

# Sirius Software

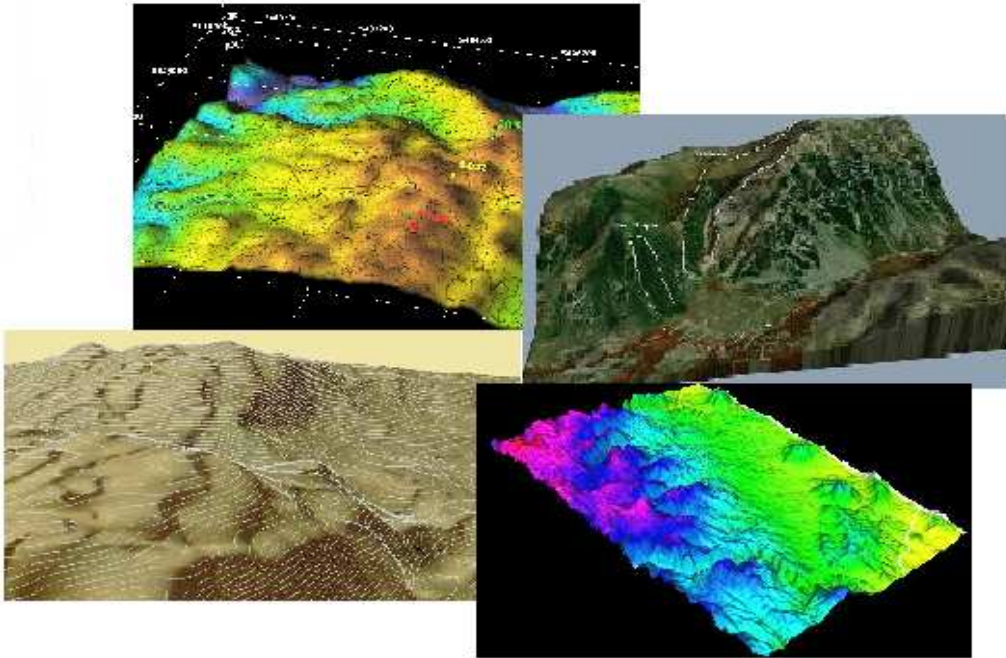
---

Duygu Yapa 1449222

Ayşe Turan 1449164

Duygu Altınok 1502095

Elif Kamer Karataş 1448836



## Climb Planner

---

[Initial Design Report]

---

16.12.2008

## Contents

1.	.....	
	INTRODUCTION .....	4
1.1.	Purpose of the Document .....	4
1.2.	Project Definition.....	4
1.3.	Project Scope .....	4
2.	DESIGN CONSIDERATIONS.....	5
2.1.	Constraints and Limitations .....	5
2.1.1.	Time .....	5
2.1.2.	Performance .....	5
2.2.	Design Goals and Objectives.....	6
2.2.1.	Portability .....	6
2.2.2.	Usability .....	6
2.2.3.	Reliability .....	6
2.2.4.	Approach and Modeling.....	6
3.	FILE AND FOLDER FORMATS.....	7
3.1.	Map Files .....	7
3.2.	System Files and Folders.....	7
3.3.	Image and Video Files.....	9
4.	LIBRARIES AND TOOLS.....	11
4.1.	Languages and Platforms .....	11
4.1.1.	Java.....	11
4.1.2.	Eclipse .....	11
4.1.3.	NetBeans .....	12
4.2.	Geographical Information Systems.....	12
4.2.1.	GeoTools .....	12
4.3.	Graphics Libraries .....	12
4.4.	Other Tools and Libraries .....	13
4.4.1.	XML.....	13
4.4.2.	MySql .....	13
5.	USER INTERFACE DESIGN .....	14
5.1.	Main Window .....	15
5.2.	File Menu .....	16
5.3.	Terrain Menu.....	16

5.4.	Weather Menu .....	19
5.5.	Help Menu .....	19
5.6.	Member Information Panel .....	19
5.7.	Constraints Panel .....	21
5.8.	3D Visualization of the Mountain Panel.....	22
5.9.	Get Activity Plan Window .....	23
6.	ARCHITECTURAL DESIGN .....	24
6.1.	Overall Architecture .....	24
6.2.	Package and Class Design.....	25
6.2.1.	GUIPackage .....	25
6.2.2.	GISPackage .....	30
6.2.3.	MemberPackage.....	31
6.2.4.	EquipmentPackage .....	32
6.2.5.	PlannerPackage .....	33
6.3.	Behavioral Design.....	35
6.4.	Functional Modeling .....	36
6.4.1.	Data Flow Diagrams .....	36
6.4.2.	Data Dictionary .....	41
6.5.	Database Design.....	49
6.5.1.	Tables .....	49
6.5.2.	ER Diagrams .....	50
7.	IMPLEMENTATION .....	52
7.1.	Current State .....	52
7.2.	Prototype Implementation .....	52
7.3.	Future Work .....	52
8.	CONCLUSION .....	53
9.	APPENDIX.....	54
10.	REFERENCES .....	55

# 1. INTRODUCTION

Climb Planner Software Project is developing by Sirius Software to ODTU DKSK. This project's mission is providing efficient and well-organized climbing experiences to mountaineers. Climb Planner will correspond to mountaineers' needs about planning an activity.

## 1.1. Purpose of the Document

This report is written to show initial design of climb planner software by Climb Planner project team. The headings and parts of this document are divided appropriately by grouping relevant subjects of whole system. Project group has tried to give the definition and design of the software system by dividing subsystems and showing the relations with each other. In addition, the other necessary issues like implementation requirements, specifications are handled.

## 1.2. Project Definition

The software basically finds a route on a terrain for mountaineers. It will provide the best activity plan for reaching a selected mountain peak by taking into account the specified constraints. In addition, it makes a visualization of the terrain and a simulation of the route. Software uses some kind of information from mountaineers, database and website to make all of these facilities.

## 1.3. Project Scope

The scope of the project has been described in the Requirement Analysis Report. Here, a brief of the scope will be given.

The software should be used by instructor mountaineer, because only instructor can know about the capabilities of the mountaineers in a club or decide to the date of an activity

This guy has some responsibilities and abilities on the software;

- He should give the climbers' information in the club to the system any time and this information will be kept in system until an instructor delete or update them.
- Before running system for getting an activity plan, he should give terrain information to the system by giving a “.dted” or “.dem” format map, the region (ex; Ankara, Turkey) and optionally vector map and satellite image.
- He can give some constraints like safest route, time and distance constraints and checkpoints.

After giving a raster map, program will show a visualization of the terrain. Then he can run the program. The program gives an activity plan (optionally in a report) which includes climbing route, estimated duration of climbing, food and equipment list, emergency equipment list and camping plan. He can see a simulation of the route on the terrain, if he wants. The additional features will be described in the rest of this document.

## 2. DESIGN CONSIDERATIONS

### 2.1. Constraints and Limitations

#### 2.1.1. Time

The project should be completed until July 2009. In addition a demo of this project should be prepared until the end of January 2009. The detailed project schedule is given as a Gantt chart in Appendix.

#### 2.1.2. Performance

Performance is an important issue and makes us anxious, because finding an optimum way in a terrain with lots of constraints which have different priorities is a hard problem. Hence we will try different solutions for this problem to reach the best performance. In addition, the visualization and simulation part we will use efficient ways.

## 2.2. Design Goals and Objectives

### 2.2.1. Portability

We have thought about this issue and decided to use portable languages and tools. In other words, this was a constraint to choose libraries and toolkits. So, we will try to make software portable on Windows and Linux platforms.

