National University of Computer & Emerging Sciences, Karachi
Spring-2024, School of Computing (BSCS, BSSE, BSAI, BSCY)

Assignment # 2

Subject: Object Oriented Programming **Total Marks: 40**	**Post Date: 17th March 2024** **Due Date: 31st March 2024**
Course Instructors: Mr. Basit Ali, Ms. Sobia Iftikhar, Ms. Sumaiyah Zahid, Ms. Abeeha Sattar, Ms. Bakhtawer Abbasi, Ms. Atiya Jokhio, Mr. Minhal Raza, Ms. Rafia Shaikh, Ms. Abeer Gauher	

Instructions to be strictly followed.

---

1. Each student should submit these files:
   a. A zip of all source files named as "A2-Q#[StudentID]" where # is the question number and Student ID is your ID.
   b. A DOC file where they copy code for each question and screen shot of the output. This document contains all the questions, answer codes and output in sequence. Name this document as "A1-[StudentID].docx".
   c. All the submissions will be made on Google Classroom.
2. Each output should have STUDENT ID and NAME of the student at the top.
3. It should be clear that your assignment would not get any credit if the assignment is submitted after the due date.
4. Zero grade for plagiarism (copy/ cheating) and late submissions.

Task One:

You are developing a cybersecurity framework that has many layers of protection. The framework includes a SecurityTool representing a generic cybersecurity tool and a FirewallTool for firewall-specific features.

Class SecurityTool:

The class SecurityTool has the following features:

- securityLevel: to represent the security level of the tool.
- cost: represents the cost of the security tool.
- no of devices: the number of devices that the tool can run simultaneously on.

Implement the following functions within the SecurityTool class:

- A parameterized constructor that sets the attributes based on the user input.
  1. The security level can only be “High”, “Medium” or “Low”.
  2. The cost of the security tool can never be 0 or less than 0.
- performScan(): a function that prints a message indicating a generic security scan.

Class FirewallTool:

The class FirewallTool has the following features:

- Ports: a list of ports from which network traffic is allowed.
- Protocols: a list of protocols that are allowed by the firewall.

Implement the following functionality within the FirewallTool class:

- A parameterized constructor that invokes the base class constructor and sets the attributes based on the user input. A firewall cam simultaneously run on only 10 devices.
- generateList( ) is generated by the following way: Take any digit from your studentID except for 0. For example if you have taken 1 then the next 23 numbers starting from 2 till 24 are your allowed port numbers.
- ProtocolList only allows traffic from HTTPS, FTP, UDP, ICMP, SSH and SNMP.
- performScan(): the function carries out the scan in the following way:
  1. If the security level is set to High then only traffic from the port list and protocol list will be allowed.
  2. If the security level is set to Medium then allow all traffic from port and protocol list along with the next two ports in sequence(for example 25 and 26).
  3. If the security level is set to Low then allow all traffic from port and protocol list along with the next 5 ports in sequence(for example 25 - 30) and from TCP and DNS protocol.

In your main function perform the scan based on the conditions.

Task Two:

You are tasked with creating an inheritance hierarchy for a gaming environment. The environment consists of different aspects of the game.

Class Player:

- Attributes: playerID (int), playerName (string), health (int)
- Parameterized constructor that sets the attributes playerID, playerName. Health is initially initialized to 100 for the players.

Class Weapon:

- Attributes: weaponsList(contains a list of weapons)
- Constructor: Initialize the weapons list. The list should at least contain 5 or more weapons
- use( ): the function asks the user which weapon they want to use from the available list of weapons.

Class Character:

- Attributes: level (int), experience (string), points (int)
- Constructor: Parameterized constructor to set all attributes. Initial level and points are always set to 0 and experience is always set to Beginner.
- Function: levelUp(), increments the level and experience. The level and experience is incremented whenever the points are incremented by 10.
  The following conditions are applied for experience:
  1. If the experience is “Beginner” change the experience to “Intermediate”.
  2. If the experience is “Intermediate” change the experience to “Advanced”.
  3. If the experience is “Advanced” change the experience to “Expert”.
- Function: playGame() – The Character can play game by using any weapon to attack the enemy. When a character attacks an enemy, the enemy’s health decrements by 5 and 10 are added to the points.

Class Enemy:

- Attributes: damage (int).
- Constructor: Parameterized constructor to set damage. Damage can be set from a value ranging from 1 to 10.
- Function: void attack(), asks the users which weapon they want to use. When an enemy attacks a character, the character’s health decrements by the damage amount.

