Code for Stack Overflow

Write the object-oriented code to implement the design of the Stack Overflow problem.

We'll cover the following

We've gone over the different aspects of Stack Overflow and observed the attributes attached to the problem using various UML diagrams. Let us now explore the more practical side of things, where we will work on implementing the Stack Overflow network using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in Stack Overflow:

- Java
- C#
- Python
- C++
- JavaScript

Stack Overflow classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

> **Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

Constants

The following code provides the definition of the various enums and custom data types being used in the Stack Overflow design:

> **Note:** JavaScript does not support enumerations so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

```
enum AccountStatus {
  ACTIVE,
  BLOCKED,
  BANNED,
  DISABLED
}

enum QuestionStatus {
  ACTIVE,
  CLOSED,
  FLAGGED,
  BOUNTIED
}

enum ClosingDetail {
  COMMUNITY_SPECIFIC_REASON,
  DUPLICATE,
  NEEDS_CLARITY,
  NEEDS_MORE_FOCUS,
  OPINION_BASED
}
```

Constant definitions

Account

The `Account` class refers to an account of a user on Stack Overflow and is responsible for containing their personal details, such as the username, password, etc. It also allows users to reset their existing passwords. The definition of this class is given below:

```
public class Account {
  private String accountId;
  private String username;
  private String password;
  private String name;
  private String email;
  private int phone;
  private AccountStatus status;

  public boolean resetPassword();
}
```

The Account class

User, admin, moderator, and guest

The `User` class will be a parent class that represents a regular Stack Overflow user. A normal user can also be an `Admin` and a `Moderator`. Another actor is represented by the `Guest` class that refers to a user who can only search and view questions as well as their answers. However, they need to register an account to ask or answer questions. The definition of these classes is provided below:

```
public class User {
  private int reputationPoints;
  private Account account;
  private List<Badge> badges;

  public boolean createQuestion(Question question);
  public boolean addAnswer(Question, question, Answer answer);
  public boolean createComment(Comment comment);
  public boolean createTag(Tag tag);
  public void flagQuestion(Question question);
  public void flagAnswer(Answer answer);
  public void upvote(int id);
  public void downvote(int id);
  public void voteToCloseQuestion(Question question);
  public void voteToDeleteQuestion(Question question);
  public void acceptAnswer(Answer answer);
}

public class Admin extends User {
  public boolean blockUser(User user);
  public boolean unblockUser(User user);
  public void awardBadge(User user, Badge badge);
}

public class Moderator extends User {
  public void closeQuestion(Question question);
  public void reopenQuestion(Question question);
  public void deleteQuestion(Question question);
  public void restoreAction(Question question);
  public void deleteAnswer(Answer answer);
```

The User, Admin, Moderator, and Guest classes

Question, answer, comment, and bounty

Stack Overflow users can create and answer questions, upvote and downvote them, and add bounties and comments to questions. The definition of these classes is provided below:

```
public class Question {
  private int id;
  private String title;
  private String content;
  private User createdBy;
  private int upvotes;
  private int downvotes;
  private int viewCount;
  private int score;
  private int voteCount;
  private Date creationDate;
  private Date modificationDate;
  private QuestionStatus status;
  private ClosingDetails closingReason;
  private Bounty bounty;

  private List<Tag> tags;
  private List<Comment> comments;
  private List<Answer> answers;
  private List<User> followers;

  public void addComment(Comment comment);
  public void addBounty(Bounty bounty);
}

public class Comment {
  private int id;
  private String content;
  private int flagCount;
  private int upvotes;
  private
```

The Question, Answer, Comment, and Bounty classes

Badge, tag, and tag list

Users can have badges that act as their reputation awards. Questions can have tags that describe the category that the question falls in. To keep a count of the tags being used, the `TagList` class is used. The definition of these classes can is provided below:

```
public class Badge {
  private String name;
  private String description;
}

public class Tag {
  private String name;
  private String description;
}

public class TagList {
  private HashMap<Tag, int> tagsCount;
  public void incrementTagCount();
  public void decrementTagCount();
}
```

The Badge, Tag, and TagList classes

Notification

The `Notification` class is responsible for sending notifications to users about any new messages, comments, posts, or friend requests via either a phone number, or an email. Its definition is provided below:

```
public class Notification {
  private int notificationId;
  private Date createdOn;
  private String content;

  public boolean sendNotification(Account account);
}
```

The Notification class

Search catalog and interface

The `SearchCatalog` class contains information on existing questions and answers. It also implements the `Search` interface class to enable the search functionality based on the given criteria (tags, usernames, and searched keywords). The definition of these two classes is provided below:

```
public interface Search {
  public List<Question> searchByTags(String name);
  public List<Question> searchByUsers(String name);
  public List<Question> searchByWords(String words);
}

public class SearchCatalog implements Search {
  private HashMap<String, List<Tag>> questionsUsingTags;
  private HashMap<String, List<User>> questionsUsingUsers;
  private HashMap<String, List<String>> questionsUsingWords;

  public List<Question> searchByTags(String name) {
    // functionality
  }

  public List<Question> searchByUsers(String name) {
    // functionality
  }

  public List<Question> searchByWords(String words) {
    // functionality
  }
}
```

The Search interface and the SearchCatalog class

Wrapping up

We've explored the complete design of Stack Overflow in this chapter. We've looked at how Stack Overflow can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.

Back

Activity Diagram for Stack Overflow

Complete

Next

Getting Ready: The Restaurant Management System