### 2.2.2. Usability

We have tried to design a user friendly software and correspond to mountaineers needs. In this context, we have made interviews with professional mountaineers and obtained their ideas about planning a climb. We have examined how much we make this software useful for mountaineers. At the end, we canceled some unnecessary and absurd parts from the project. For example, we canceled rafting or skiing experience of climbers because a climber never carries rafting or skiing equipments when he goes to climbing even if he has these kinds of experiences. We also added some extra features that make software better for users like visualization of the terrain.

### 2.2.3. Reliability

As we mentioned before, we have made our studies in wide range. So, we have thought almost all aspects of the climbing issue. We have taken up references from the mountaineers Hasan Hüseyin BOĞAZ and Tunç FINDIK and the book of “Zirvelerin Özgürlüğü”. Besides we have searched lots of libraries, tools and algorithms for the implementation part and selected the most appropriate ones for our project.

### 2.2.4. Approach and Modeling

Because of the complexity of our project, we have decided to use Object Oriented Approach. This approach models our system so makes the design more understandable and the Object Oriented Design elaborates the models to produce implementation specifications. For representing these models, we use Unified Modeling Language (UML) that has a number of different notations for representing models.

## 3. FILE AND FOLDER FORMATS

In this project, we use some kind of file formats and folder arrangement. We will mention about these formats under map files, image and video files and system files and folders headings.

### 3.1. Map Files

We will use map files for getting terrain information in a format from user. We have thought the consistency of the file formats with our GIS tool, when searching map files. The “.dted” and “.dem” raster map formats are suggested to us for elevation information of the terrain by Aselsan. We have searched on them and we saw that they are usable and readable for our GIS tool. As satellite image we also think about the easy access from internet. Hence we have found the “.geotiff” format. On the other hand, as vector file, we have decided to use shape file format (.shp) because it contains the necessary information that we want.

### 3.2. System Files and Folders

The software will use some file and folder format specifications. We have designed a file format which contains all existing information of a project when it is saved. We will save the project file with a “.srs” format because we want to use an original file format and this file will have the project’s information in XML format. Because XML is a useful format and Java has ready functions for reading, writing or parsing etc. for this file format.

The Climb Planner’ sample workspace view is below.

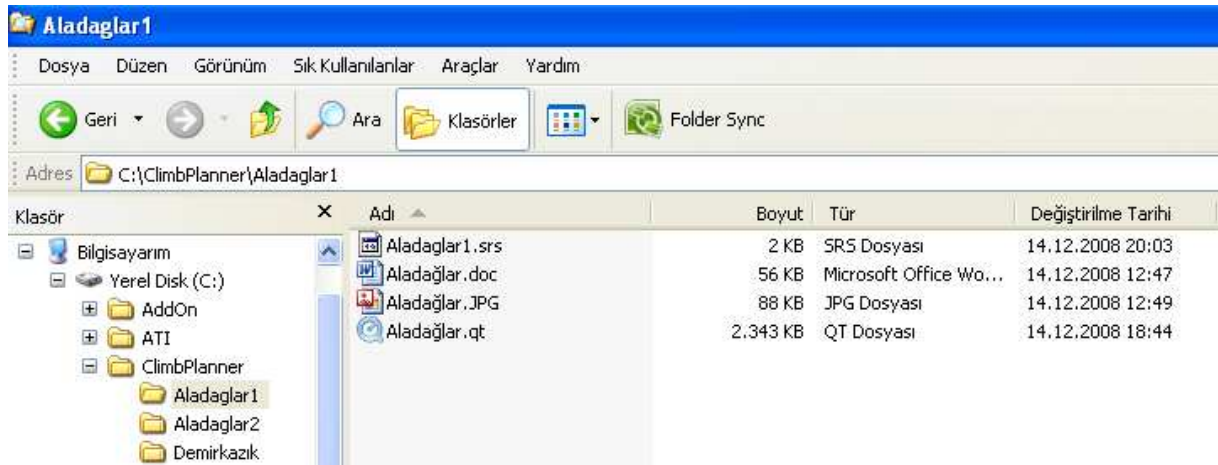


Figure 3.1. ClimbPlanner workspace

As we mention above the “.srs” file contain the project current state information. It is in XML format. The Aladaglar1.srs file’s content is like that;

```
<?xml version="1.0" encoding="UTF-8" ?>
= <projectDescription>

  <projectName>Aladaglar1</projectName>
  <projectPath>C:\ClimbPlanner\Aladaglar1</projectPath>
=  <climbers>
    <id>1449222</id>
    <id>1449164</id>
    <id>1502095</id>
    <id>1448836</id>
  </climbers>
=  <projectMaps>
    <rasterMaps>C:\ClimbPlanner\aladag.dem</rasterMaps>
    <vectorMaps>C:\ClimbPlanner\aladag.shp</vectorMaps>
  </projectMaps>
=  <climbRegion>
    <country>Turkey</country>
    <city>Kayseri</city>
  </climbRegion>
=  <weather>
    <lowTemp>2</lowTemp>
    <highTemp>20</highTemp>
    <pressure>1024.7</pressure>
    <windSpeed>4</windSpeed>
    <windDirection>2</windDirection>
    <season>2</season>
```



```

    <rain>false</rain>
</weather>

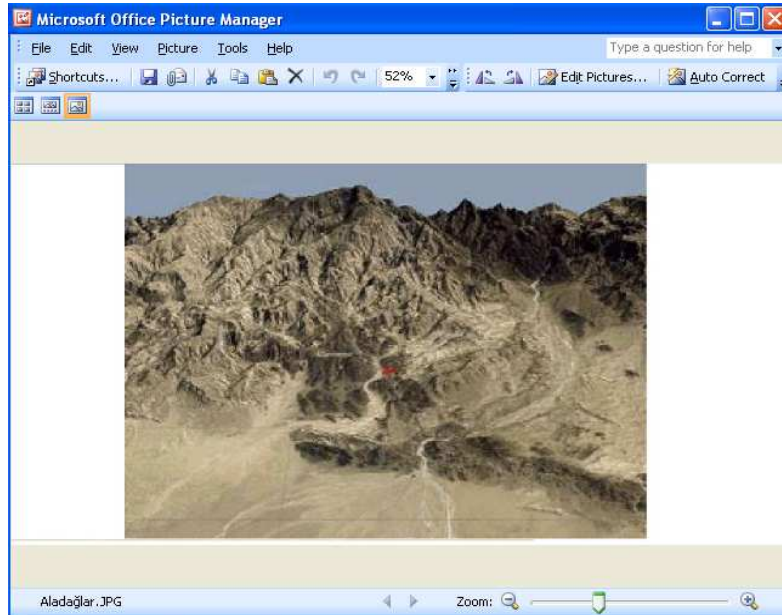
= <projectConstraints>
    <number>1</number>
    <number>4</number>
</projectConstraints>
= <climbDate>
    <day>20</day>
    <mounth>6</mounth>
    <year>2009</year>
</climbDate>
<!-- Start Coordinate -->
<coordinate>
    <longitude />
    <latitude />
    <elevation />
</coordinate>
<!-- End Coordinate -->
= <coordinate>
    <longitude />
    <latitude />
    <elevation />
</coordinate>
<!-- Check Points (if exist) -->
= <avalanche>
    <longitude />
    <latitude />
    <elevation />
</avalanche>
= <fallingRock>
    <longitude />
    <latitude />
    <elevation />
</fallingRock>
</projectDescription>

```

### 3.3. Image and Video Files

We will use the image and video files for saving the visualization and simulation of the terrain. For image file we will use a generic file format “.jpeg”. We think that this file format will be consistent with the systems of users. The user can save the visualization of terrain as an image, which is a screenshot of the visualization, anytime.

