**PSG College of Technology, Coimbatore – 641 004**

**Department of Applied Mathematics and Computational Sciences**

**15XW63 SOFTWARE PATTERNS**

**Problem Sheet – 1**

**Class:** MSc SS **Semester:**  VI

1. The client wanted us to create a music streaming application that could enable its users to create and share song playlists known as “JukeBaux.” The music app would give its user the option to join the playlists of other users and also transfer music files according to their location.

The client approached us to develop the music streaming app that could load songs from Apple Music, Spotify, etc., and extend the app's capabilities allowing the users to collaborate with each other and share songs.

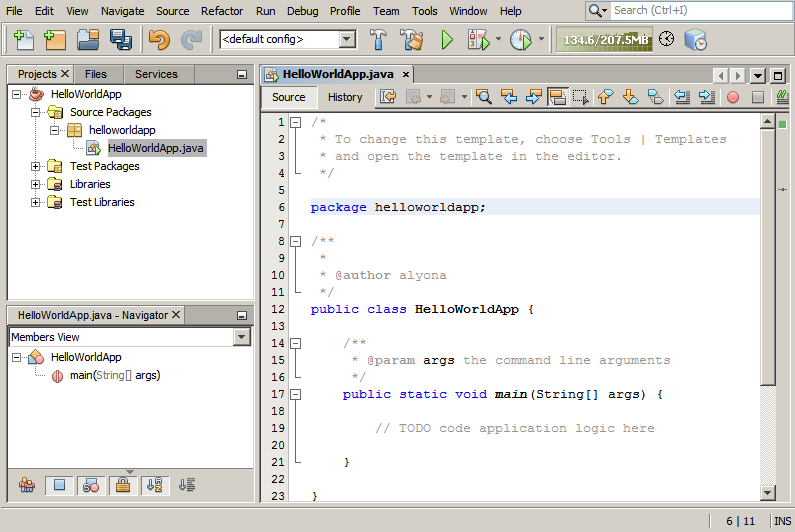
Design this application using Factory Method to create objects of Apple Music, Spotify, etc. Implement a Java program using Factory Method.

1. Design an online application for efficient work time planning in medical organizations. The application allows users to review and edit the schedule for the organization’s staff members and volunteers. It provides access to a reliable notification system for informing volunteers that they will be needed.

Implement a Java program using Abstract Factory pattern to create staff members and volunteers.

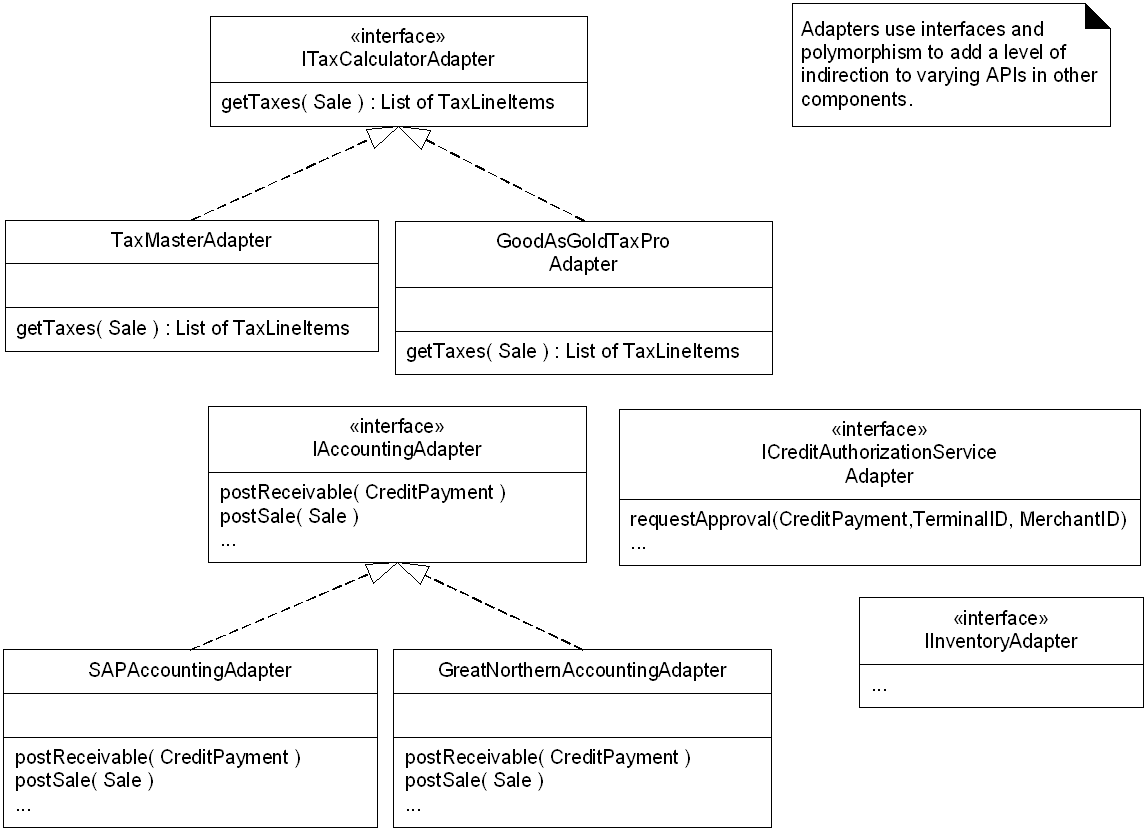
1. Design a prototype tool to simulate a software development environment to create smart programs either in C++, Java or Python to increase programmer productivity. The default code snippets may be created as files with extension **cpp, java or py**.

Implement a Java program using Prototype pattern to create the default code snippets created for every language.



1. Implement a Java program using Adapter pattern.

The figure below shows how an application can use multiple implementations (adaptees) that are functionally similar (e.g., tax calculators, accounting modules, credit authorization services, etc.) but have different APIs. We want to avoid hard-coding our domain-layer code to handle the different possible ways to calculate tax, post sales, authorize credit card requests, etc. Those are all external modules that might vary, and for which we can't modify the code. The adapter allows us to do the hard-coding in the adapter, whereas our domain-layer code always uses the same interface (the IWhateverAdapter interface).



1. Implement a Java program using Decorator pattern.

Consider a case of a pizza shop. In the pizza shop they will sell few pizza varieties and they will also provide toppings in the menu. Now imagine a situation wherein if the pizza shop has to provide prices for each combination of pizza and topping. Even if there are four basic pizzas and 8 different toppings, the application would go crazy maintaining all these concrete combination of pizzas and toppings.

Draw a class diagram and implement the above real time case study.