RALPHIE'S TUTOR

Team Number: 14
Team Members: Ruhi Saraf
Samarpita Debnath
Santhanakrishnan Ramani

Agenda

- Introduction
- Softwares Used
- Application Demo
- Design Patterns

Introduction

Ralphie Tutor

Tutor searching app where Tutors can register themselves and students can find tutors.

- Create Account & Login
- Forgot Password
- Delete, Update & View Profile
- Search & Message
- Rate & Review tutors

Software Used

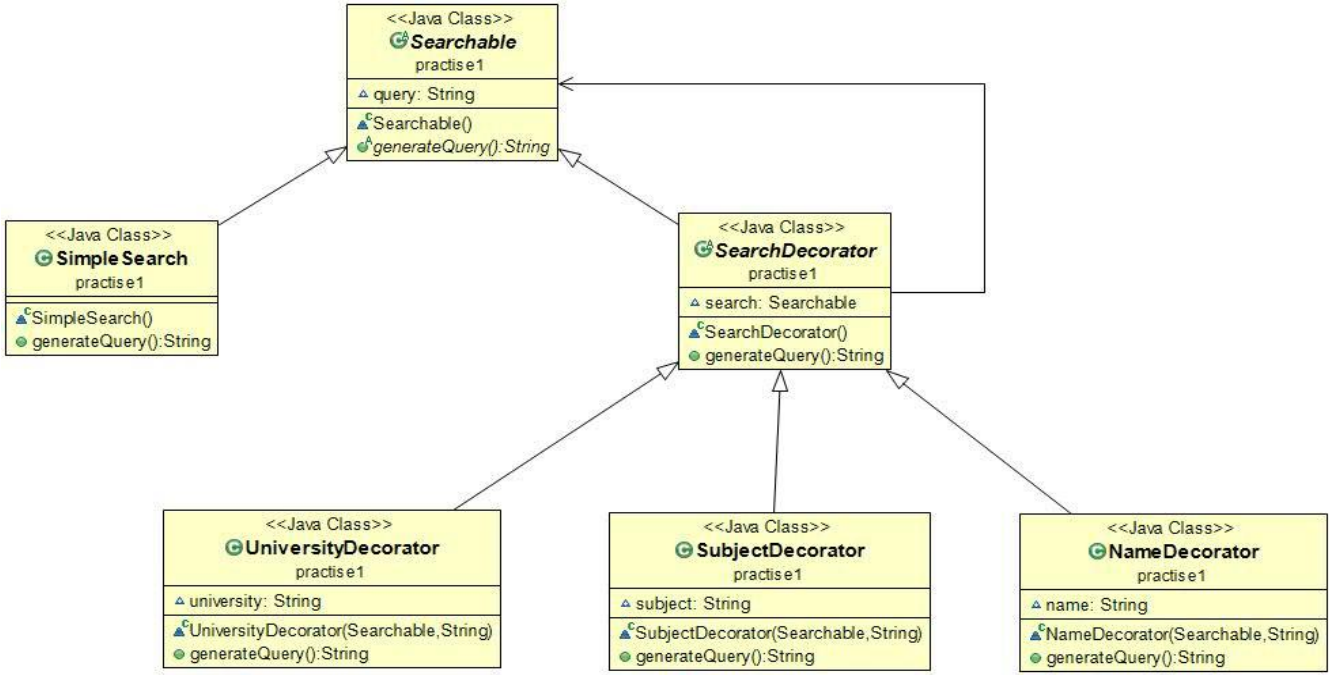
- Programming Language - Java
- IDE - Android Studio
- Data Persistence - mLab (Database-as-a-service for MongoDB)
- Rest API - Retrofit HTTP Client