The “.jpeg” file of this image is like that;



*Figure 3.2. Visualization of the terrain*

In the simulation part, user can save the simulation as a “.qt” file that can be opened with QuickTime multimedia framework. A screenshot of this video is below. In this video, camera will follow the red colored route which is planned by Climb Planner from the start coordinate to the end coordinate.



*Figure 3.3. Simulation of the route*

## 4. LIBRARIES AND TOOLS

### 4.1. Languages and Platforms

#### 4.1.1. Java

Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. The syntax of Java is largely derived from C++. Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object oriented language. All code is written inside a class and everything is an object, with the exception of the intrinsic data types (ordinal and real numbers, boolean values, and characters), which are not classes for performance reasons. [1] We preferred Java for its high-level object orientation, its rich IDEs, wide developer support and documentation, wide library choices; but the our main reason was that we wanted to use GeoTools which is a Java library. Also for our project's visualization & simulation parts, Java was the best choice for its available libraries.

#### 4.1.2. Eclipse

Eclipse is a software platform comprising extensible application frameworks, tools and a runtime library for software development and management. It is written primarily in Java to provide software developers and administrators an integrated development environment (IDE). In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its capabilities by installing plugins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs provide translations into over a dozen natural languages. It was easy for us to decide to use Eclipse as it's the most powerful Java IDE; it has dozens of plugins to support our work, it has automatic code generation as well as easy usage such as showing and explaining runtime, compile-time and

code mistakes. Also Eclipse is easy to use with “svn” as well as it provides the Eclipse Workbench - views, editors, perspectives, wizards; Equinox OSGi - a standard bundling framework, Core platform - boot Eclipse, run plug-ins, the Standard Widget Toolkit (SWT) - a portable widget toolkit, JFace - viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors etc.[2]

#### **4.1.3. NetBeans**

A free, open-source Integrated Development Environment for software developers. The NetBeans IDE provides several new features and enhancements, such as rich PHP, JavaScript and Ajax editing features, improved support for using the Hibernate web framework and the Java Persistence API, and tighter GlassFish v3 and MySQL integration. [3] We used NetBeans for designing our GUI interfaces; as we use Java by far NetBeans is the best choice for interface design for its easy usage and developer-friendly.

### **4.2. Geographical Information Systems**

#### **4.2.1. GeoTools**

GeoTools is an open source (LGPL) Java code library which provides standards compliant methods for the manipulation of geospatial data, for example to implement Geographic Information Systems (GIS). [4] We decided to use GeoTools as we needed a GIS library and GeoTools provides all GIS capabilities. Moreover, as we use Java; GeoTools & Java combination would be powerful enough.

### **4.3. Graphics Libraries**

We decided to use GeoTools' graphic capabilities for our project's visualization and simulation parts, since we mostly use GeoTools for the other parts of our project.

## 4.4. Other Tools and Libraries

### 4.4.1. XML

Extensible Markup Language (XML) provides a foundation for creating documents and document systems. XML operates on two main levels: first, it provides syntax for document markup; and second, it provides syntax for declaring the structures of documents. XML's simplicity is its key selling point, perhaps even its strongest feature. Our team have plenty of reasons to use XML : simplicity , extensibility , interoperability. For the simplicity , XML's rigid set of rules helps make documents more readable to both humans and machines. XML document syntax contains a fairly small set of rules, making it possible for our team to get started right away. For the extensibility; XML is extensible in two senses. First, it allows developers to create their own DTDs, effectively creating 'extensible' tag sets that can be used for multiple applications. Second, XML itself is being extended with several additional standards that add styles, linking, and referencing ability to the core XML set of capabilities. XML complements Java, a force for interoperability, very well, and a considerable amount of early XML development has been in Java. A generic application programming interface (API) for parsers, the Simple API in XML (SAX), is freely available. [5]

### 4.4.2. MySQL

MySQL database is the world's most popular open source database because of its fast performance, high reliability, ease of use, and dramatic cost savings. Our team preferred MySQL for its easy usage and mostly for its Java compability. [6] Since MySQL is also a Sun Microsystems product ; not only Java has JDBC support for SQL , also the combination of Java & MySQL is a very powerful combination due to its high performance.

## 5. USER INTERFACE DESIGN

In the stage of designing a user interface, the most important point that should be paid attention is that the application should be easy to use and understand. Although the functionality that an application provides to users is important, the way in which it provides that functionality is just as important. An application that is difficult to use or understand won't be used.

For designing clear and easy to use user interface the following points should be considered:

- **Consistency** : The user interface should work consistently. Tools like buttons, lists etc. should be put in consistent places, and a consistent color scheme should be used.
- **Ease of navigation between windows**: If it is difficult to get from one screen to another, then users will quickly become frustrated and give up. When the flow between screens matches the flow of the work the user is trying to accomplish, then the application will make sense to users.
- **Effective labeling**: The displayed text on the screens is a primary source of information for users. If the text is worded poorly, then the user interface will be perceived poorly by users. Using full words and sentences, as opposed to abbreviations and codes, makes text easier to understand.
- **Effective handling of mistakes made by users**: The user interface should be designed to recover from mistakes made by users.
- **Avoiding unnecessary items**: Crowded screens are difficult to understand and, hence, are difficult to use.
- **Effective grouping**: Items that are logically connected should be grouped together on the screen to communicate they are connected, whereas items that have nothing to do with each other should be separated.

These are all known issues of designing user interfaces. Considering these, we have developed a prototype of our application just to show the main design we made in our mind. There may be some changes in the design in the next steps of the project. In the following sections, we will explain the main points of our graphical user interface.

