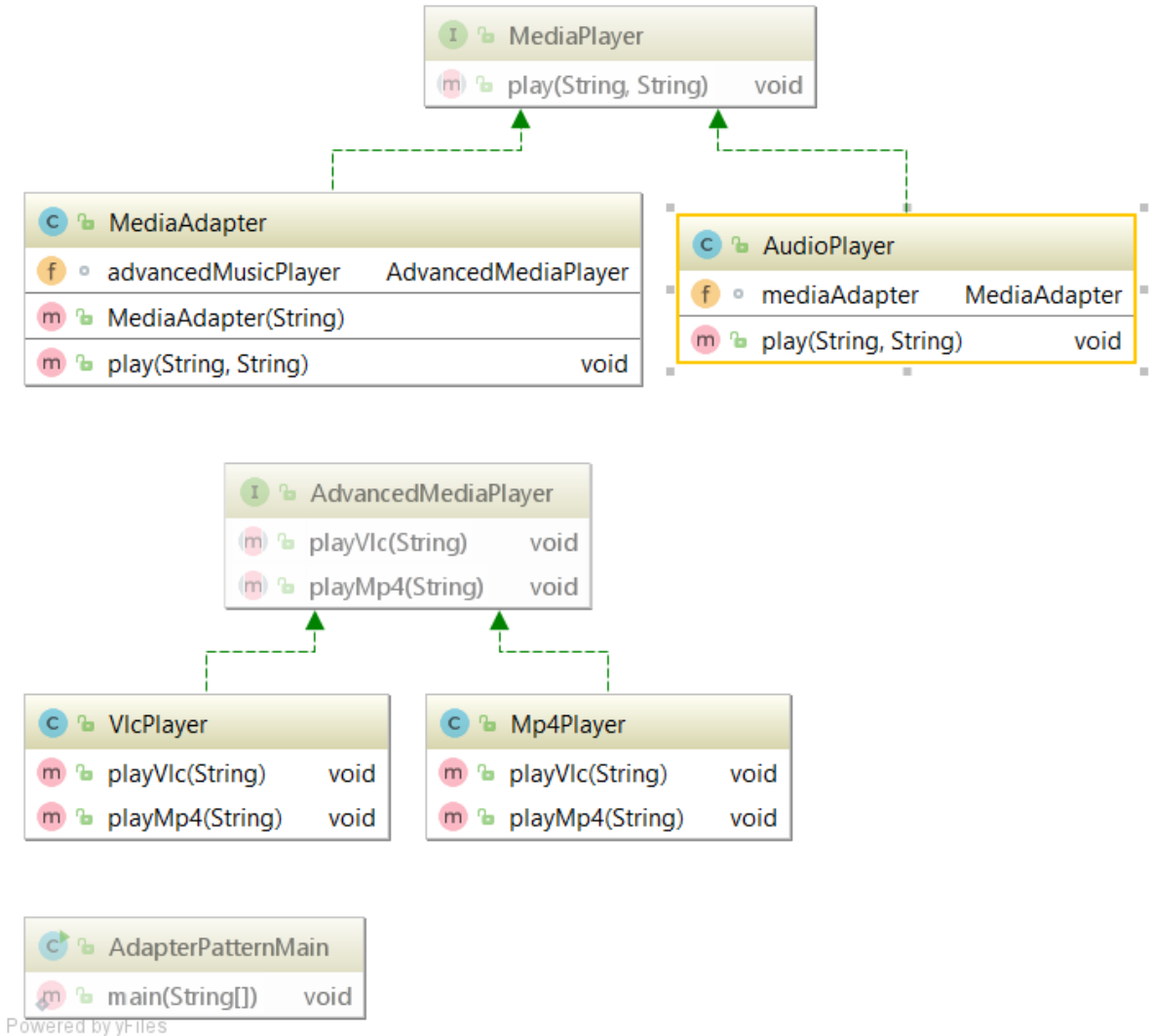
INTEC BRUSSEL MEER KANSEN OP WERK	**Datum**: 26/10/2023 **Opleiding**: Java Developer **Lesmodule**: Java Basis **Test**: **Tijd**:
Resultaat: **/ 20**	**Naam**:

1 AUDIOPLAYER /10

Bekijk onderstaand UML diagram aandachtig.



Een beetje duiding

We hebben een MediaPlayer-interface en een concrete klasse AudioPlayer die deze interface implementeert. AudioPlayer kan standaard audiobestanden in mp3-formaat afspelen.

We hebben ook nog een andere interface AdvancedMediaPlayer en concrete klassen die deze interface implementeren zoals VlcPlayer en Mp4Player. Deze klassen kunnen VLC en mp4-formaat bestanden afspelen.

