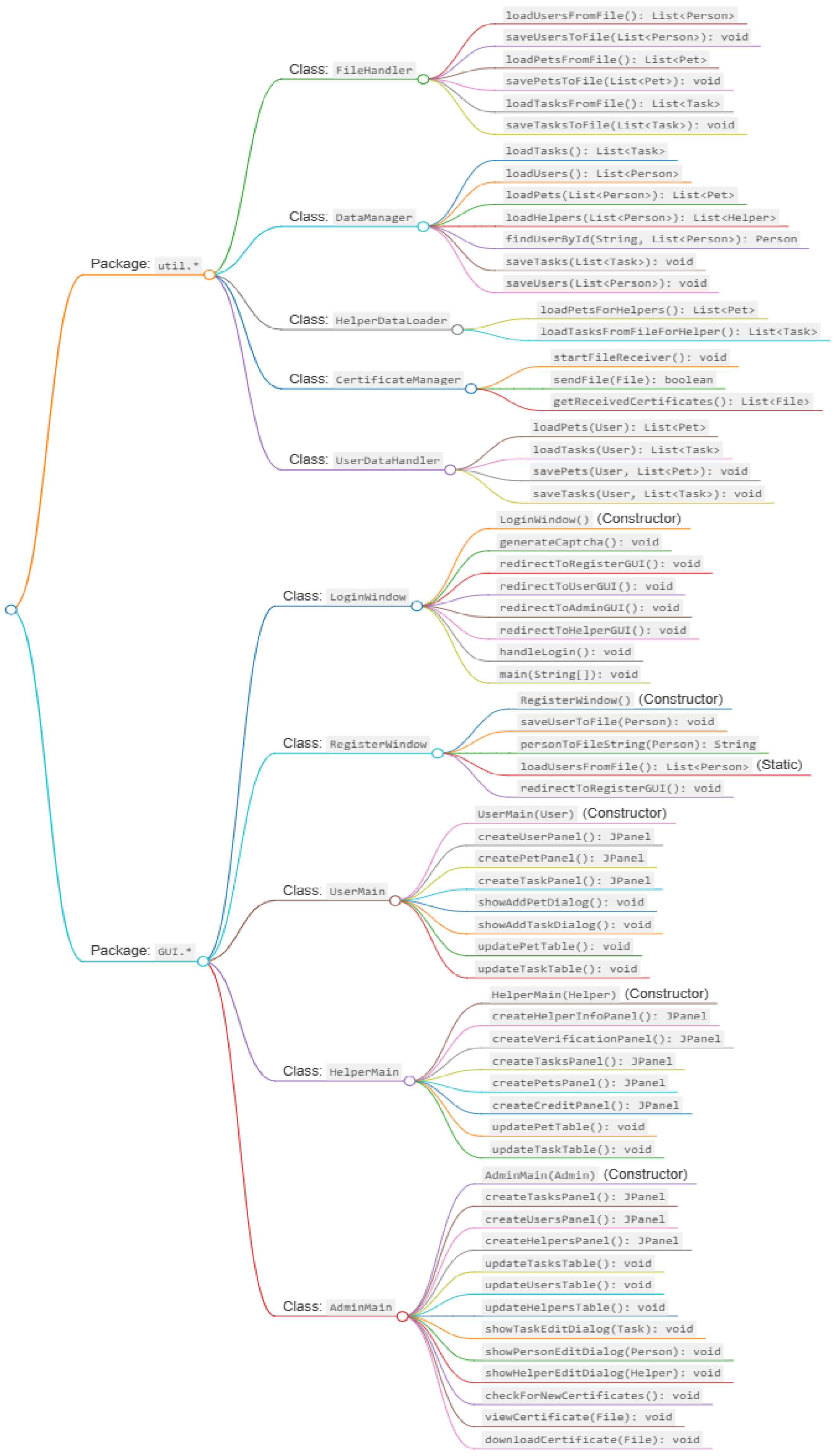
Structure of the Program

```
Person
 ├──── User
 ├──── Helper
 └──── Admin

Pet
 ├──── Dog
 ├──── Cat
 └──── OtherPet
```


Package: util.*
Class: FileHandler
loadUsersFromFile(): List<Person>
saveUsersToFile(List<Person>): void
loadPetsFromFile(): List<Pet>
savePetsToFile(List<Pet>): void
loadTasksFromFile(): List<Task>
saveTasksToFile(List<Task>): void
Class: DataManager
loadTasks(): List<Task>
loadUsers(): List<Person>
loadPets(List<Person>): List<Pet>
loadHelpers(List<Person>): List<Helper>
findUserById(String, List<Person>): Person
saveTasks(List<Task>): void
saveUsers(List<Person>): void
Class: HelperDataLoader
loadPetsForHelpers(): List<Pet>
loadTasksFromFileForHelper(): List<Task>
Class: CertificateManager
startFileReceiver(): void
sendFile(File): boolean
getReceivedCertificates(): List<File>
Class: UserDataHandler
loadPets(User): List<Pet>
loadTasks(User): List<Task>
savePets(User, List<Pet>): void
saveTasks(User, List<Task>): void
Package: GUI.*
Class: LoginWindow
LoginWindow() (Constructor)
generateCaptcha(): void
redirectToRegisterGUI(): void
redirectToUserGUI(): void
redirectToAdminGUI(): void
redirectToHelperGUI(): void
handleLogin(): void
main(String[]): void
Class: RegisterWindow
RegisterWindow() (Constructor)
saveUserToFile(Person): void
personToFileString(Person): String
loadUsersFromFile(): List<Person> (Static)
redirectToRegisterGUI(): void
Class: UserMain
UserMain(User) (Constructor)
createUserPanel(): JPanel
createPetPanel(): JPanel
createTaskPanel(): JPanel
showAddPetDialog(): void
showAddTaskDialog(): void
updatePetTable(): void
updateTaskTable(): void
Class: HelperMain
HelperMain(Helper) (Constructor)
createHelperInfoPanel(): JPanel
createVerificationPanel(): JPanel
createTasksPanel(): JPanel
createPetsPanel(): JPanel
createCreditPanel(): JPanel
updatePetTable(): void
updateTaskTable(): void
Class: AdminMain
AdminMain(Admin) (Constructor)
createTasksPanel(): JPanel
createUsersPanel(): JPanel
createHelpersPanel(): JPanel
updateTasksTable(): void
updateUsersTable(): void
updateHelpersTable(): void
showTaskEditDialog(Task): void
showPersonEditDialog(Person): void
showHelperEditDialog(Helper): void
checkForNewCertificates(): void
viewCertificate(File): void
downloadCertificate(File): void