## 5.1. Main Window

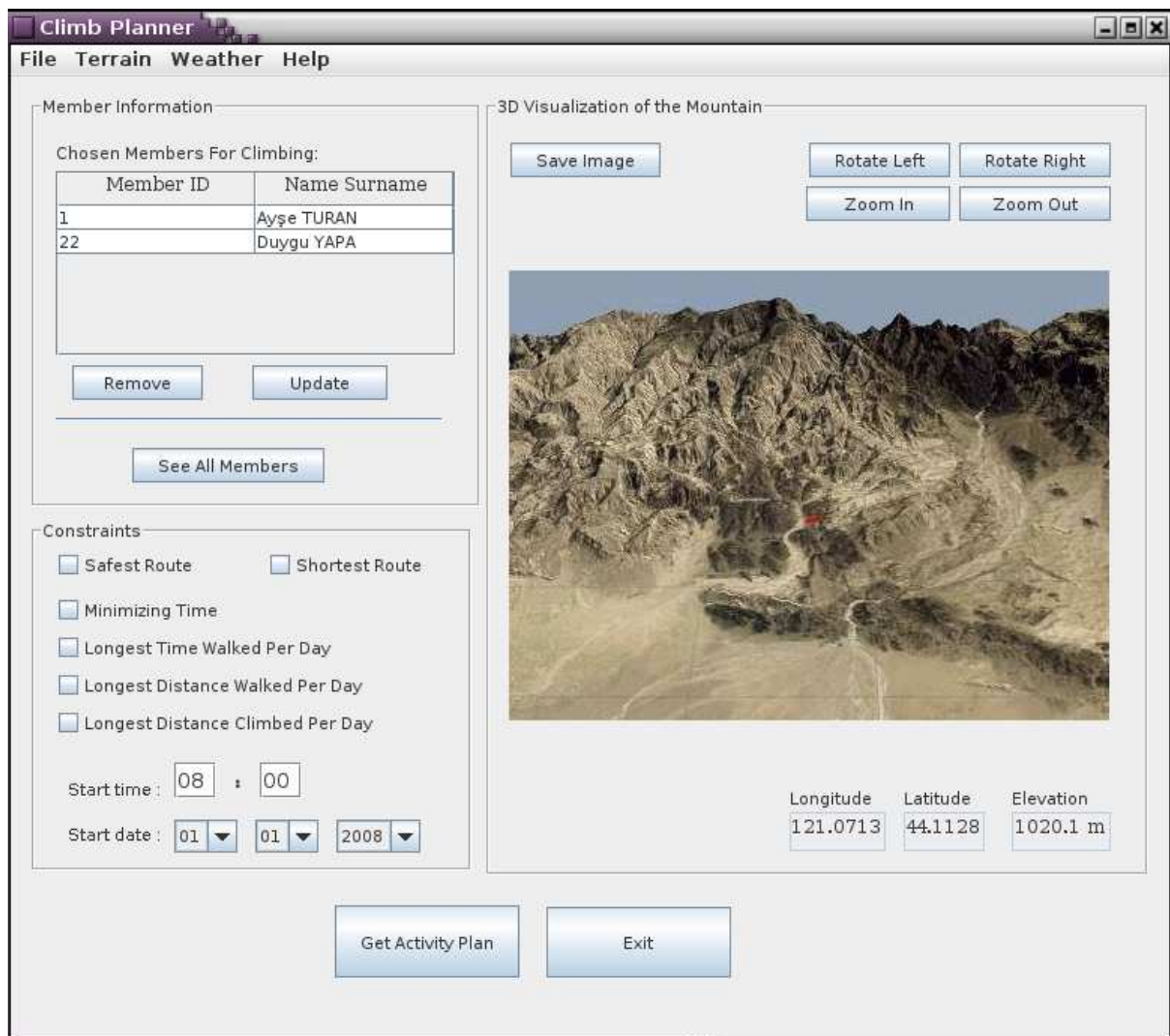


Figure 5.1. Main Window of ClimPlanner

In Figure 5.1, you see the main window of Clim Planner Software. It consists of a menubar including File, Terrain, Weather and Help menus, Member Information, Constraints and 3D Visualization of the Mountain Panels and Get Activity Plan and Exit buttons.

## 5.2. File Menu

From File Menu, the following options can be chosen:

- **New Project:** If New Project is clicked; after closing the old one, a new project will be open with default values (Default values are explained in the following sections).
- **Open Project:** If Open Project is clicked a file chooser for opening a “.srs” file will appear. Clicking the desired project with “.srs” extension, and “Open”, the project will be open.
- **Save Project:** If Save Project is clicked, a file chooser for saving the current project will appear and writing the name and clicking “Save” will save the project to the desired location.
- **Exit:** When Exit is clicked, if the current project has not saved recently, first an option pane that asks whether to save the project before exiting or not, will appear. According to chosen option, the Climb Planner will exit either with saving the project or without saving it.

## 5.3. Terrain Menu

From Terrain Menu, the following options can be chosen:

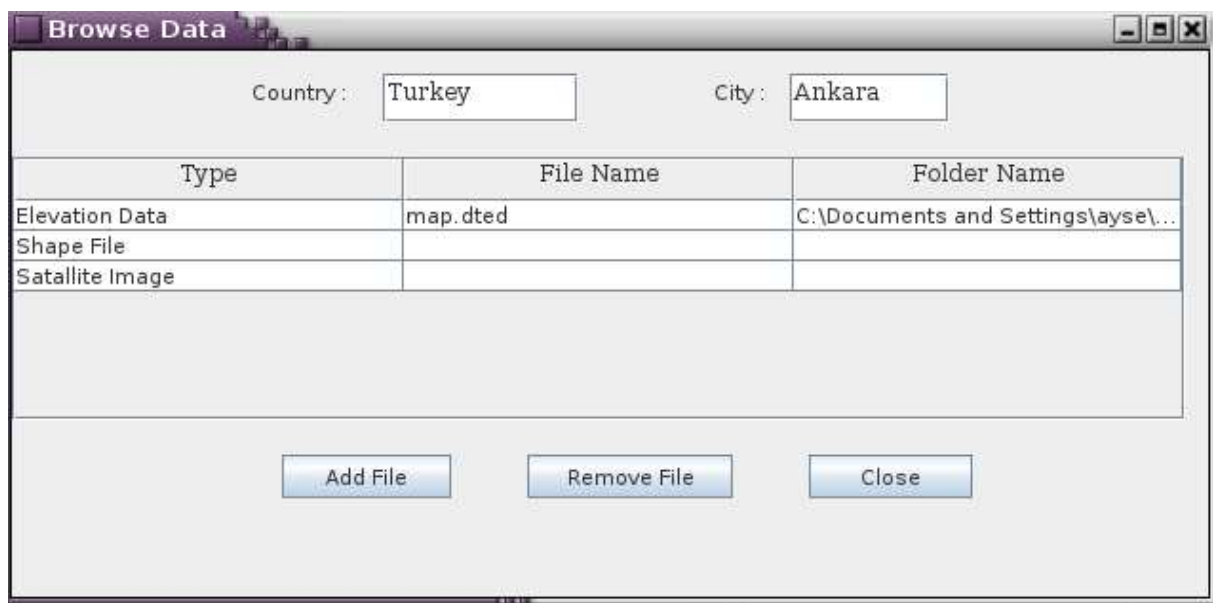
- **Browse Data:** If Browse Data is clicked, the window given in Figure 5.2 will appear.

From Browse Data window,

- User enters the Country and City of the mountain. After the feedback we have gotten from Aselsan, we have changed the taking weather information part a bit. In the new version, the weather information is not taken manually any more; instead, it is taken from a website that makes forecasting. And country and city information is put for that purpose. After user enters the country and city of the mountain, the weather information is taken from the website even



without informing the user. (But if user wants to see the weather information, he can see by clicking “See Weather Information” under Weather Menu). At the beginning of a new project country is “Turkey” and city is “Ankara” as default. And an elevation data will be given as default (for visualization purposes) and there is no shape file and satellite image.



*Figure 5.2. Browse Data Window*

- If one of the rows in table is chosen and then “Add File Button” is pressed, a file chooser will appear.

If the chosen row is

- Elevation Data, then user can give only elevation files with “.dted” or “.dem” extensions. The given elevation file is replaced by the old one and it is reflected to the Elevation Data row. There should be exactly one elevation data file.
- Shape File or Satellite Image then a similar procedure will apply with file extensions “.shp” and “.geotiff” respectively. Here, the shape file can represent water wells, rivers and lakes. Different from elevation data, there can be no shape file or satellite image. These files are not

necessary but the presence of them makes activity plan more accurate.

- If one of the rows in table is chosen and then “Remove File Button” is pressed, the chosen file is removed from the project. Since without an elevation data, it is impossible to get any output (activity plan, visualization, simulation etc.) removing an elevation data is impossible. The change in elevation data can only be made by “Add File Button”.
- If “Close Button” is pressed, the Browse Data Window is closed returning to the main window of Climb Planner. (Figure 5.1)

- **Add**

- **Map:** If adding map is chosen, then a file chooser that shows files with “.dted” and “.dem” extensions will appear. Choosing one of them and pressing “Add” will replace the elevation data file with the newly selected file.
- **Shape File:** If adding a shape file is chosen, then a file chooser that shows files with “.shp” extensions will appear. Choosing one of them and pressing “Add” will replace the lake file with the newly selected file if an old one exists. If not, the newly selected lake file is simply added to the project.
- **Satellite Image:** Same procedure as adding a shape file will apply with “.geotiff” extension.