We willen AudioPlayer ook in andere formaten laten spelen. Om dit te bereiken moeten wij een klasse MediaAdapter maken die de MediaPlayer interface implementeert en gebruik maakt van objecten van AdvancedMediaPlayer om het vereiste formaat te kunnen afpelen.

AudioPlayer gebruikt de klasse MediaAdapter die het gewenste audiotype doorgeeft zonder de klasse te kennen kan die het gewenste formaat spelen.

AdapterPatternDemo, onze demoklasse, zal de klasse AudioPlayer gebruiken om verschillende formaten te spelen.

Stap 1

Creëer de interfaces Media Player en Advanced Media Player.

- ❏ MediaPlayer heeft 1 methode play(String audioType, String fileName);
- ❏ AdvancedMediaPlayer heeft 2 methodes playVlc(String fileName) en playMp4 (String fileName) beide zonder implementatie.

Stap 2

Creëer concrete klassen die implementeren van de AdvancedMediaPlayer interface Geef deze de naam VlcPlayer en Mp4Player.

- ❏ VlcPlayer implementeert de methoden van AdvancedMediaPlayer Maar geeft enkel de methode playVlc een implementatie namelijk een sysout:

System.out.println(“Playing vlc file. Name: ” + fileName);

- ❏ Mp4Player op zijn beurt laat de implementatie van de methode playVlc leeg en implementeert enkel de methode playMp4 met de volgdende sysout: System.out.println(“Playing mp4 file. Name: “ + fileName);

Stap 3

Creëer de klasse MediaAdapter die de MediaPlayer interface implementeert.

- ❏ Declareer een eigenschap van het type AdvancedMediaPlayer (gebruik een zinvolle naam).

- ❏ De constructor heeft een String audioType als parameter en deze zal gebruikt worden om te zien of je de eigenschap als een instantie van VlcPlayer of Mp4player moet aanmaken. (hint bekijk mss de String Java API)

- ❏ Implementeer de methode play hierin maak je een keuze op basis van je argumenten om te zien of je de methode playVlc of playMp4 moet aangeroepen worden. Geef hieraan het argument fileName mee.

Stap 4

Creëer de concrete klasse die implementeert van de MediaPlayer interface Geef deze de naam AudioPlayer.

- ❏ Declareer in deze klasse een eigenschap van het type MediaAdapter.

- ❏ Implementeer de methode play hier zodanig dat je op basis van je argumenten (String audioType, String Filename) ofwel een sysout doet in het geval de String audioType gelijk is aan mp3 of gebruik de eigenschap mediaAdapter om de play methode aan te roepen (in het geval van mp4 of vlc).

Stap 5

Maak de klasse AdapterPatternDemo en neem onderstaand vb over:

```
public class AdapterPatternDemo {
	public static void main(String[] args) {
		AudioPlayer audioPlayer = new AudioPlayer();
		audioPlayer.play("mp3", "beyond the horizon.mp3");
		audioPlayer.play("mp4", "alone.mp4");
		audioPlayer.play("vlc", "far far away.vlc");
		audioPlayer.play("avi", "mind me.avi");
	}
}
```

Stap 6

Controleer de output:

