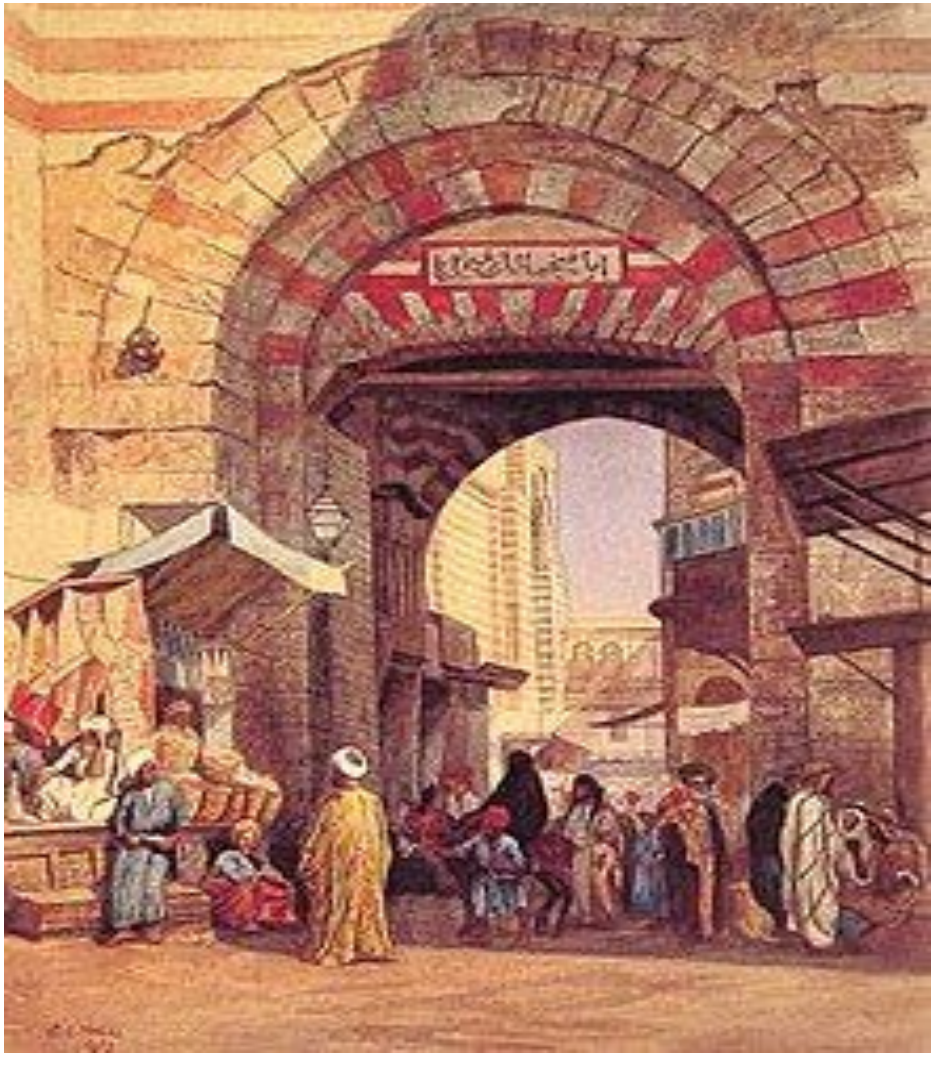
# Bazaar



# SystemInOut (command line interface)

Responsibilities:

- getInput: gets a users input
- sendOutput: prints a string to the user

Collaborators:

Collaborate with all controllers

```java
/**
 * Takes in input from the user as a String and gives output back to the user.
 */

public class SystemInOut implements InOut {
    BufferedReader reader;

    public SystemInOut() { reader = new BufferedReader(new InputStreamReader(System.in)); }

    // get user input
    public String getInput() throws IOException {
        return reader.readLine();
    }

    // send user a message they can respond to
    @Override
    public void sendOutput(Object s) { System.out.println(s); }
}
```

# Master (entity class)

## Responsibilities:

- Stores all users corresponding to the username of the user
- Stores all products corresponding to the product ID
- Stores the list of products corresponding to the tag word that is associated with that list of products.