- **Remove**

- **Shape File:** Selecting “Remove Shape File” will just remove the previously selected shape file from the project. If there is no shape file before, nothing will happen.
- **Satellite Image:** Selecting “Remove Satellite Image” will just remove the previously selected satellite image file from the project. If there is no satellite image before, nothing will happen.

Adding and Removing data from Terrain Menu is functionally the same as in the Browse Data Window.

## 5.4. Weather Menu

In Weather Menu there is only one option : “See Weather Information”. If this option is chosen, then the data gathered from the website using the given Country and City will be shown to the user.

## 5.5. Help Menu

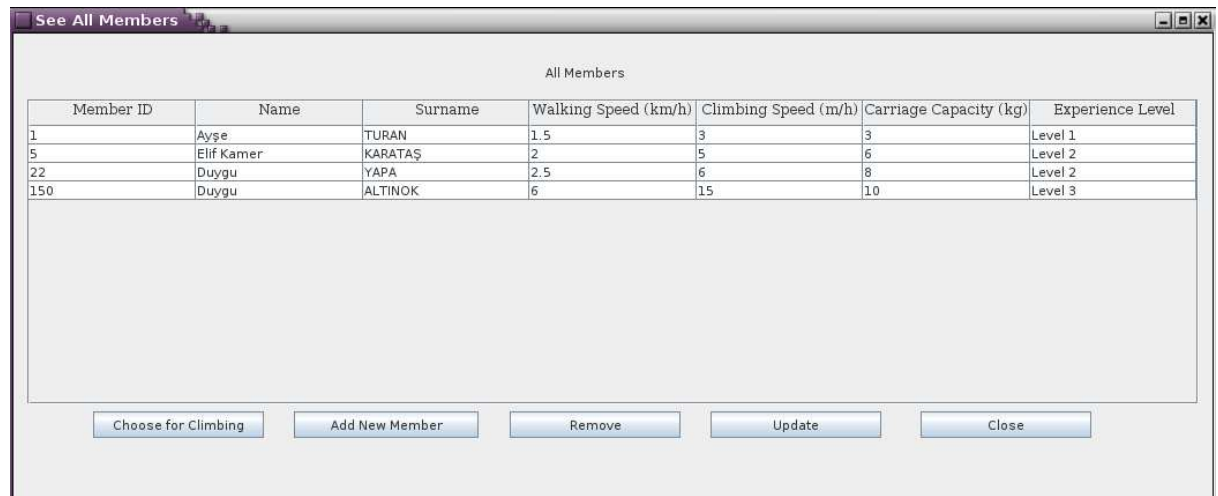
From Help Menu, the following Options can be chosen:

- **Help Topics:** If Help Topics option is chosen, then a window showing the user documentation for the user interface will appear.
- **About ClimbPlanner:** If this option is chosen, then a short description and the developers of ClimbPlanner will appear.

## 5.6. Member Information Panel

In Member Information Panel,

- There is a list of members (their ids and names) that will go climbing. If user chooses one of them and presses
  - “Remove Button”, then that member is removed from chosen members list.
  - “Update Button”, then a window (given in Figure 5.5) that will contain the information of the selected member is shown. Updating the necessary field and pressing “OK” will updates the information of the selected member.
- There is a button to see the information of all existing members (both chosen for climbing and not) If user presses this button, the window given in Figure 5.3 will appear.



*Figure 5.3. See All Members*

From See All Members Window, user can perform the following actions:

- **Choose a Member for Climbing:** If one of the member is selected and "Choose for Climbing Button" is pressed, the member is added to the chosen members list and is shown in the member information panel of the Main Window.
- **Add New Member:** If user presses "Add New Member Button", then the window given in Figure 5.4 will appear. Filling the informations, and pressing "OK" gives the member a unique id and adds member to the database (Member Table in Section 6.4) and See All Members window is updated accordingly.
- **Remove an Existing Member:** If user selects a member and presses "Remove Button", then the selected member is removed from the database permanently and See All Members Window is updated accordingly. If the removed user is in chosen members for climbing list, he will also be removed from that list.
- **Update Member Information:** If user selects a member and presses "Update Button", then the information of the selected member will appear. (Figure 5.5) Changing the necessary fields and pressing "OK", updates the information of the selected member.



**Add New Member**

New Member Information

Name :

Surname :

Walking Speed (km/h) :

Climbing Speed (m/h) :

Carriage Capacity (kg) :

Experince Level :

OK Cancel

Figure 5.4. Add New Member Window



**Update Member**

Selected Member Information

Name :

Surname :

Walking Speed (km/h) :

Climbing Speed (m/h) :

Carriage Capacity (kg) :

Experince Level :

OK Cancel

Figure 5.5. Update Member Window

- **Close Window:** If “Close” is pressed, See All Members Window will be closed returning back to the main window. (Figure 5.1)

## 5.7. Constraints Panel

In Constraints Panel, there are constraints that are taken into account when preparing the activity plan. User can choose more than one constraint. He also specifies the start date and time of the activity. Since weather is one of the major factors in preparing activity plan and weather forecast can be made up to fifteen days, the start date combo boxes will be updated for a 15 days period. The default start date will be the day that the Climb Planner is currently used. For example, if the software is used at 15.12.2008, then the start date can be chosen from 15.12.2008 up to 30.12.2008.

## 5.8. 3D Visualization of the Mountain Panel

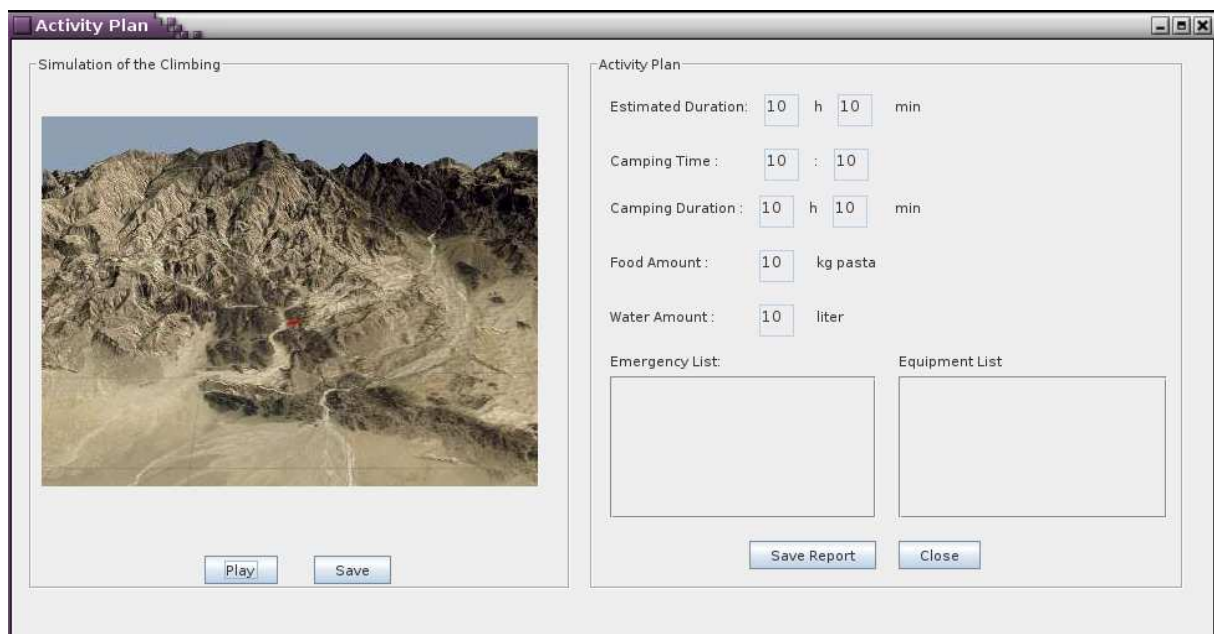
In this panel the 3D visualization of the mountain using all of the given maps will be shown.

