بسمه تعالی

پروژه مسیجینگ توسط گروه سه نفره از صها تولید شده است ( 1-غلامرضا فریدونپور 2- معین الوندی 3- مجتبی صیادی)

این پروژه از اینترفیسهای زیر تشکیل شده است.

1-Action

2-UserAction

3-IMessageInterface

کلاسهای زیر در این پروژه هستند :

1. Main
2. User
3. UserActionImplement
4. MyMessage
5. MessageActionImplement
6. Admin
7. Client
8. Guest

ودر نهایت از کلاس UserType که به صورت Enum تعریف شده و انواع User درآن درنظر گرفته شده است .

دراین پروژه سه پکیج جهت دسته بندی درنظرگرفته شده است .

1. DATA PACKAGE که شامل کلاس UserType است البته این کلاس از نوع Enum است.
2. MODEL PACKAGE که شامل کلاسهای اصلی پروژه میباشد.
3. SERVICE PACKAGE که شامل اینترفیسها و کلاسهای مربوط به پیاده سازی این اینترفیسها میباشد .

ابتدا در مورد اینترفیسها و کلاسها توضیحاتی داده میشود و سپس منطق کل برنامه که همان کلاس Main است شرح داده خواهد شد.

در ابتدا اینترفیس Action

public interface Action<U> {  
 void create(U u);  
 void update(String username , U u);  
 void remove(U u);  
 void remove(String username );  
 Set<U> read();  
}

همانطور که میبینید پارامترهای ورودی بصورت جنریک تعریف شده و متدهای ایجاد و به روزرسانی و حذف درآن دیده شده و متد read () که از کالکشن Set استفاده کرد ه برای دیدن تمام آبجکتها پیش بینی شده است .

اینترفیس UserAction که در واقع از اینترفیس Action ارث بری میکند در شکل زیر دیده میشود:

public interface UserAction<U> extends Action<U> {  
 U findByUserName(String username);  
 U LoginCheck(String username ,String password);  
}

متد FindUserByName در این اینترفیس دیده شده که پارامتر ورودی آن یک رشته است و خروجی آن جنریک U و متد LoginCheck که نام کاربری و رمز عبور را چک میکند وخروجی آن هم دوباره جنریک U است .

اینترفیس IMessageInterface در شکل زیر دیده میشود :

public interface IMessageInterface {  
 void SendMessage(User user,String messageText);  
 void printAllMessage(Map<Integer,MyMessage> AllMessage);  
}

این اینترفیس شامل دو متد است : 1 – SendMessage این متد 2 پارامتر user , و messageText را بعنوان ورودی دریافت میکند و در واقع با این دو پارامتر یک مسیج ایجاد میکند.

2- printAllMessage این متد قراراست که تمام مسیجهای ارسالی را چاپ کند.

کلاس user که شامل چند فیلد و چندین متد است در شکل زیر دیده میشود :

public class User {  
 private int id;  
 private String firstName;  
 private String lastName;  
 private int age;  
 private String phone;  
 private String address;  
 private String username;  
 private String password;  
 public UserType type;  
  
 public int getAge() {  
 return age;  
 }  
 public void setAge(int age) {  
 this.age = age;  
 }  
 public String getPhone() {  
 return phone;  
 }  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
 public String getAddress() {  
 return address;  
 }  
 public void setAddress(String address) {  
 this.address = address;  
 }  
 public String getUsername() {  
 return username;  
 }  
 public void setUsername(String username) {  
 this.username = username;  
 }  
 public String getPassword() {  
 return password;  
 }  
 public void setPassword(String password) {  
 this.password = password;  
 }  
 public UserType getType() {  
 return type;  
 }  
 public void setType(UserType type) {  
 this.type = type;  
 }  
 public int getId() {  
 return id;  
 }  
 public void setId(int id) {  
 this.id = id;  
 }  
 public String getFirstName() {  
 return firstName;  
 }  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
 public String getLastName() {  
 return lastName;  
 }  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
}

هم فیلدها و هم متدها پیچیدگی خاصی ندارد و نیازی به توضیح اضافه نیست.

