Week 01: How Java Works

Challenge 01: Phrase-O-Matic Revisited (30 mins)

Remember our first program Phrase-O-Matic (I know it's quite lame – I do apologise). It's a poor man's version of ChatGPT (https://openai.com/blog/chatgpt/), in fact, it is not smart at all.

Unlike ChatGPT, which was trained countless time (running reinforcement learning, I can't explain further because that is completely out-of-scope here), our Phrase-O-Matic program is very basic with predefined words stored in the form of an array. It's not smart at all, but let's do some changes that will improve the functionality of the program.

For this challenge, we will enhance the capability of Phrase-O-Matic to write a marketing paragraphs. The modified codes should be able to generate "n" number of random phrases. In order to do that, you have to refactor your code:

TODO#1 – Ask user to provide argument indicating the number of sentences required, and
TODO#2 – Create a method to generate random phrases called `generatePhrase(String [] s1, String [] s2, String [] s3)` that will return the random phrase
TODO#3 – In your `main()` method, create a loop that will generate random phrases depending on the argument given by the user.

You can refer to the skeleton code for details. Is it possible to make it more efficient? Did you notice that you have to pass the String array each time you want to generate a new phrase? Well, object to the rescue TODO#4 – Create a Phrase class, and refactor your code !

Challenge 02: NotSpotify (60 mins)

Is it possible to create a Spotify-like application? Yes, of course you can. Is Spotify built on Java? Unfortunately, that is not the case (click to find out more – if you are interested), but Netflix is built on Java (or at least when they started). We are not going to discuss the Java architecture behind these enterprise applications right now – you'll get to do that much later in the module. In this lab session, we will be looking at the basic object hierarchy that we've covered in this week lesson.

So, imagine you are building a console based Spotify (sorry, we don't want to deal with the copyright issues, just yet, that's why we are calling it "NotSpotify"). Firstly, you are going to define an abstract class called **Genre**. Secondly, create three sub-genres (**Rock**, **Pop**, and **Rap)**. But why do we need all this ?

Before the Spotify subscription model, if you remember iTunes, you have to pay for every song you download (it was 99 cents – if you take into account the inflation, you definitely should be paying more for Spotify, much less having a FREE account). Anyway, we are going to have a `setPrice(int)` abstract method in your abstract superclass, and for each genre (i.e., subclass), we will charge our user as follows:

- Pop (€2.99 for 4 stars and above, €1.99 for 3 stars, and €0.00 for 2 stars and less)
- Rock (€3.99 for 4 stars and above, €2.99 for 3 stars, and €0.99 for 2 stars and less)
- Rap (€4.99 for 4 stars and above, €3.99 for 3 stars, and €1.99 for 2 stars and less)

You will also need to create a class **Song**. When you instantiate the **Song** object, you must be able to tell title of the song, the artist, the rating of the song, the genre, and the price. Of course you can do this, because we will assign our variables in the **Song** constructor.

Just like Spotify, in our main, we should be able to create a playlist of song objects depending on the selected genre – you can check the main() method for details. Once a playlist has been created, you should be able to provide the total charge for the listed songs and print the song information belonging to a specific playlist.

To complete the task, you have to:

TODO#1 Complete the declaration and definition of our inheritance three (**Genre**, **Rock**, **Rap**, and **Pop**). Do remember to complete the body of the abstract method `setPrice(int)` in the concrete class.

TODO#2 Complete the declaration and definition of the **Song** class. Don't forget to override the toString() method to display the information of the title. Note: toString() is overridden from `java.lang.Object` – if not, you'll get the string representation of the object.

TODO#3 Check out the main() method, and refactor the codes to calculate the total fees and display song information into a method called `getPlayList(List)`. We have not covered List (or you did?), so you only need to change the relevant variables. Pay attention to the function signature.

Note: If you find the instances lame (and not too your musical preference), you can change them 😊