- As user moves the mouse on the terrain, the longitude, latitude and elevation of the terrain will be shown in given boxes.
- If user clicks left mouse button then a popup menu will appear. In the popup menu the following options exist:
  - **Start coordinate:** If this option is chosen, then the clicked position will be the start coordinate. If there is a start point given before, that point will be removed since there can be only one start point.
  - **End coordinate:** If this option is chosen, then the clicked position will be the end coordinate. If there is an end point given before, that point will be removed since there can be only one end point.
  - **Checkpoint:** If this option is chosen, the clicked position will be added to the checkpoints list. During the preparation of the activity plan, passing through the given checkpoints will be taken into account as a constraint.
  - **Risk of avalanche:** By choosing this option, user can point the risk of avalanche.
  - **Risk of falling rocks:** By choosing this option, user can point the risk of falling rocks.
- If user clicks right mouse button, if there exists any points defined on that position, it will be removed.
- If user clicks the save image button, a file chooser will appear and pressing save, saves the image to a desired location.
- User can view the other sides of the mountain by pressing “Rotate Left” and “Rotate Right” buttons or keyboard’s left and right arrow keys.

- User can zoom in and out the terrain by pressing “Zoom In” and “Zoom Out” buttons or keyboard’s up and down arrow keys.

## 5.9. Get Activity Plan Window

When user presses Get Activity Plan Button, if the start or end coordinate was not given, the software will give warning. After marking the missing coordinate(s), the window shown in Figure 5.6 will appear.



*Figure 5.6. Activity Plan Window*

From the Activity Plan Window, user can simulate the climbing by pressing “Play Button”. If he presses “Save Button”, a file chooser appears and pressing “Save”, saves the simulation to the desired location.

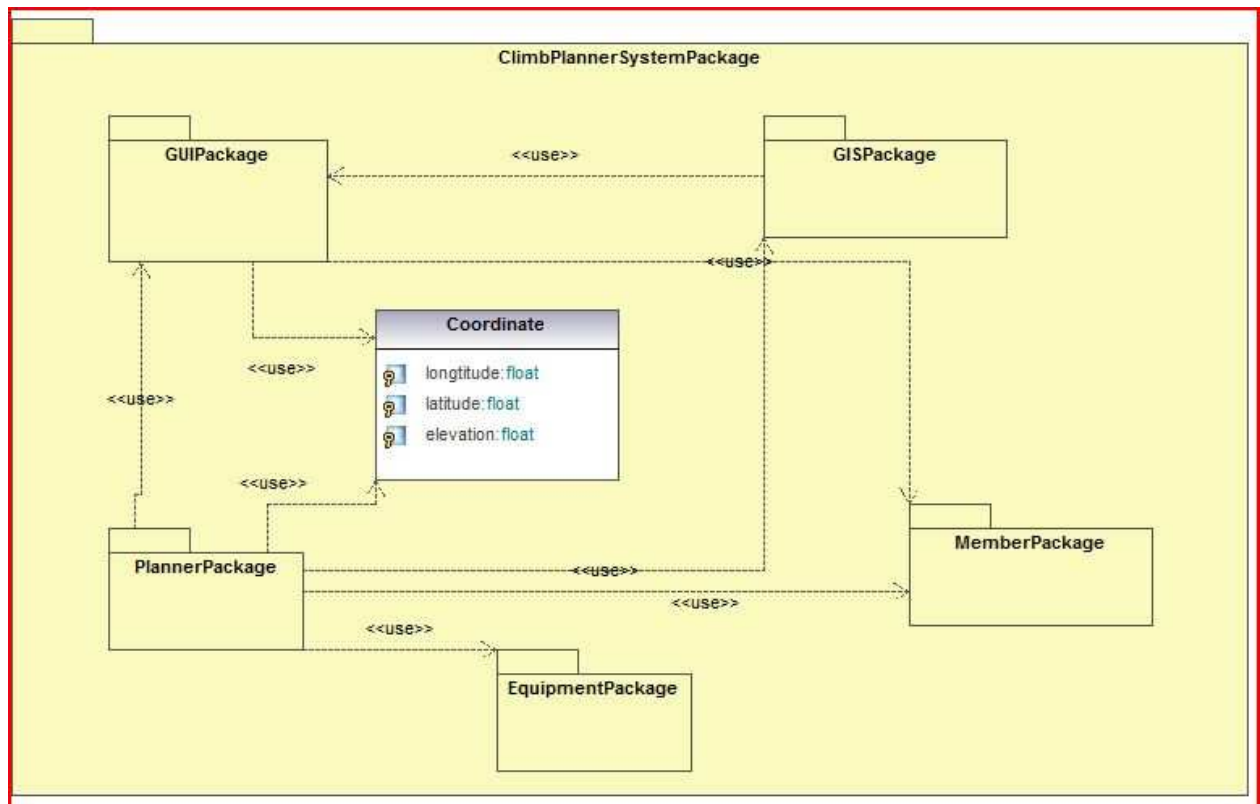
In the Activity Plan Panel, there exists the generated output about the climbing. In case of more than one camping, all of the campings will be listed. If there is no camping, nothing will be displayed for camping.

If user presses “Save Report”, a file chooser for saving the activity plan will appear. Pressing “Save”, saves the report. If user presses “Close”, the Activity Plan will be closed without saving the activity plan report.

## 6. ARCHITECTURAL DESIGN

### 6.1. Overall Architecture

Climb Planner Software is designed to be composed of 5 different packages namely GUI package, GIS package, Equipment package, Planner package and Member package that interact with each other. You can see the general view of the system in Figure 6.1. detailed design and explanations of the packages are given in following subsections.



6.1. General View of the System



## 6.2. Package and Class Design

### 6.2.1. GUIPackage

GUIPackage is the package that handles the user interaction and commands. Inputs are obtained from the user to be used in other packages for planning. Some important features it provides to user can be listed as follows:

- User can manage the member database of the club
- User can see the 3D visualization of the terrain and can mark coordinates on the image for different purposes: start and end point specification, checkpoint specification, avalanche risk specification, falling rock risk specification.
- User can see the simulation of the prepared plan.

Below you can find the class diagram of the GUIPackage.

As you can see in the in Figure 6.2 we have omitted the class data and method fields to fit the diagram into the page. Below you can find the individual classes of the package in their expanded forms.