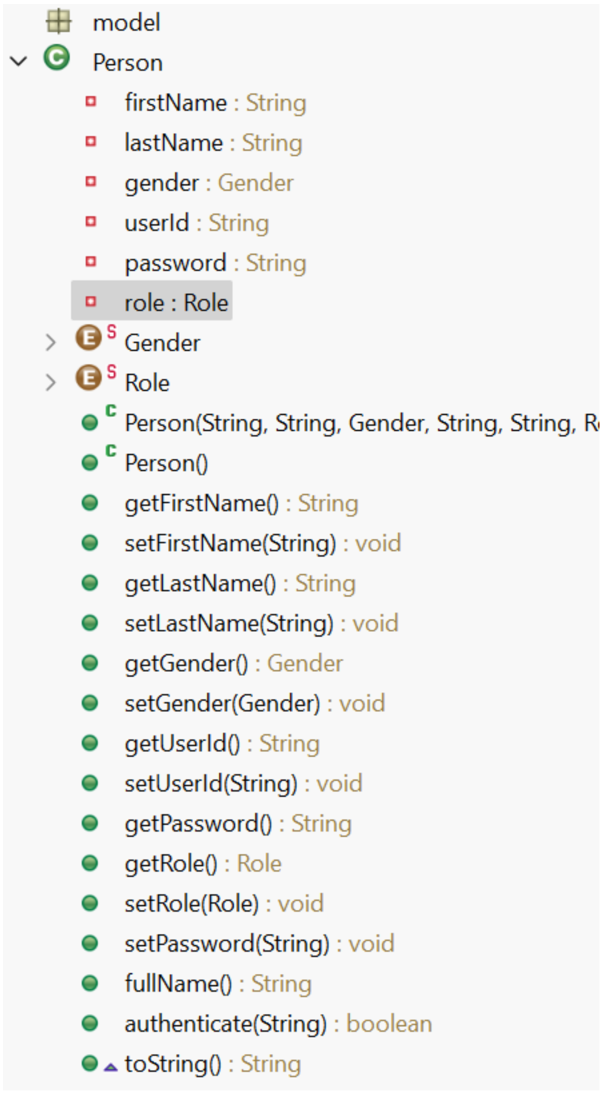
Purpose of Program:

This system is built to connect pet owners (Users) with helpers (Helpers) who volunteer or get paid to take care of pets. Admins oversee and manage everything, ensuring smooth operation and trustworthiness through certificates and verification.

How It Works

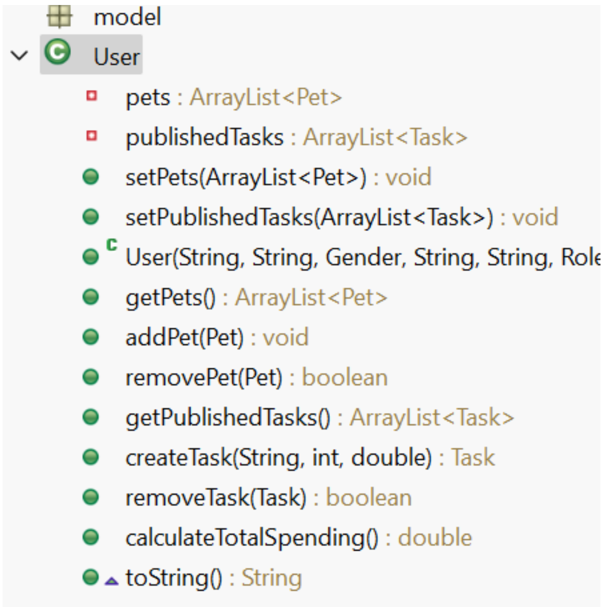
- **Model Layer:**
  Defines the core entities — people (Users, Helpers, Admins), pets (Dogs, Cats, Others), and tasks (pet care jobs). Users add pets and create tasks, Helpers take on tasks and build credit, Admins control the whole shebang. Each class has tailored methods to handle its responsibilities.
- **Utils Layer:**
  Handles data persistence — loading and saving users, pets, tasks, and helpers from files. Also manages file transfers for certificates, keeping things legit and verified.
- **GUI Layer:**
  The front line where users log in (with CAPTCHA to stop bots), register, and interact. Users manage their pets and tasks, Helpers verify themselves and accept tasks, Admins get a full dashboard to manage users, pets, tasks, and certificates.

Package model.*;



```
model
Person
    firstName : String
    lastName : String
    gender : Gender
    userId : String
    password : String
    role : Role
  Gender
  Role
    Person(String, String, Gender, String, String, R
    Person()
    getFirstName() : String
    setFirstName(String) : void
    getLastName() : String
    setLastName(String) : void
    getGender() : Gender
    setGender(Gender) : void
    getUserId() : String
    setUserId(String) : void
    getPassword() : String
    getRole() : Role
    setRole(Role) : void
    setPassword(String) : void
    fullName() : String
    authenticate(String) : boolean
    toString() : String
```

1. **Constructors**:
    - Person(String firstName, String lastName, Gender gender, String userId, String password, Role role): Initializes a Person object with the provided details.
    - Person(): Default constructor that initializes a Person with empty/default values.
2. **Getters**:
3. **Setters**:
4. **Methods**:
    - fullName(): Combines and returns the first and last name as a full name string.
    - authenticate(String inputPassword): Checks if the input password matches the person's password.
5. **Override Method**:
    - toString(): Returns a string representation of the Person object, including first name, last name, gender, and user ID.
    -