Playing mp3 file. Name: beyond the horizon.mp3

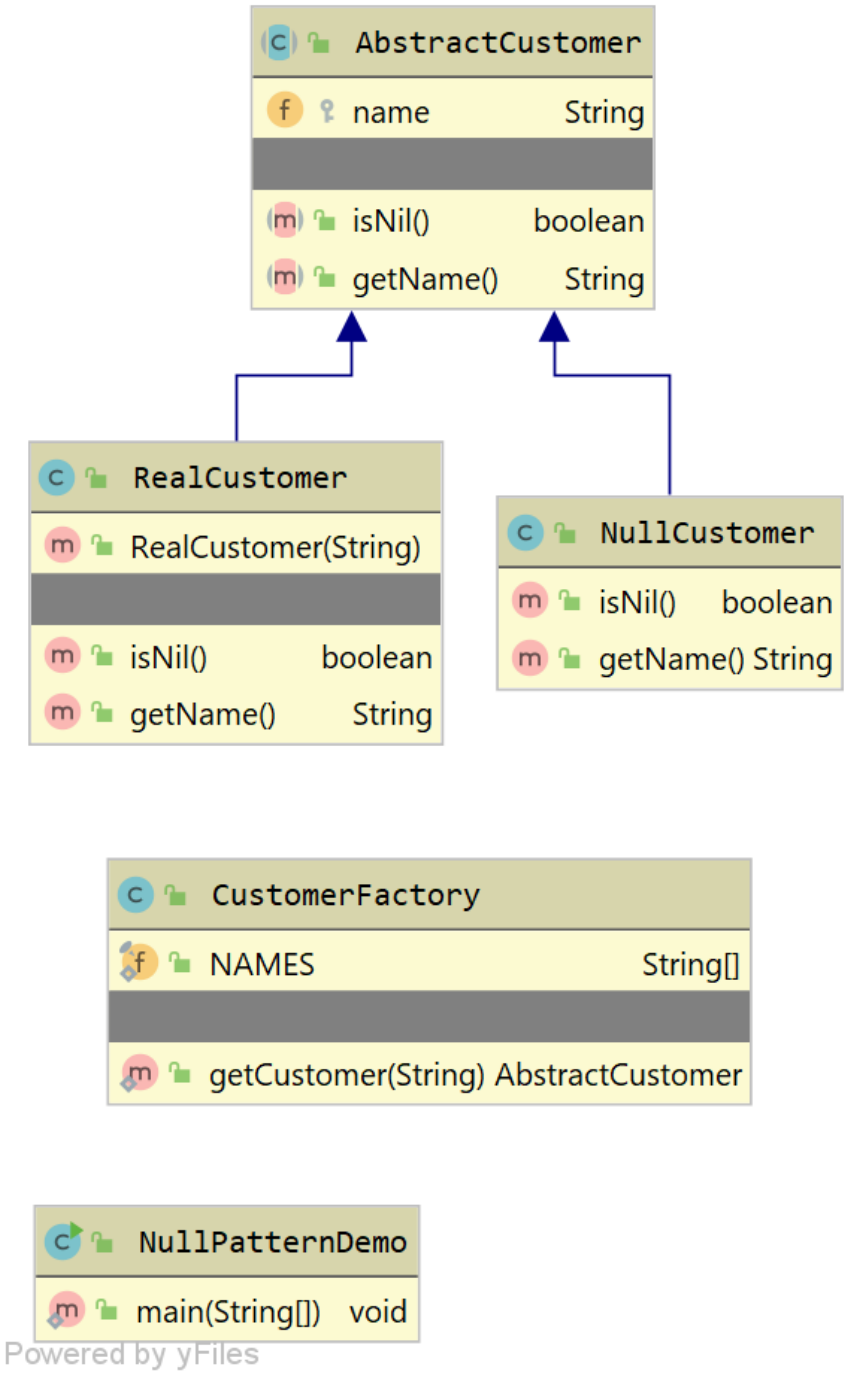
Playing mp4 file. Name: alone.mp4

Playing vlc file. Name: far far away.vlc

Invalid media. avi format not supported

2 CUSTOMER /10

Bekijk onderstaand UML diagram aandachtig.



Stap 1

Maak de abstracte klasse AbstractCustomer

- ❏ Deze heeft als eigenschap een protected String name
- ❏ En de volgende 2 abstracte methoden:
    - ❏ public methode isNil met als returntype boolean
    - ❏ public methode getName met als returntype String

Stap 2

Maak de concrete klasse Realcustomer die extend van de klasse AbstractCustomer

- ❏ Deze heeft een constructor die als parameter een String name heeft
- ❏ Geef een zinvolle implementatie aan getName
- ❏ Return false bij de implementatie van isNil

Stap 3

Maak de concrete klasse NullCustomer die ook extend van de klasse AbstractCustomer

- ❏ Implementeer de getName methode door de String literal “Not Available in Customer Database” terug te geven.
- ❏ De methode isNil implementeer je door true terug te geven.

Stap 4

Maak de CustomerFactory klasse

- ❏ Hierin maak je een final array van String objecten en initialiseer deze onmiddellijk met een aantal namen.

- ❏ Maak een static methode getCustomer die als returnwaarde een AbstractCustomer geeft en een parameter String name heeft.

    - ❏ Je itereert in deze methode over je array van String objecten en kijkt als je op de index een waarde vind die gelijk is aan het argument name dat werd meegegeven aan deze methode.

        - ❏ Indien dit het geval is zal je een new RealCustomer teruggeven die als argument de name mee krijgt

        - ❏ Indien je klaar bent met itereren maar geen match vond geef je een new NullCustomer terug.

Stap 5

Gebruik nu de CustomerFactory klasse om ofwel RealCustomer objecten of NullCustomer objecten te creëeren op basis van de naam die je meegeeft.

Zie vb:

```
public class NullPatternDemo {
    public static void main(String[] args) {
        AbstractCustomer customer1 = CustomerFactory.getCustomer("Rob");
        AbstractCustomer customer2 = CustomerFactory.getCustomer("Bob");
        AbstractCustomer customer3 = CustomerFactory.getCustomer("Julie");
        System.out.println("Customers");
        System.out.println(customer1.getName());
        System.out.println(customer2.getName());
        System.out.println(customer3.getName());
    }
```