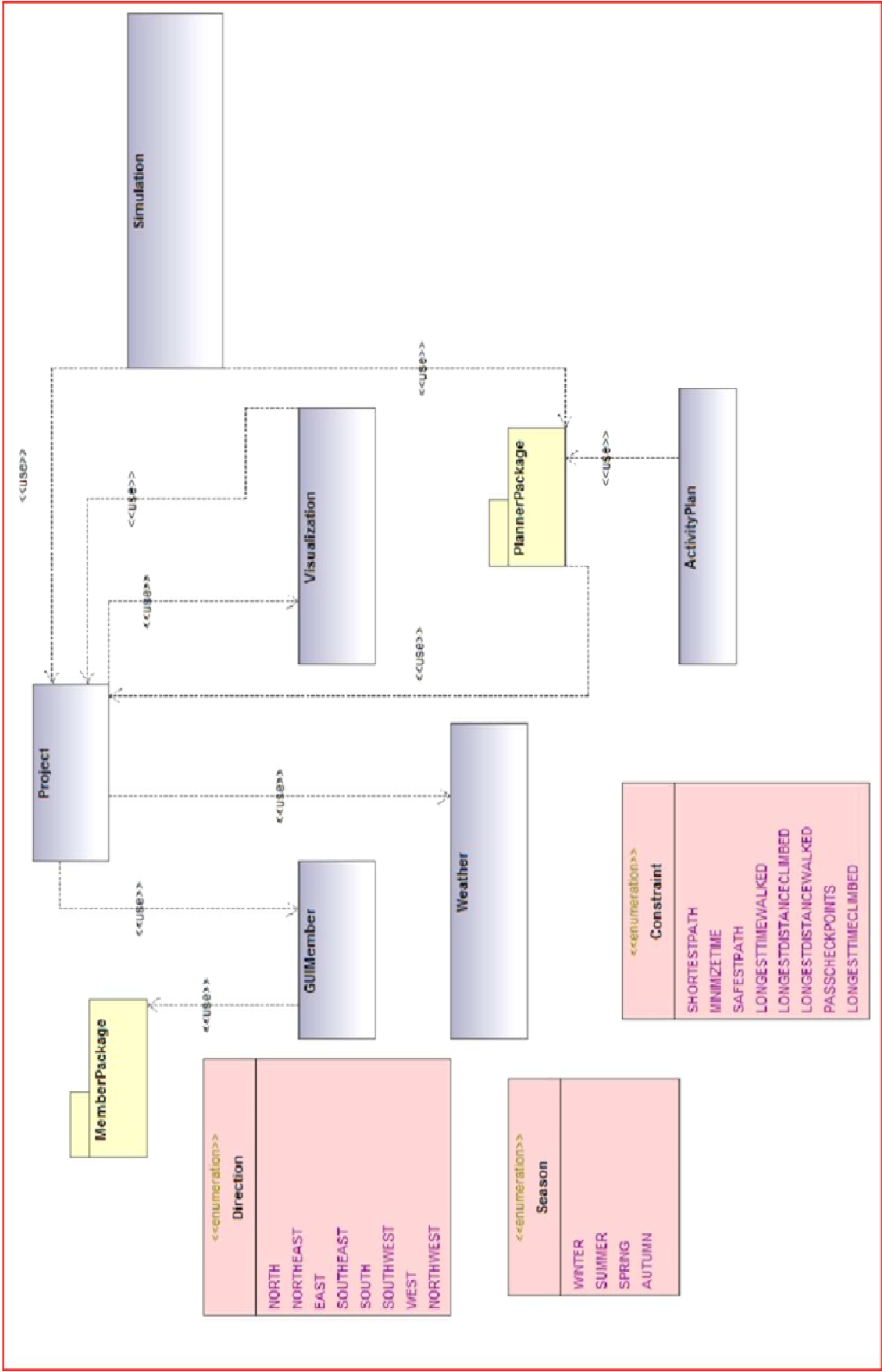


Figure 6.2. GUIPackage Class Diagram

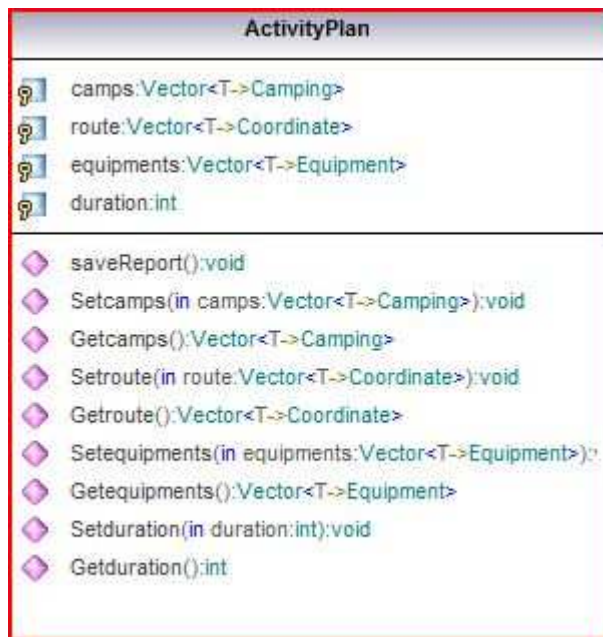


Figure 6.3. ActivityPlan Class

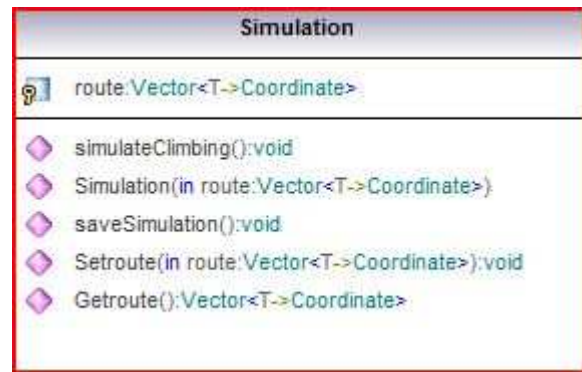


Figure 6.4. Simulation Class



Figure 6.5. GUIMember Class

Underlined methods represent Static methods. Vector<T->type\_name> representation is used to show Vector<type\_name> data types on this document. Also italic class names represents abstract classes.




































Visualization	
	start:Coordinate
	end:Coordinate
	checkPoints:Vector<T->Coordinate>
	avalanche:Vector<T->Coordinate>
	fallingRocks:Vector<T->Coordinate>
	maps:Vector<T->Map>
	currentPoint:Coordinate
	handleMouseMotion():void
	handleMouseClick():void
	removeStartPoint():void
	removeStartPoint():void
	removeCheckPoint():void
	removeAvalanche():void
	removeFalingRock():void
	renderTerrain():void
	saveTerrain():void
	rotateLeft():void
	rotateRight():void
	handleKeyboard(in leftorright:boolean):void
	Setstart(in start:Coordinate):void
	Getstart():Coordinate
	Setend(in end:Coordinate):void
	Getend():Coordinate
	SetcheckPoints(in checkPoints:Vector<T->Coordinate>):void
	GetcheckPoints():Vector<T->Coordinate>
	Setavalanche(in avalanche:Vector<T->Coordinate>):void
	Getavalanche():Vector<T->Coordinate>
	SetfallingRocks(in fallingRocks:Vector<T->Coordinate>):void
	GetfallingRocks():Vector<T->Coordinate>
	Setmaps(in maps:Vector<T->Map>):void
	Getmaps():Vector<T->Map>
	SetcurrentPoint(in currentPoint:Coordinate):void
	GetcurrentPoint():Coordinate
	Visualization()
	Visualization(in start:Coordinate, in end:Coordinate, in check:Vector<T->Coordinate>, in c