درواقع در این کلاس همه فیلدها private بوده وتنها بوسیله متدها قابلیت تغییر دارند یعنی Encapsulation

کلاس بعدی UserActionImplement است که در شکل زیر دیده میشود :

public class UserActionImpl implements UserAction<User>{  
 @Override  
 public void create(User user) {  
 Main.*users*.add(user);  
 }  
  
 @Override  
 public void update(String username, User userb) {  
 User user1=findByUserName(username);  
 userb.setType(user1.getType());  
 userb.setPassword(user1.getPassword());  
 remove(user1);  
 create(userb);  
 }  
 @Override  
 public void remove(User user) {  
 Main.*users*.remove(user);  
 }  
  
 @Override  
 public void remove(String username) {  
 User user1=findByUserName(username);  
 remove(user1);  
 }  
  
 @Override  
 public Set<User> read() {  
 return Main.*users*;  
 }  
  
 @Override  
 public User findByUserName(String username) {  
  
 for (User u1 : read()) {  
 if (u1.getUsername().equals(username))  
 return u1;  
 }  
 return null;  
 }  
 public User LoginCheck(String username ,String password){  
 User user=findByUserName(username);  
 if(user != null){  
 if(user.getPassword().equals(password))  
 return user;  
 else  
 return null;  
 }  
 return null; }  
}

این کلاس از اینترفیس UserAction ارث بری میکند و تمام متدهای آنرا پیاده سازی کرده است .

کلاس اصلی این پروژ در واقع کلاس MyMessage است که در شکل زیر ساختار آن دیده میشود :

public class MyMessage {  
 private String MessageText;  
 private User user;  
 private LocalDate SendDate;  
  
 public String getMessageText() {  
 return MessageText;  
 }  
 public void setMessageText(String messageText) {  
 MessageText = messageText;  
 }  
 public User getUser() {  
 return user;  
 }  
 public void setUser(User user) {  
 this.user = user;  
 }  
  
 public LocalDate getSendDate() {  
 return SendDate;  
 }  
 public MyMessage(String messageText, User user) {  
 MessageText = messageText;  
 this.user = user;  
 setSendDate();  
 }  
 private void setSendDate() {  
 SendDate =LocalDate.*now*();  
 }  
 @Override  
 public String toString() {  
 return "Message{ " +  
 "Text = '" + MessageText + '\'' +  
 ", user = " + user.getUsername() +  
 ", SendDate = " + SendDate +  
 '}' ;  
 }  
}

هر مسیج با سه فیلد متن مسیج و کاربر ارسال کننده و تاریخ ارسال مشخص میشود و در ساختار map که از نوع LinkedHashMap بوده وبا عنوان AllMessage ذخیره میشود ، البته این کار در کلاس Main انجام میشود.

کلاس MessageActionImplementation از اینترفیس IMessageInterface ارث بری میکند و متدهای آن را پیاده سازی میکند .

این کلاس شامل دو متد SendMessage , printAllMessageمیباشد.

public class MessageActionImpl implements IMessageInterface{  
 @Override  
 public void SendMessage(User user, String messageText) {  
 MyMessage myMessage = new MyMessage(messageText, user);  
 Main.*AllMessage*.put(Main.*MapKey*,myMessage );  
 }  
 @Override  
 public void printAllMessage(Map<Integer, MyMessage> AllMessage) {  
 for (Map.Entry<Integer,MyMessage> q :AllMessage.entrySet()){  
 System.*out*.println(q);  
 }  
 }  
}

کلاس Main که کلاس اصلی بوده و شامل منطق و منوهای اصلی برنامه است در زیر توضیح داده میشود .