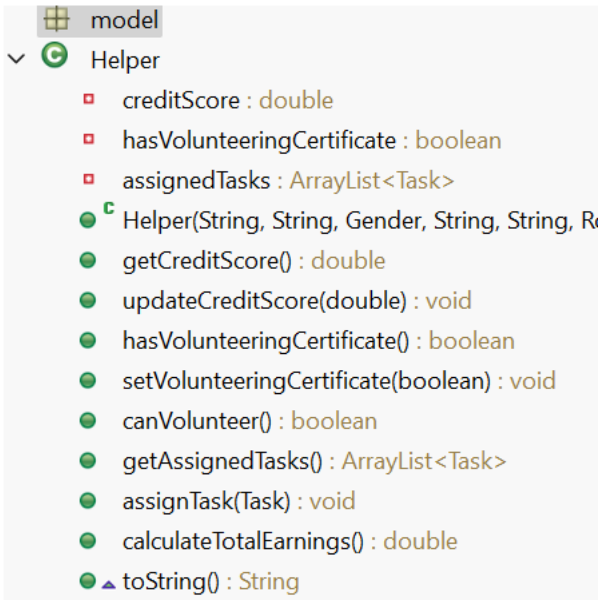
```
model
User
    pets : ArrayList<Pet>
    publishedTasks : ArrayList<Task>
    setPets(ArrayList<Pet>) : void
    setPublishedTasks(ArrayList<Task>) : void
    User(String, String, Gender, String, String, Role
    getPets() : ArrayList<Pet>
    addPet(Pet) : void
    removePet(Pet) : boolean
    getPublishedTasks() : ArrayList<Task>
    createTask(String, int, double) : Task
    removeTask(Task) : boolean
    calculateTotalSpending() : double
    toString() : String
```

1. **Constructor**:
    - User(...): Initializes a User object with the provided details and initializes empty lists for pets and tasks.
2. **Pet Management**:
    - addPet(Pet pet): Adds a pet to the user's list of pets.
    - removePet(Pet pet): Removes a pet from the user's list of pets.
3. **Task Management**:
    - createTask(...): Creates a new task, associates it with the user, and adds it to the list of published tasks.

- removeTask(Task task): Removes a task from the user's list of published tasks.
- calculateTotalSpending(): Calculates the total spending on tasks that have been taken, based on working hours and hourly wage.

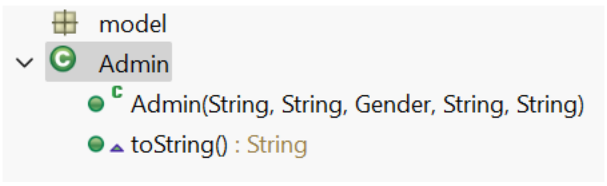
4. **Getters, Setters, and** toString.

5.



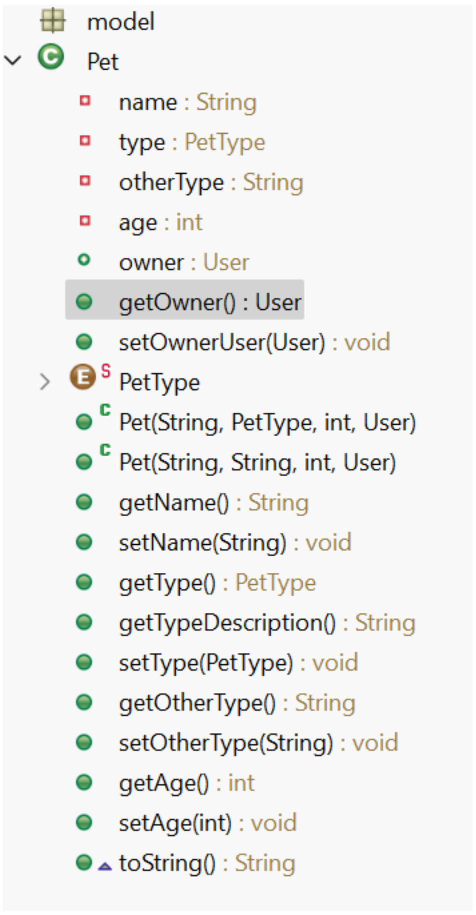
```
model
Helper
  creditScore : double
  hasVolunteeringCertificate : boolean
  assignedTasks : ArrayList<Task>
  Helper(String, String, Gender, String, String, R
  getCreditScore() : double
  updateCreditScore(double) : void
  hasVolunteeringCertificate() : boolean
  setVolunteeringCertificate(boolean) : void
  canVolunteer() : boolean
  getAssignedTasks() : ArrayList<Task>
  assignTask(Task) : void
  calculateTotalEarnings() : double
  toString() : String
```

1. **Constructor**:
   - Helper(...): Initializes a Helper with default values (credit score: 5.0, no volunteering certificate, empty task list).
2. **Credit Score Management**:
   - updateCreditScore(double change): Adjusts the credit score, clamping it between 0 and 10.
3. **Volunteering Status**:
   - canVolunteer(): Returns true if the helper has a volunteering certificate.
4. **Task Management**:
   - assignTask(Task task): Assigns a task to the helper if they are certified; throws an exception otherwise.
   - calculateTotalEarnings(): Sums up earnings from all assigned tasks.
5. **Getters, Setters, and** toString:



```
model
Admin
  Admin(String, String, Gender, String, String)
  toString() : String
```

1. **Constructor**:
   - Admin(...): Initializes an Admin with provided details and sets the role to ADMIN by default.
2. **Getters, Setters, and** toString:



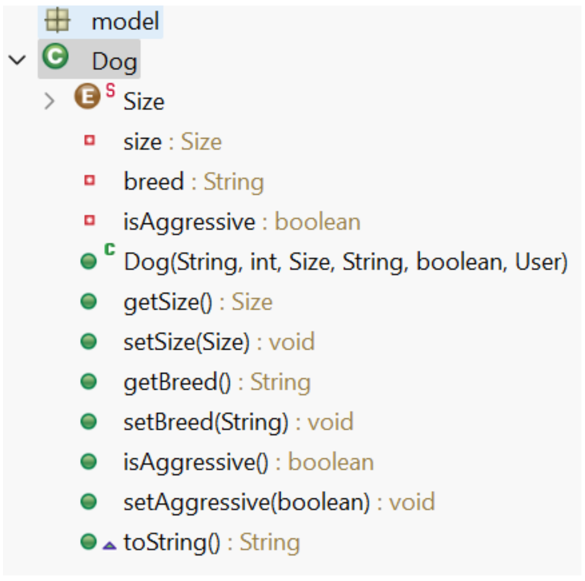
```
model
Pet
  name : String
  type : PetType
  otherType : String
  age : int
  owner : User
  getOwner() : User
  setOwnerUser(User) : void
  PetType
  Pet(String, PetType, int, User)
  Pet(String, String, int, User)
  getName() : String
  setName(String) : void
  getType() : PetType
  getTypeDescription() : String
  setType(PetType) : void
  getOtherType() : String
  setOtherType(String) : void
  getAge() : int
  setAge(int) : void
  toString() : String
```

1. **Constructors**:
   - Pet(String name, PetType type, int age, User owner): Initializes a pet

for DOG or CAT types (throws an exception for OTHERS).

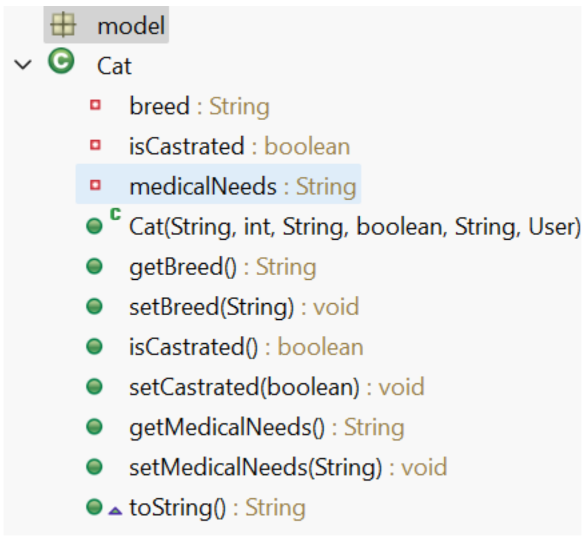
- Pet(String name, String otherType, int age, User owner): Initializes a pet for OTHERS type, requiring a custom type description.

2. **Special Methods**:
   - getTypeDescription(): Returns a formatted string for the pet type (includes custom description for OTHERS).
3. **Getters, Setters, and** toString:



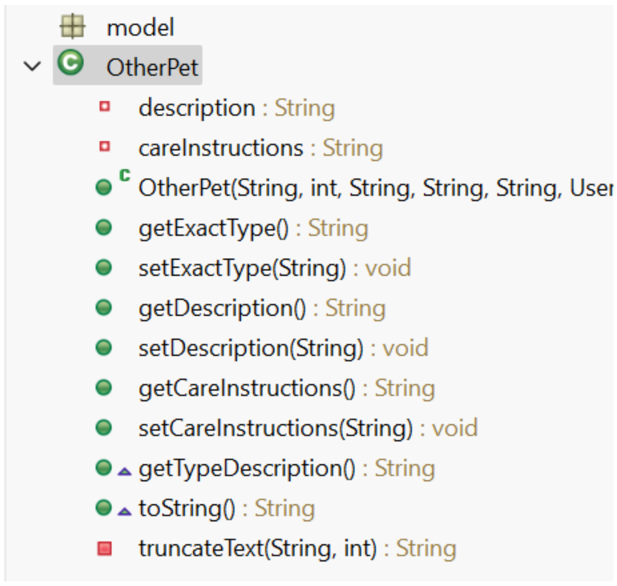
```
model
Dog
  Size
  size : Size
  breed : String
  isAggressive : boolean
  Dog(String, int, Size, String, boolean, User)
  getSize() : Size
  setSize(Size) : void
  getBreed() : String
  setBreed(String) : void
  isAggressive() : boolean
  setAggressive(boolean) : void
  toString() : String
```

1. **Constructor**:
   - Dog(...): Initializes a Dog with specific attributes (name, age, size, breed, aggression status, and owner), setting the pet type to DOG.
2. **Getters, Setters, and** toString:



```
model
Cat
  breed : String
  isCastrated : boolean
  medicalNeeds : String
  Cat(String, int, String, boolean, String, User)
  getBreed() : String
  setBreed(String) : void
  isCastrated() : boolean
  setCastrated(boolean) : void
  getMedicalNeeds() : String
  setMedicalNeeds(String) : void
  toString() : String
```

1. **Constructor**:
   - Cat(...): Initializes a Cat with specific attributes (name, age, breed, castration status, medical needs, and owner), setting the pet type to CAT.
2. **Getters, Setters, and** toString:
3.



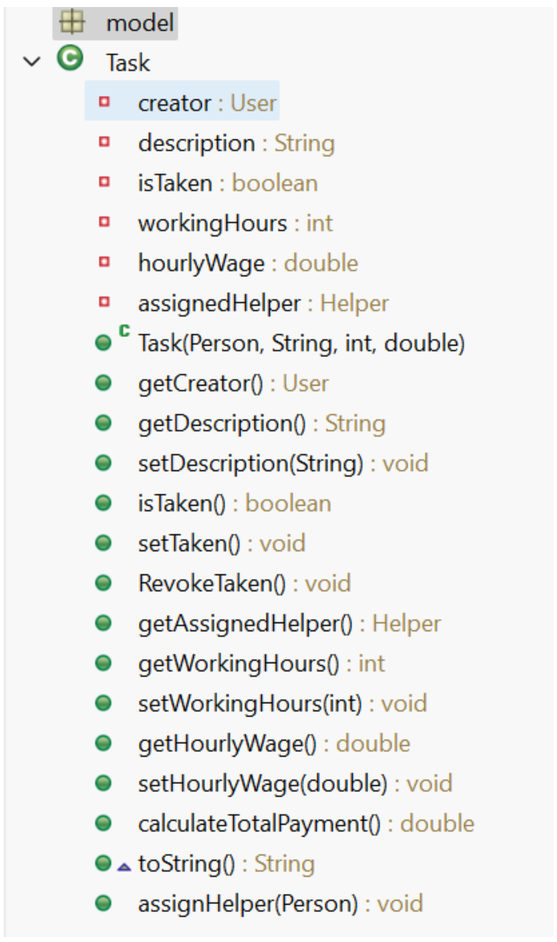
```
model
OtherPet
  description : String
  careInstructions : String
  OtherPet(String, int, String, String, String, Use
  getExactType() : String
  setExactType(String) : void
  getDescription() : String
  setDescription(String) : void
  getCareInstructions() : String
  setCareInstructions(String) : void
  getTypeDescription() : String
  toString() : String
  truncateText(String, int) : String
```