In your main function, simulate the gaming environment and by showing all the experience starting from “Beginner” to “Expert”.

Task Three:

Daraz Loyalty Program System

In this scenario, Daraz is launching a loyalty program for its customers.

Design a class named DarazPersonData with the following member variables:

- lastName (string)
- firstName (string)
- address (string)
- city (string)
- state (string)
- zip (string)
- phone (string)
- Write the appropriate accessor and mutator functions for these member variables.

Next, design a class named DarazCustomerData. The DarazCustomerData class should have the following member variables:

- customerNumber (an int)
- loyaltyPoints (an int)

The customerNumber variable will hold a unique integer for each customer. The loyaltyPoints variable will track the loyalty points earned by the customer. Write appropriate accessor and mutator functions for these member variables.

Design a class named DarazLoyaltyProgram to manage the loyalty program:

- Include functions to add loyalty points for purchases, redeem loyalty points for discounts, and display the total loyalty points for a customer.

Demonstrate the classes in a program by creating objects and performing operations such as adding loyalty points for purchases, redeeming loyalty points for discounts, and displaying total loyalty points for a customer.

Input Validation: Do not accept negative values for loyalty points or invalid customer numbers.

Task Four:

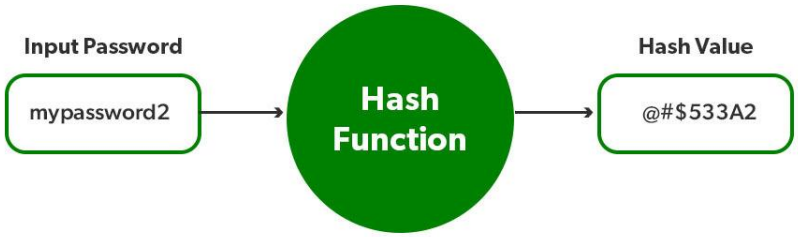
You've been tasked with designing the core components of a social media app similar to Instagram. The app will allow users to create profiles, post content, interact with posts (e.g., liking, commenting), and view their feed. There are different types of users, each with specific functionalities and access levels.

Tasks:

User Class Design:

- Design a base class User to represent common attributes and functionalities shared by all users, including username, email, and password.
- Implement user verification and password encryption to enhance security.

[Choose a suitable encryption algorithm (e.g., bcrypt, Argon2) for securely hashing passwords.]



Derived User Classes:

- Create derived classes for different types of users: RegularUser, and BusinessUser.
- Each derived class should inherit from the User class and provide specialized behavior based on the user's role and access level.
  - RegularUser Class:
    - Limited Posting: Regular users can post a maximum of 5 posts. Implement logic to enforce this limit.
    - Interactions: Regular users can like posts, comment on posts, and view their feed.
      - The RegularUser class maintains an array feed to store pointers to Post objects.
      - The addToFeed() method adds a post to the feed if there is space available.
      - The viewFeed() method displays the posts in the feed by iterating over the array and calling the display() method of each Post object.
      - Note: max feed size is 10; static const int MAX_FEED_SIZE = 10;
  - BusinessUser Class:
    - Post Promotion: Business users can promote their posts to reach a larger audience. Implement a method to promote posts.
      - User Validation: Ensure that only BusinessUser objects can invoke the promotePost() method.

        - Promotion Limit: Apply a limit on the number of posts a business user can promote. [let's say 10 posts only]
        - Post Visibility: A custom logic within the promotePost() method to increase the post's likes by double and views by thrice.
    - Analytics Integration: Enhance the User and Post classes to include the following analytics functionalities:
        - Likes Tracking: Implement methods to track and retrieve the number of likes for each post.
        - Comments Tracking: Implement methods to track and retrieve the number of comments for each post.
        - Views Tracking: Implement methods to track and retrieve the number of views for each post.

<u>Post Class Design:</u>

- Define a class Post to represent individual posts in the app. Consider properties like postId, content, likes, comments, etc., and methods for adding comments, liking posts, and displaying post details.

Interaction Simulation:

- Simulate interactions within the app by creating instances of different types of users and posts. Demonstrate how users can post content, interact with posts (e.g., liking, commenting), and view their feed. Use polymorphism to ensure that the same methods can be used uniformly across different user types.