قبل از متد main متغیرهای static تعریف میشوند که بتوانیم کاربران و مسیج ها را ذخیره و مدیریت کنیم :

public static Set<User> *users* = new HashSet<>();  
public static Map<Integer,MyMessage> *AllMessage* = new LinkedHashMap<>();  
public static Integer *MapKey* = 1;

در متد main بعداز تعریف ومقدار دهی متغیرهای مورد نیاز دو نمونه از کلاسهای مورد نیاز نیز ایجاد میشود :

UserActionImpl userAction = new UserActionImpl();  
MessageActionImpl messageAction = new MessageActionImpl();

با استفاده از کلاس User یک کاربر User admin و یک کاربر User gust ساخته میشود :

User user1 = new User();  
user1.setUsername("admin");  
user1.setPassword("abcd");  
user1.setType(UserType.*ADMIN*);  
userAction.create(user1);  
//----------------------------------  
User user2 = new User();  
user2.setUsername("gust");  
user2.setPassword("abcd");  
user2.setType(UserType.*GUEST*);  
userAction.create(user2);

در ابتدا منوی اصلی که شامل سه حالت LOGIN ,REGISTER , EXIT میباشد ظاهر خواهد شد .

در صورتیکه عددی بین 1و3 وارد شود منو انتخاب میشود و درغیراینصورت پیغام خطا صادر شده و دوباره منتظر انتخاب شما میماند .

while (true) {  
 System.*out*.println(" 1- Login user ");  
 System.*out*.println(" 2- Register new user ");  
 System.*out*.println(" 3- Exit ");  
 do {  
 System.*out*.print(" Enter action (1 -3) : ");  
 while (!scanner.hasNextInt()) {  
 System.*out*.print(" Error \n Please enter just number ");  
 scanner.next();  
 } //---- End While  
 inputMenuUser1 = scanner.nextInt();  
 if (inputMenuUser1 <= 0 || inputMenuUser1 > 3)  
 System.*out*.print(" Please enter number between 1 and 3 : ");  
  
 }//---- End do While  
 while (inputMenuUser1 <= 0 || inputMenuUser1 > 3);

در صورت انتخاب منوی LOGIN نام کاربری و رمز عبور را از ورودی گرفته ودر صورتیکه هر دو مورد درست باشد و کابر از نوع 1 یعنی ADMIN باشد منوی اصلی که شامل 7 مورد است نمایش داده میشود.

System.*out*.print(" Please Enter UserName : ");  
username= scanner.next();  
System.*out*.println(" ------------------------------- ");  
System.*out*.print(" Please Enter PassWord : ");  
password= scanner.next();  
//-------------------------------------------------------  
User UserSuccess = userAction.LoginCheck(username,password);  
if(UserSuccess != null ) {  
 if (UserSuccess.getType() == UserType.*ADMIN*) {  
 while (true) {  
  
 System.*out*.println(" 1- Create User ");  
 System.*out*.println(" 2- Update User ");  
 System.*out*.println(" 3- Remove User ");  
 System.*out*.println(" 4- Show all User ");  
 System.*out*.println(" 5- Send Message ");  
 System.*out*.println(" 6- Show Message ");  
 System.*out*.println(" 7- Exit ");  
 do {  
 System.*out*.print(" Enter action (1 -7) : ");  
 while (!scanner.hasNextInt()) {  
 System.*out*.print(" Error \n Please enter just number ");  
 scanner.next();  
 } //---- End While  
 inputMenuUser = scanner.nextInt();  
 if (inputMenuUser <= 0 || inputMenuUser > 7)  
 System.*out*.print(" Please enter number between 1 and 7 : ");  
  
 }//---- End do While  
 while (inputMenuUser <= 0 || inputMenuUser > 7);

درصورت انتخاب منوی اول یعنی ایجاد کاربر ابتدا نوع کابراز شما پرسیده میشود و سپس نام و رمز عبور کاربر جدید باید وارد کنید .