1. **Constructor**:
   - OtherPet(...): Initializes an OtherPet with specific attributes (name, age, exact type, description, care instructions, and owner), using the OTHERS pet type.
2. **Special Methods**:
   - getTypeDescription(): Overrides the

parent method to add an "Exotic: " prefix to the type description.

  - truncateText(String text, int maxLength): Helper method to shorten long text for display purposes.

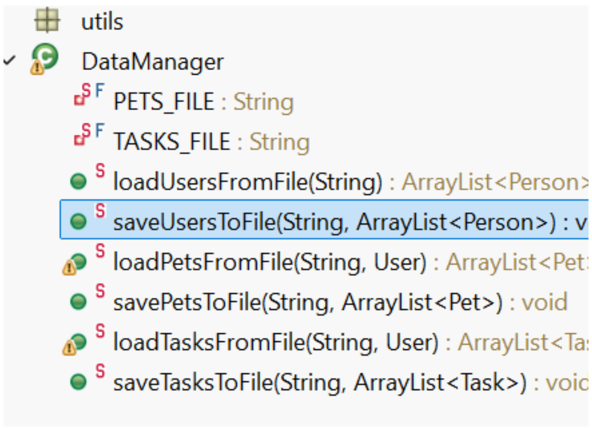
3. **Getters, Setters, and** toString



```
model
Task
  creator : User
  description : String
  isTaken : boolean
  workingHours : int
  hourlyWage : double
  assignedHelper : Helper
  Task(Person, String, int, double)
  getCreator() : User
  getDescription() : String
  setDescription(String) : void
  isTaken() : boolean
  setTaken() : void
  RevokeTaken() : void
  getAssignedHelper() : Helper
  getWorkingHours() : int
  setWorkingHours(int) : void
  getHourlyWage() : double
  setHourlyWage(double) : void
  calculateTotalPayment() : double
  toString() : String
  assignHelper(Person) : void
```

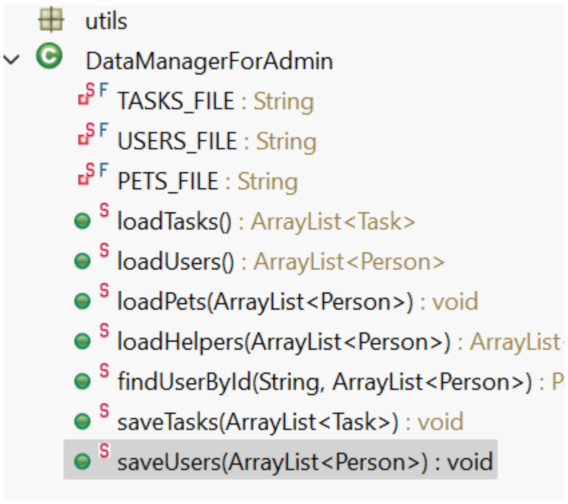
1. **Constructor**:
   - Task(...): Initializes a task with creator, description, working hours, and hourly wage. Sets default values (isTaken = false, assignedHelper = null).
2. **Task Status Management**:
   - setTaken(): Marks the task as taken.
   - RevokeTaken(): Reverts the task to "not taken" status.
3. **Helper Assignment**:
   - assignHelper(Person helper): Assigns a helper to the task (casts to Helper type).
4. **Financial Calculation**:
   - calculateTotalPayment(): Computes total payment based on working hours and hourly wage.
5. **Getters, Setters, and** toString:

Package util.*;



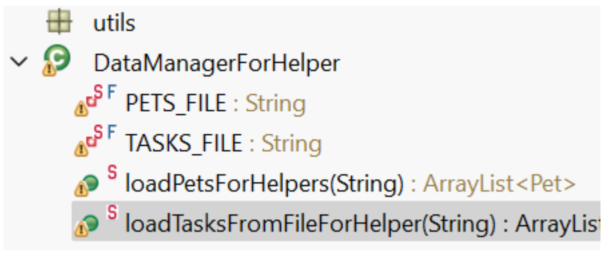
```
utils
DataManager
    PETS_FILE : String
    TASKS_FILE : String
    loadUsersFromFile(String) : ArrayList<Person>
    saveUsersToFile(String, ArrayList<Person>) : v
    loadPetsFromFile(String, User) : ArrayList<Pet
    savePetsToFile(String, ArrayList<Pet>) : void
    loadTasksFromFile(String, User) : ArrayList<Ta
    saveTasksToFile(String, ArrayList<Task>) : voic
```

1. **User Management**:
   - loadUsersFromFile(): Loads user data from a CSV file, creates appropriate Person subclasses (Admin/Helper/User), and handles validation.
   - saveUsersToFile(): Saves user data to a CSV file with headers, including role-specific fields like credit scores for Helpers.
2. **Pet Management**:
   - loadPetsFromFile(): Loads pet data from a CSV file, creating specific pet types (Dog/Cat/OtherPet) and associating them with owners.
   - savePetsToFile(): Saves pet data to a CSV file with type-specific formatting for different pet classes.
3. **Task Management**:
   - loadTasksFromFile(): Loads task data from a CSV file, handling quoted descriptions and numeric fields.
   - saveTasksToFile(): Saves task data to a CSV file with standard formatting.