## Collaborators:

User
Product

```java
import ...

/**
 * A class that store all users and products in the marketplace
 */
public class Master{
    // a dictionary mapping the string usernames to users
    public static HashMap<String, User> userDict = new HashMap<>();
    // a dictionary mapping the string product IDs to products
    public static HashMap<String, Product> productDict = new HashMap<>();
    // a dictionary mapping the string tag word to list of products
    public static HashMap<String, ArrayList<Product>> productTagMap = new HashMap<>();



}
```

# masterManager (use case)

## Responsibilities:

- Get a user from the master from a given username
- Accesses users when the user inputs correct username
- Save the username of a user as keys and the profile of the user as values
- Product getter based on product ID and product list getter based on tag word.
- User dictionary setter(add users to master)
- Product dictionary setter(add products to master)
- Product tag map which maps tags to the products.
- Product tag map getter to get the list of products associated with a tag.
- Add product to the Master product data base and add the product to dictionary associated with the tag.
-

## Collaborators:

- Master

```java
/**
 * A use case class that accesses and modulated manager entity.
 */
public class masterManager{

    /**
     * A getter of a user from the User dictionary. Which has the username of the user as Strings and the User as the
     * If the username exit in the dictionary, return the user, othermwise, return a string indicating that the
     * user does not exist.
     * If the object does not exist, return false, and the command line interface will give a message that
     * the product does not exist.
     * @return the userDict
     */
    public static Object getterUser(String username){
        if (Master.userDict.containsKey(username)){
            return Master.userDict.get(username);
        }else{
            return false;
        }
    }

    /**
     * A getter of the products dictionary. Which has the ID of the product of the user as Strings
```

# loginOptionsController (Controller)

Requirements:

- userInput method asks if the user wants to sign in or sign up
- If they select sign in,
  - It prompts the user to sign in via sign in controller
  - Runs OptionController to continue with options to browse, search, etc.
- If they select sign up
  - If directs therm to createUserController to create the User

Collaborators

- createUserController
- SignInController

```
switch (userDecision) {
    case "signin":
        signInController newSignInController = new signInController();
        Object existingUser = newSignInController.userSignIn(inOut);
        if (existingUser instanceof User) {
            userOptionsController optionsController = new userOptionsController((User) existingUser);
            optionsController.userInput(inOut);
            return true;
        }
        inOut.sendOutput( s: "Something went wrong");
        return false;
    case "signup":
        createUserController newUserController = new createUserController();
        newUserController.userCreator(inOut);
        return true;
}
```

# createUserController (controller)

Responsibilities:

- Prompt the user to input their new username which is called userSignUp
- userCreater takes input from the user and creates an new User entity via UserManager

Collaborators:

- userManager

```java
boolean validUsername = false;
String username;

// code to enforce a limit to the length of the username, 20 characters max
while (!validUsername){
    username = inOut.getInput();
    // check if user can be created, if not then username is either too long or has already been taken
    if (username.length() <= 20 && userManagerInstance.createUser(username)){
        validUsername = true;
        inOut.sendOutput( s: "User profile successfully created.");
        // print out username for the user to look over
        inOut.sendOutput( s: "Username: " + username);
    }
    else if (username.length() > 20){
        inOut.sendOutput( s: "Username is too long. Try again.");
    }
    else{
        inOut.sendOutput( s: "Username has already been taken. Try again.");
    }
}
```

# userManager(use case)

## Responsibilities:

- (public) Methods:
  - createUser (creates a new user)
  - changeUsername (changes Username of the user)
  - addToShoppingCartSearches (adds a product to the user's shopping cart based on index of Product in user's current Searches list)
  - addToShoppingCartFeed (adds a product to user's shopping cart associated with post at index entered which is index of post in the user's feed)
  - emptyShoppingCart (removes all products from a user's shopping cart)
  - addToPostList (adds a new post to the user's list of posts)
  - addFollowing (adds a new following (type User) to another User's list of following)
  - verifyUser (changes user to being verified, isVerified = true)
  - getFeedList (gets feed of the user)

## Collaborators:

- User (entity class)
- masterManager (use case)
- Post

# User (entity class)

## Responsibilities:

- Attributes of user: username, listPosts, shoppingCart, currentSearches, listfollowing, feed, isVerified, password (to be implemented later)
- constructor (sets username to username, password to password - implementation later and everything else to empty arraylists)
- constructor
- Getter and setter for username, listPosts, shoppingCart, currentSearches, listfollowing, feed, isVerified

## Collaborators:

- Post
- Product

```java
public static boolean createUser(String username){
    User newUser = new User(username);
    // this username does not exist
    if (checkUsernameStatus(username)){
        // this username not already in use
        // Add user to Hashmap that keeps track of all users
        masterManager.setterUser(username, newUser);
        return true;
    }
    // user not added, username already in use
    return false;
}
```

Signup/new User

```
"C:\Users\Umayrah Chonee\.jdks\corretto-11.0.12\bin\java.exe" "-jav
What would you like to do? Input one of signin or signup
signup
Input username (must be less than 20 characters):
Umayrah
User profile successfully created.
Username: Umayrah
What would you like to do? Input one of signin or signup
signup
Input username (must be less than 20 characters):
Yasmin
User profile successfully created.
Username: Yasmin
What would you like to do? Input one of signin or signup
```