System.*out*.println(" ----------------------------------------");  
System.*out*.println(" 1- Admin ");  
System.*out*.println(" 2- Client ");  
System.*out*.println(" 3- Guest ");  
System.*out*.println(" 4- Exit ");  
//-----------------------------  
do {  
 System.*out*.print(" Enter action (1 -4) : ");  
 while (!scanner.hasNextInt()) {  
 System.*out*.println(" Error \n Please enter just number ");  
 scanner.next();  
 } //---- End While  
 inputUserType = scanner.nextInt();  
 if (inputUserType <= 0 || inputUserType > 4)  
 System.*out*.println(" Please enter number between 1 and 4 ");  
  
}//---- End do While  
while (inputUserType <= 0 || inputUserType > 4);  
System.*out*.println(" -------------------------------------------- ");  
System.*out*.print(" Enter UserName : ");  
username = scanner.next();  
System.*out*.println(" -------------------------------------------- ");  
System.*out*.print(" Enter PassWord : ");  
password = scanner.next();

بسته به نوع کاربر انتخابی کد زیر جهت ایجاد آن اعمال میگردد :

switch (inputUserType) {  
 case 1:  
 User user = new User();  
 user.setUsername(username);  
 user.setPassword(password);  
 user.setType(UserType.*ADMIN*);  
 userAction.create(user);  
 //-------------------------  
 System.*out*.println(" user admin created ");  
 break;  
 case 2:  
 User user4 = new User();  
 user4.setUsername(username);  
 user4.setPassword(password);  
 user4.setType(UserType.*CLIENT*);  
 userAction.create(user4);  
 //---------------------------------  
 System.*out*.println(" user client created ");  
 break;  
 case 3:  
 User user3 = new User();  
 user3.setUsername(username);  
 user3.setPassword(password);  
 user3.setType(UserType.*GUEST*);  
 userAction.create(user3);  
 //---------------------------------  
 System.*out*.println(" user guest created ");  
 break;

منوی بعدی به روزرسانی یک کاربر است :

System.*out*.print(" Enter old UserName FOR UPDATE : ");  
oldUserName = scanner.next();  
System.*out*.print(" Enter new UserName : ");  
newUserName = scanner.next();  
User user4 = new User();  
user4.setUsername(newUserName);  
userAction.update(oldUserName, user4);  
System.*out*.println(" updated user successfully ");

همانطور که در کد میبینید USER قدیمی و UDER جدید از شما دریافت شده ، کاربر قدیمی ازسیستم حذف وکاربر جدید به جای آن اضافه میشود .

منوی بعدی حذف یک کاربر میباشد که در کد زیر مشاهده میشود:

System.*out*.print(" Enter UserName TO REMOVE : ");  
oldUserName = scanner.next();  
userAction.remove(oldUserName);  
System.*out*.println(" removed user successfully");

منوی بعدی نمایش همه کاربران سیستم میباشد :

for (User u1 : userAction.read())  
 System.*out*.println(u1.getUsername() + "(" + u1.getType() + ")");

منوی شماره 5 که جهت ایجاد Message میباشد در کد زیر نمایش داده شده است :

System.*out*.print(" Please Enter your Message to Send : ");  
scanner.nextLine();  
userMessageText = scanner.nextLine();  
messageAction.SendMessage(UserSuccess,userMessageText);  
System.*out*.println(" your Message sent successfully ");  
*MapKey* ++;

همانطور که میبینید پیامهای ارسالی با تاریخ و مشخصات فرستنده در map ذخیره میشود .

منوی شماره 6 که قرار است کلیه پیامها و تاریخ ارسال و کاربر فرستنده را نمایش دهد توسط قطعه کد زیر نمایش داده میشود :

messageAction.printAllMessage(*AllMessage*);

و اما منوی آخر که با توجه به کد آن باعث خروج از سیستم شده و اجرای برنامه را متوقف میکند :

System.*out*.println(" Thank you .");  
*exit*(0);