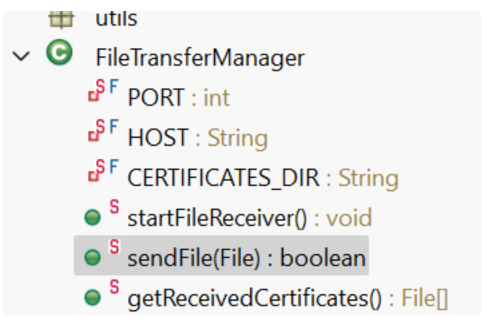
```
utils
DataManagerForAdmin
    TASKS_FILE : String
    USERS_FILE : String
    PETS_FILE : String
    loadTasks() : ArrayList<Task>
    loadUsers() : ArrayList<Person>
    loadPets(ArrayList<Person>) : void
    loadHelpers(ArrayList<Person>) : ArrayList
    findUserById(String, ArrayList<Person>) : P
    saveTasks(ArrayList<Task>) : void
    saveUsers(ArrayList<Person>) : void
```

1. **Data Loading Functions**:
   - loadTasks(): Loads all tasks from file, creating Task objects without owner association
   - loadUsers(): Loads all users from file, creating appropriate Person subclasses (Admin/Helper/User)
   - loadPets(): Loads pets from file and associates them with their owners in the provided user list
   - loadHelpers(): Filters and returns only Helper objects from the user list
2. **Utility Functions**:
   - findUserById(): Searches for a user by ID in the provided list
3. **Data Saving Functions**:
   - saveTasks(): Saves all tasks to file with proper formatting
   - saveUsers(): Saves all user data to file including role-specific fields
   -

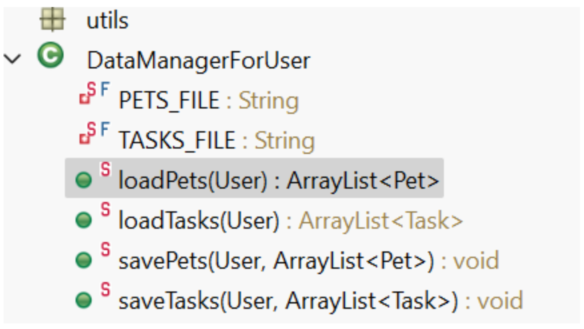


```
utils
DataManagerForHelper
    PETS_FILE : String
    TASKS_FILE : String
    loadPetsForHelpers(String) : ArrayList<Pet>
    loadTasksFromFileForHelper(String) : ArrayLis
```

1. **Pet Data Loading**:
   - loadPetsForHelpers(): Loads all pets from file without owner filtering (helpers can view all pets), creating appropriate pet subclass objects (Dog/Cat/OtherPet) with null owners.
2. **Task Data Loading**:
   - loadTasksFromFileForHelper(): Loads all tasks from file without owner association (helpers can view all tasks), creating Task objects with null creators.


utils
FileTransferManager
PORT : int
HOST : String
CERTIFICATES_DIR : String
startFileReceiver() : void
sendFile(File) : boolean
getReceivedCertificates() : File[]

1. **File Receiving**:
   - startFileReceiver(): Starts a server thread that continuously listens for incoming PDF certificate files and saves them to a designated directory.
2. **File Sending**:
   - sendFile(): Client-side function that sends a PDF file to the server after validation, returning success/failure status.
3. **Certificate Management**:
   - getReceivedCertificates(): Retrieves all received PDF certificates from the storage directory for admin review.


utils
DataManagerForUser
PETS_FILE : String
TASKS_FILE : String
loadPets(User) : ArrayList<Pet>
loadTasks(User) : ArrayList<Task>
savePets(User, ArrayList<Pet>) : void
saveTasks(User, ArrayList<Task>) : void

loadPets(User owner)

- Loads only the pets belonging to the specified user from pets.txt
- Returns an ArrayList of Pet objects filtered by owner ID

loadTasks(User owner)

- Loads only the tasks created by the specified user from tasks.txt
- Returns an ArrayList of Task objects filtered by creator ID

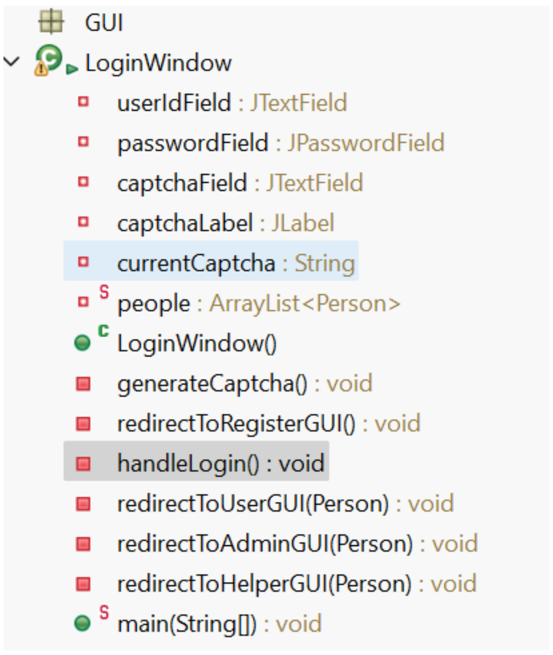
savePets(User owner, ArrayList<Pet> userPets)

- Saves the user's pets while preserving other users' pet data
- Updates pets.txt with merged data (current user's pets + all others)

saveTasks(User owner, ArrayList<Task> userTasks)

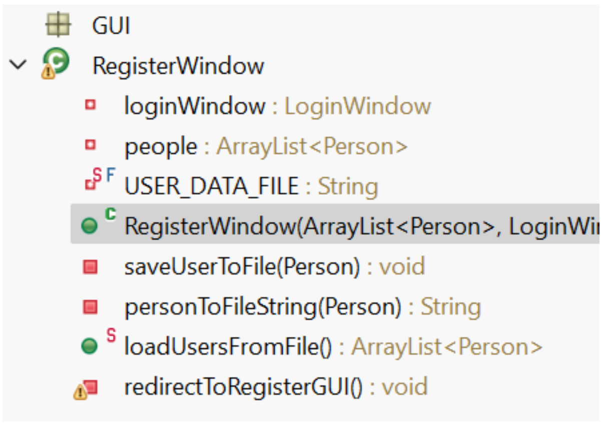
- Saves the user's tasks while preserving other users' task data
- Updates tasks.txt with merged data (current user's tasks + all others)