# signInController (controller)

Responsibilities:

- UserSignIn method takes input of the username and checks if it is a valid username via masterManager that has been created already and if it is, it lets you sign in and if it does not, it prompts the user to input valid username

Collaborators:

- masterManager

```
String username;

// code to enforce a limit to the length of the username, 20 characters max
while (true){
    username = inOut.getInput();
    // check if user can be created, if not then username is either too long or has already been taken
    Object result = masterManager.getterUser(username);
    if (result instanceof User){
        inOut.sendOutput( s: "User profile has been opened.");
        return result;
    }
    else{
        inOut.sendOutput( s: "Please input a valid username or rerun the program to signup.");
    }
}
```

# userOptionsController (controller)

Responsibilities:

- Prompt the user to input search, create post, Browse, or buy from cart
- From this information, it runs methods in specific controllers or user manager to perform what they asked

Collaborators:

- userManager
- createPost (controller)
- Search (controller)
- Browse (controller)

```
if (Objects.equals(userDecision, b: "post")) {
    // redirects to createPostController class
    createPostController postController = new createPostController();
    postController.postCreator(input, this.user);
}
else if(Objects.equals(userDecision, b: "browse")) {
    // redirects to browseController and return feed
    browseController browseController = new browseController();
    browseController.searchFeed(input, this.user);
}
else if (Objects.equals(userDecision, b: "search"))  {
    // redirects to searchController and returns relevant search info
    searchController searchController = new searchController();
    searchController.searchProducts(input, this.user);
}
else{ //(Objects.equals(userDecision, "Buy Cart"))
    userManager userManager = new userManager();
    userManager.emptyShoppingCart(user);
    input.sendOutput( s: "Shopping cart is empty.");
}
```

Signin

```
"C:\Users\Umayrah Chonee\.jdks\corretto-11.0.12\bin\java.exe" "-javaagent:C:\
What would you like to do? Input one of signin or signup
signup
Input username (must be less than 20 characters):
Umayrah
User profile successfully created.
Username: Umayrah
What would you like to do? Input one of signin or signup
signin
Input username:
Umayrah
User profile has been opened.
What would you like to do? Input one of Search, Post, Browse, Buy Cart
```

# createPostController (controller)

Responsibilities:

- postCreator, takes in the user and gets input from that user about the options for the post that they want to create.
- Relays this information to the postManager via createPost method from postManager

Collaborators:

- postManager

```
inOut.sendOutput("Input the Post Description");
String postdescription = inOut.getInput();

inOut.sendOutput("Input the number of Tags you want the Post to have");
int tagnumber = Integer.parseInt(inOut.getInput());

ArrayList<String> tags = new ArrayList<>();

for (int counter = 0; counter < tagnumber; counter++){
    inOut.sendOutput("Input a Tag for the post");
    tags.add(inOut.getInput());
}
if (sizetf){
    postmanager.createPost(tags,user,postdescription,product_name,id,price,category,sizeOther,quantity);
}
else{
    postmanager.createPost(tags,user,postdescription,product_name,id,price,category,quantity);
}
inOut.sendOutput("Post has been created");
```

```
Input username:
yasmin
User profile has been opened.
What would you like to do? Input one of Search, Post, Browse, Buy Cart
Post
Input Product Name:
coat
Input Product Price:
22.22
Input Product Category:
outfits
Input Product ID:
1213
Input Product Quantity
1
Input true or false If product Has a size
```

```
Input Product ID:
1213
Input Product Quantity
1
Input true or false If product Has a size
true
Input the Product size
12
Input the Post Description
this is a coat
Input the number of Tags you want the Post to have
1
Input a Tag for the post
dress
Post has been created
```

# postManager(use case)

## Responsibilities:

- CreatePost method to take information from the user and create a product using product Manager
- Adds the post to the post list for user via user manager
- Adds product to the tag map using the given tag via the master manager

## Collaborators:

- productManager
- Post
- Usermanager
- masterManager

```
public Post createPost (ArrayList<String > t, User u, String d, String productName, String id, float price,
                        String category, String size, int quantity){
    Product p = productmanager.createProduct(productName, id, price, category, size, quantity);
    Post post = new Post(p,t,d,u);
    usermanager.addToPostList(post,u);
    for (String tag : post.getTags()) {
        masterManager.setterProductTagMap(tag, post.getPost_topic());
    }
    return post;
}
```

# Post (entity class)

## Responsibilities:

- Constructor
  - Create the post
-
- instance attributes: product: type Product, tag words for product,, userMadePost: User, description
- getPost_Topic: return the product that it represents
- getTags return ArrayList<String> of tags for the post
- getDescription return the description of the post

