Object-Oriented Programming Concepts Used

1. **Inheritance**:
    - The RegularUser and AdminUser classes inherit from the abstract User class. This allows them to reuse attributes and methods such as getUsername() and setPassword(), while also implementing or overriding abstract methods like getMenuOptions().
2. **Polymorphism**:
    - Polymorphism is evident in the getMenuOptions() method, where the behavior changes depending on the specific subclass (RegularUser or AdminUser). The createUser method in UserHandler also demonstrates polymorphism, as it returns objects of type User but the actual instance can be either AdminUser or RegularUser.
3. **Encapsulation**:
    - The User class uses private fields (firstName, lastName, username, etc.) with public getters and setters, controlling how these fields are accessed or modified. This protects the data and ensures proper validation or computation when required.
4. **Abstraction**:
    - The User class is abstract, meaning it cannot be instantiated directly and serves as a blueprint for specific types of users (RegularUser and AdminUser). It defines common functionality and enforces implementation of specific methods (getMenuOptions) in derived classes.
5. **Interfaces**:
    - The IHashCreator interface defines a contract for hash creation. This allows for flexibility and adherence to the Dependency Inversion Principle, as the areValidCredentials method in UserHandler can accept any implementation of IHashCreator.
6. **Composition**:
    - The UserHandler class depends on IHashCreator implementations (MD5HashCreator and SHA1HashCreator) for hashing functionality. This relationship is a composition since UserHandler doesn't create the hashers itself but uses external implementations.

Class Relationships and Diagram Explanation

1. **User and its Subclasses**:
    - **Relationship**: Generalization

    - AdminUser and RegularUser inherit from User, showcasing an "is-a" relationship.
2. **UserHandler and User**:
    - **Relationship**: Association
    - The UserHandler class works with User objects by creating and validating them.
3. **UserHandler and IHashCreator**:
    - **Relationship**: Dependency
    - The UserHandler class depends on the IHashCreator interface for hashing passwords, allowing for flexible hash method implementation.
4. **IHashCreator and Its Implementations**:
    - **Relationship**: Realization
    - MD5HashCreator and SHA1HashCreator implement the IHashCreator interface, ensuring consistent behavior for hash generation.

Class Diagram Structure

- **Abstract Class**: User
    - **Attributes**: firstName, lastName, username, password, status, type
    - **Methods**: getFirstName(), setFirstName(), getMenuOptions()
- **Subclasses**: RegularUser, AdminUser
    - **Inherits**: User
    - **Overrides**: getMenuOptions()
- **Interface**: IHashCreator
    - **Methods**: getHashFromString(input: String): String
- **Implementations**: MD5HashCreator, SHA1HashCreator
- **Controller**: UserHandler
    - **Uses**: User, IHashCreator