Package GUI.*;



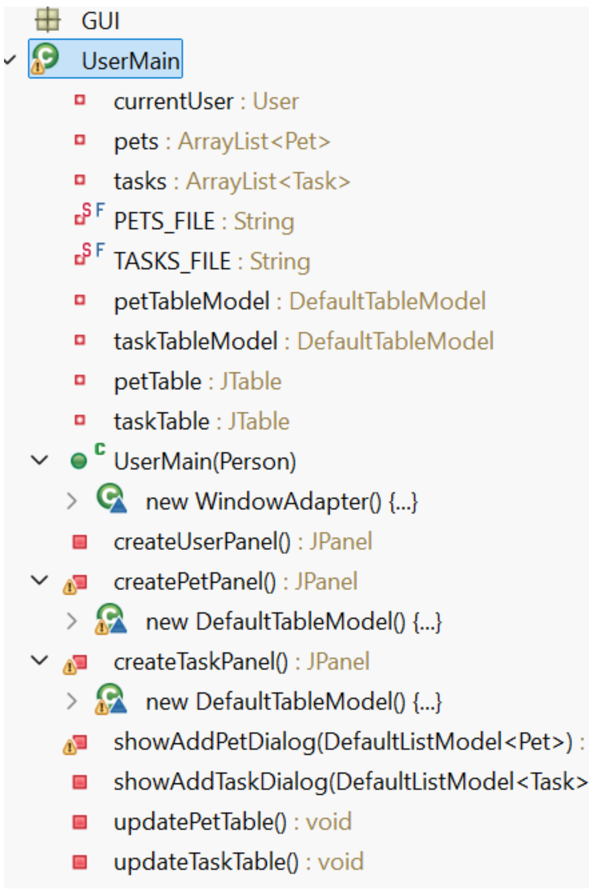
```
GUI
LoginWindow
    userIdField : JTextField
    passwordField : JPasswordField
    captchaField : JTextField
    captchaLabel : JLabel
    currentCaptcha : String
    S people : ArrayList<Person>
    C LoginWindow()
    generateCaptcha() : void
    redirectToRegisterGUI() : void
    handleLogin() : void
    redirectToUserGUI(Person) : void
    redirectToAdminGUI(Person) : void
    redirectToHelperGUI(Person) : void
    S main(String[]) : void
```

1. **Constructor**:
    - LoginWindow(): Sets up the login GUI with user ID, password fields, CAPTCHA verification, and login/register buttons.
2. **CAPTCHA Handling**:
    - generateCaptcha(): Creates a random 6-character CAPTCHA code and displays it.
3. **Navigation Methods**:
    - redirectToRegisterGUI(): Opens the registration window and closes login.
    - redirectToUserGUI(), redirectToAdmin GUI(), redirectToHelperGUI(): Redirects to appropriate dashboards based on user role.
4. **Core Logic**:
    - handleLogin(): Validates credentials, verifies CAPTCHA, authenticates user, and redirects based on role.
5. **Main Method**:
    - main(): Launches the login window using Swing's event dispatch thread.



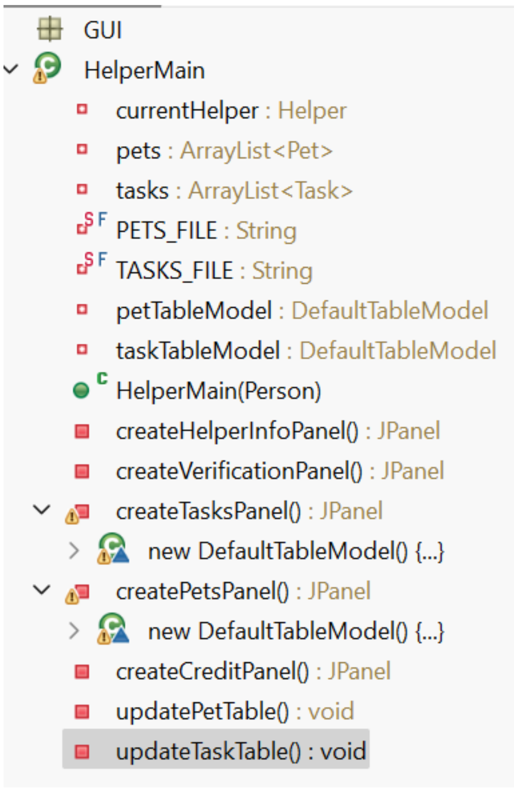
```
GUI
RegisterWindow
    loginWindow : LoginWindow
    people : ArrayList<Person>
    SF USER_DATA_FILE : String
    C RegisterWindow(ArrayList<Person>, LoginWi
    saveUserToFile(Person) : void
    personToFileString(Person) : String
    S loadUsersFromFile() : ArrayList<Person>
    redirectToRegisterGUI() : void
```

1. **Constructor**:
    - RegisterWindow(): Sets up the registration form with fields for user details and a registration button.
2. **Data Handling**:
    - saveUserToFile(): Appends new user data to the users file.
    - personToFileString(): Formats user data into CSV format for file storage.
3. **User Management**:
    - loadUsersFromFile(): Static method that loads all registered users from file, creating appropriate Person subclasses (Admin/Helper/User).
    - redirectToRegisterGUI(): Reopens the registration window (though appears unused in current context).



```
GUI
UserMain
    currentUser : User
    pets : ArrayList<Pet>
    tasks : ArrayList<Task>
    PETS_FILE : String
    TASKS_FILE : String
    petTableModel : DefaultTableModel
    taskTableModel : DefaultTableModel
    petTable : JTable
    taskTable : JTable
    UserMain(Person)
        new WindowAdapter() {...}
    createUserPanel() : JPanel
    createPetPanel() : JPanel
        new DefaultTableModel() {...}
    createTaskPanel() : JPanel
        new DefaultTableModel() {...}
    showAddPetDialog(DefaultListModel<Pet>) :
    showAddTaskDialog(DefaultListModel<Task>
    updatePetTable() : void
    updateTaskTable() : void
```

1. **Constructor**:
    - UserMain(): Initializes the user dashboard with pet and task management interfaces.
2. **UI Creation**:
    - createUserPanel(): Displays user profile information in a formatted box.
    - createPetPanel(): Sets up the pet management table with add/delete functionality.
    - createTaskPanel(): Configures the task management table with add/delete features.
3. **Dialog Handlers**:
    - showAddPetDialog(): Displays a dynamic form for adding different pet types (Dog/Cat/Other).
    - showAddTaskDialog(): Shows a form for creating new tasks.
4. **Data Management**:
    - updatePetTable(): Refreshes the pet table with current data.
    - updateTaskTable(): Updates the task table with latest information.