Application Demo

Design Pattern - Decorator

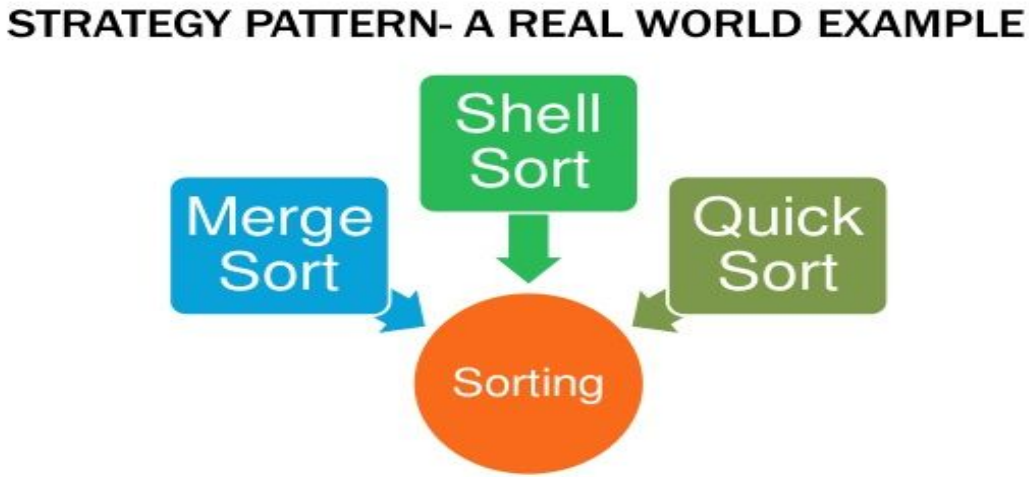
Design Pattern - Decorator

Ralphie Tutor



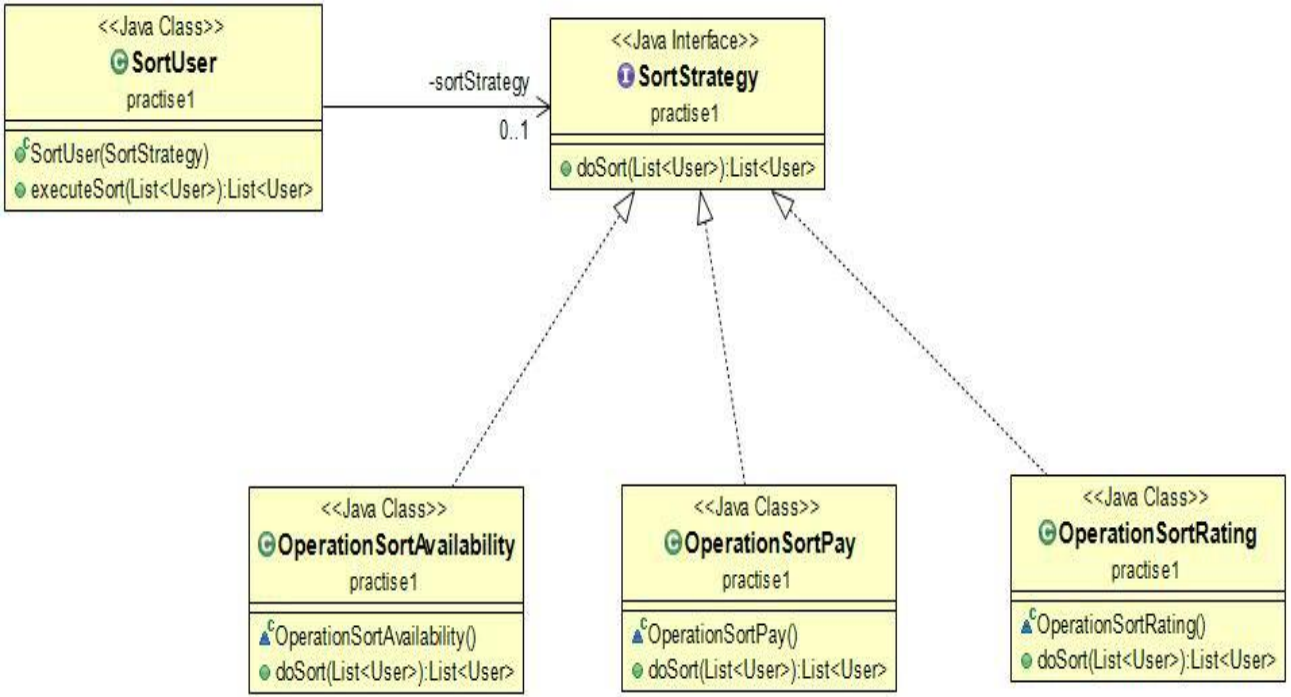
```
abstract class Searchable {
    String query;
    public abstract String generateQuery();
}
class SimpleSearch extends Searchable {
    SimpleSearch() {
        query = "SELECT * FROM users  WHERE";
    }
    public String generateQuery() {
        return query;
    }
}
abstract class SearchDecorator extends Searchable {
    Searchable search;
    public String generateQuery(){
        return query;
    }
}
class UniversityDecorator extends SearchDecorator {
    String university;
    UniversityDecorator(Searchable search, String university) {
        this.search = search;
        this.university = university;
    }

    public String generateQuery() {
        return search.generateQuery() + "  university=" + university;
    }
}
class SubjectDecorator extends SearchDecorator {
    String subject;
    SubjectDecorator(Searchable search, String subject) {
        this.search = search;
        this.subject = subject;
    }
    public String generateQuery() {
        return search.generateQuery() + " subject=" + subject;
    }
}
```

Design Pattern - Strategy



Design Pattern - Strategy

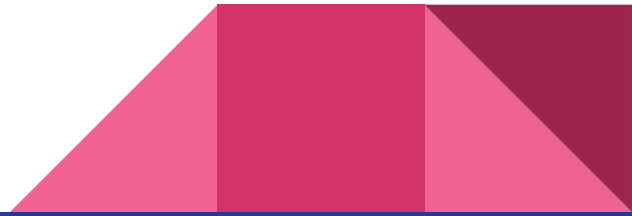
Ralphie Tutor



```
class SortUser {
    private SortStrategy sortStrategy;
    public SortUser(SortStrategy sortStrategy) {
        this.sortStrategy = sortStrategy;
    }
    public List<User> executeSort(List<User> userList){
        return sortStrategy.doSort(userList);
    }
}

interface SortStrategy {
       public List<User> doSort(List<User> userList);
}
class OperationSortRating implements SortStrategy {
    public List<User> doSort(List<User> userList) {
        return userList;
    }
}
class OperationSortPay implements SortStrategy {
    public List<User> doSort(List<User> userList) {
        return userList;
    }
}
class OperationSortAvailability implements SortStrategy {
    public List<User> doSort(List<User> userList) {
        return userList;
    }
}
```

Other Design Patterns Used

- Adapter - RecycleView Adapter, ListViewAdapters etc.
- Observable - Fragment Listeners



Thank You !