## Collaborators:

- Product
- User

# productManager(use case)

## Responsibilities:

- Create product directly from the information and adds in to the master via the masterManager
- Increase the quantity of a product
- Decrease quantity of product
- Check if the product ID is in in the database

## Collaborators:

- Product
- MasterManager

```java
public Product createProduct(String name, String id, float price, String category, String size, int quantity) {
    Product newProduct = new Product(name, id, price, category, size, quantity);
    if (checkProductStatus(id)){
        //product not yet created
        masterManager.setterProduct(id, newProduct);
        return newProduct;
    }
    return null;
}
```

# Product (entity class)

## Responsibilities:

- Constructor
  - Create the product
  - Holds information about the product
- instance attributes: name, price (float that will be added to the price of other products in the cart of the user), category, size(None, or list of sizes)
- Attribute: basically the description of the product
- Getters and setters
  - Price, dimensions, colours etc.

## Collaborators:

None

# searchController (controller)

Takes in user input (tag they are searching for), then it searches the master for any product that matches the input via the masterManager and it adds to this user's currentSearches attribute via cartController

```
What would you like to do? Input one of signin or signup
Signin
Input username:
Diego
User profile has been opened.
What would you like to do? Input one of Search, Post, Browse, Buy Cart
Search
Search by tag:
shoes
[Product@22f71333]
```

# cartController (controller)

Takes in user input to add desired product to the user's cart from the search option. Buy cart empties the user's cart and reduces the quantity value of the product by one.

```
shoes
[Product@22f71333]
What product would you like to buy? (Input index):
0
Successfully added to cart.
What would you like to do? Input one of signin or signup
signin
Input username:
Diego
User profile has been opened.
What would you like to do? Input one of Search, Post, Browse, Buy Cart
Buy Cart
Shopping cart is empty.
```

# browseController (Controller)

Responsibilities:

- Search the feed of the user in searchFeed
- Output the feed to the user by accessing this user' followers' selling items via the user manager
- Calls cart controller to prompt if they wish to buy anything in their feed

Collaborators:

- userManager
- cartController

Next Steps: give options to follow

Main.java | browseController.java | SystemInOut.java | Master.java | masterManager.java | userOptionsController.java

```java
/**
 * Gives user four options: Search, Post, Browse or Buy Cart.
 *
 * @param input an object that implements InOut interface
 * @throws IOException
 */
public void userInput(SystemInOut input) throws IOException {

    try{
        input.sendOutput( s: "What would you like to do? Input one of Search, Post, Browse, Buy Cart");
        String userDecision = input.getInput().toLowerCase();
            if (Objects.equals(userDecision, b: "post")) {
                // redirects to createPostController class
                createPostController postController = new createPostController();
                postController.postCreator(input, this.user);
            }
            else if(Objects.equals(userDecision, b: "browse")) {
                // redirects to browseController and return feed
                browseController browseController = new browseController();
                browseController.searchFeed(input, this.user);
            }
            else if (Objects.equals(userDecision, b: "search"))  {
                // redirects to searchController and returns relevant search info
                searchController searchController = new searchController();
                searchController.searchProducts(input, this.user);
            }
            else{ //(Objects.equals(userDecision, "Buy Cart"))
                userManager userManager = new userManager();
                userManager.emptyShoppingCart(user);
                input.sendOutput( s: "Shopping cart is empty.");
```

```java
    /**
     * Takes in a User and an object that implements inOut and passes the User's
     * feed to the cart Controller so that User can buy from their feed.
     *
     * @param inOut an object that implements InOut interface
     * @param user A User object
     */

    public void searchFeed(InOut inOut, User user){
        userManager userManager = new userManager();
        List<Post> PostList = userManager.getFeedTotal(user);
        inOut.sendOutput(PostList);
        if (!PostList.isEmpty()) {
            cartController cart = new cartController();
            cart.addToCartFeed(inOut, user);
        } else {
            inOut.sendOutput( s: "There was an error. You are not currently following anyone.");
        }
    }
}
```

```
Input username:
Please input a valid username or rerun the program to signup.
yasmin
User profile has been opened.
What would you like to do? Input one of Search, Post, Browse, Buy Cart
browse
[]
There was an error. You are not currently following anyone.
What would you like to do? Input one of signin or signup
```

# What went well

- Clear user instructions
- Lots of different functions
-

# Improvements

- Back button/command
- More tests
- Print product as a string (for search)
- Allow multiple users to use program

# Next Steps

- Purchase through bank
- Chat with other users
- Interact with posts
- Display images
- Store information via cloud

# Questions

- How does a user purchase items
- How do we use the cloud