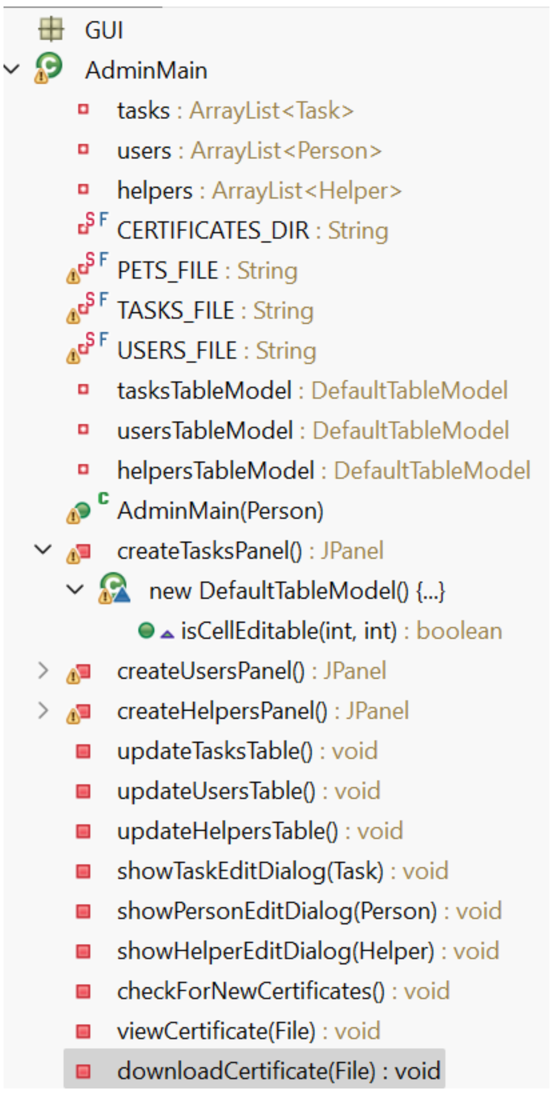


```
GUI
HelperMain
    currentHelper : Helper
    pets : ArrayList<Pet>
    tasks : ArrayList<Task>
    PETS_FILE : String
    TASKS_FILE : String
    petTableModel : DefaultTableModel
    taskTableModel : DefaultTableModel
    HelperMain(Person)
    createHelperInfoPanel() : JPanel
    createVerificationPanel() : JPanel
    createTasksPanel() : JPanel
        new DefaultTableModel() {...}
    createPetsPanel() : JPanel
        new DefaultTableModel() {...}
    createCreditPanel() : JPanel
    updatePetTable() : void
    updateTaskTable() : void
```

1. **Constructor**:
    - HelperMain(): Initializes the helper dashboard interface with verification status, task management, and pet viewing capabilities.
2. **UI Creation**:
    - createHelperInfoPanel(): Displays helper profile information including credit score.
    - createVerificationPanel(): Shows verification status with certificate upload functionality.
    - createTasksPanel(): Sets up available tasks table with task acceptance feature.

- createPetsPanel(): Configures pets viewing table.
- createCreditPanel(): Displays credit rating as star icons with total earnings.

3. **Data Management**:
    - updatePetTable(): Refreshes the pet table with current data.
    - updateTaskTable(): Updates the task table showing only available tasks.



```
GUI
AdminMain
    tasks : ArrayList<Task>
    users : ArrayList<Person>
    helpers : ArrayList<Helper>
    CERTIFICATES_DIR : String
    PETS_FILE : String
    TASKS_FILE : String
    USERS_FILE : String
    tasksTableModel : DefaultTableModel
    usersTableModel : DefaultTableModel
    helpersTableModel : DefaultTableModel
    AdminMain(Person)
    createTasksPanel() : JPanel
        new DefaultTableModel() {...}
            isCellEditable(int, int) : boolean
    createUsersPanel() : JPanel
    createHelpersPanel() : JPanel
    updateTasksTable() : void
    updateUsersTable() : void
    updateHelpersTable() : void
    showTaskEditDialog(Task) : void
    showPersonEditDialog(Person) : void
    showHelperEditDialog(Helper) : void
    checkForNewCertificates() : void
    viewCertificate(File) : void
    downloadCertificate(File) : void
```

1. **Constructor**:
    - AdminMain(): Initializes the admin dashboard with three management panels (tasks, users, helpers) and certificate handling.

2. **UI Creation**:
    - createTasksPanel(): Sets up task management table with edit/delete/refresh functionality.
    - createUsersPanel(): Configures user management table with edit/delete/refresh options.
    - createHelpersPanel(): Creates helper management table with additional certificate download feature.

3. **Table Updates**:
    - updateTasksTable(): Refreshes task data in the table.
    - updateUsersTable(): Updates user data display.
    - updateHelpersTable(): Refreshes helper information including certification status.

4. **Edit Dialogs**:
    - showTaskEditDialog(): Displays form for editing task details.
    - showPersonEditDialog(): Shows form for modifying user information.
    - showHelperEditDialog(): Provides interface for editing helper-specific fields like credit score.

5. **Certificate Management**:
    - checkForNewCertificates(): Checks and displays newly received certificates.
    - viewCertificate(): Opens certificate file for viewing.
    - downloadCertificate(): Saves certificate to chosen location.