Figure 6.6. Visualization Class




Project	
	projectName:String
	projectPath:String
	chosenMembers:Vector<T>Member>
	projectPathMaps:Vector<T>String>
	country:String
	city:String
	projectCons:Vector<T>Constraint>
	startDate:Date
	terrain:Visualization
	weather:Weather
	createNewProject():void
	loadProject(in path:String):void
	saveProject(in path:String):void
	SetprojectName(in projectName:String):void
	GetprojectName():String
	SetprojectPath(in projectPath:String):void
	GetprojectPath():String
	SetchosenMembers(in chosenMembers:Vector<T>Member>):void
	GetchosenMembers():Vector<T>Member>
	SetprojectPathMaps(in projectPathMaps:Vector<T>String>):void
	GetprojectPathMaps():Vector<T>String>
	Setcountry(in country:String):void
	Getcountry():String
	Setcity(in city:String):void
	Getcity():String
	SetprojectCons(in projectCons:Vector<T>Constraint>):void
	GetprojectCons():Vector<T>Constraint>
	SetstartDate(in startDate:Date):void
	GetstartDate():Date
	Setterrain(in terrain:Visualization):void
	Getterrain():Visualization
	Setweather(in weather:Weather):void
	Getweather():Weather

Figure 6.7. Project Class

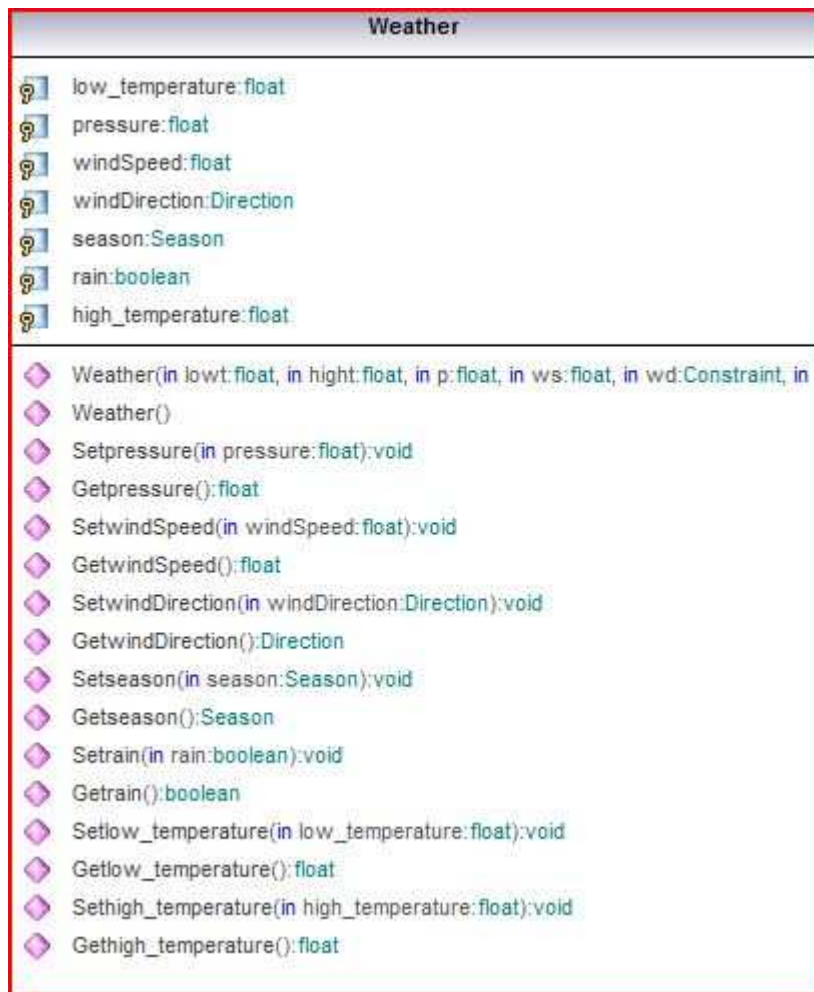
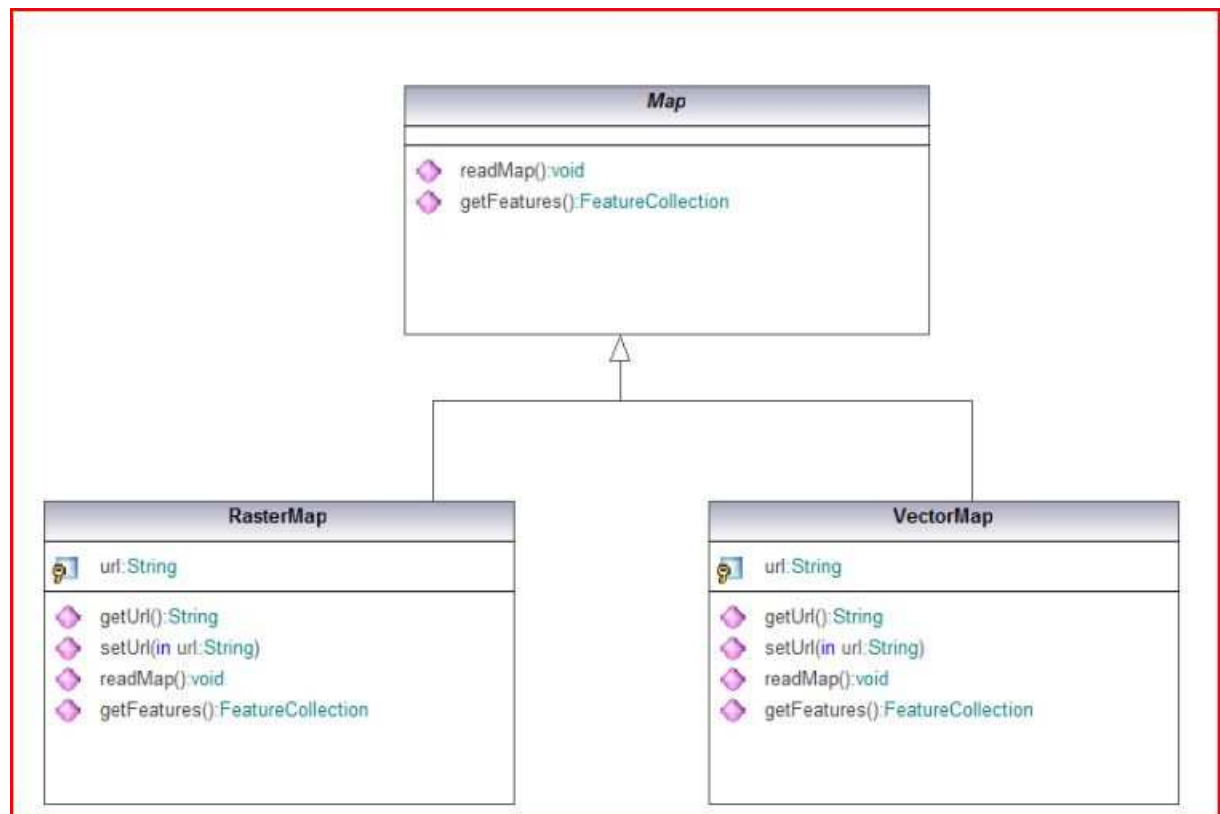


Figure 6.8. Weather Class

### 6.2.2. GISPackage

This is the package that extracts data from the map file paths obtained from GUIPackage. This package is capable to get features of both raster and vector maps. Below you can find the class diagram of GISPackage.



*Figure 6.9. GISPackage Class Diagram*

### 6.2.3. MemberPackage

MemberPackage is the package that apply the changes and operations to the database of members. User always interacts with the GUIPackage when applying some changes to the member database but these changes are actually applied and reflected to the database by MemberPackage. GUIPackage is not allowed to access database directly. Below you can find the class diagram of this package.



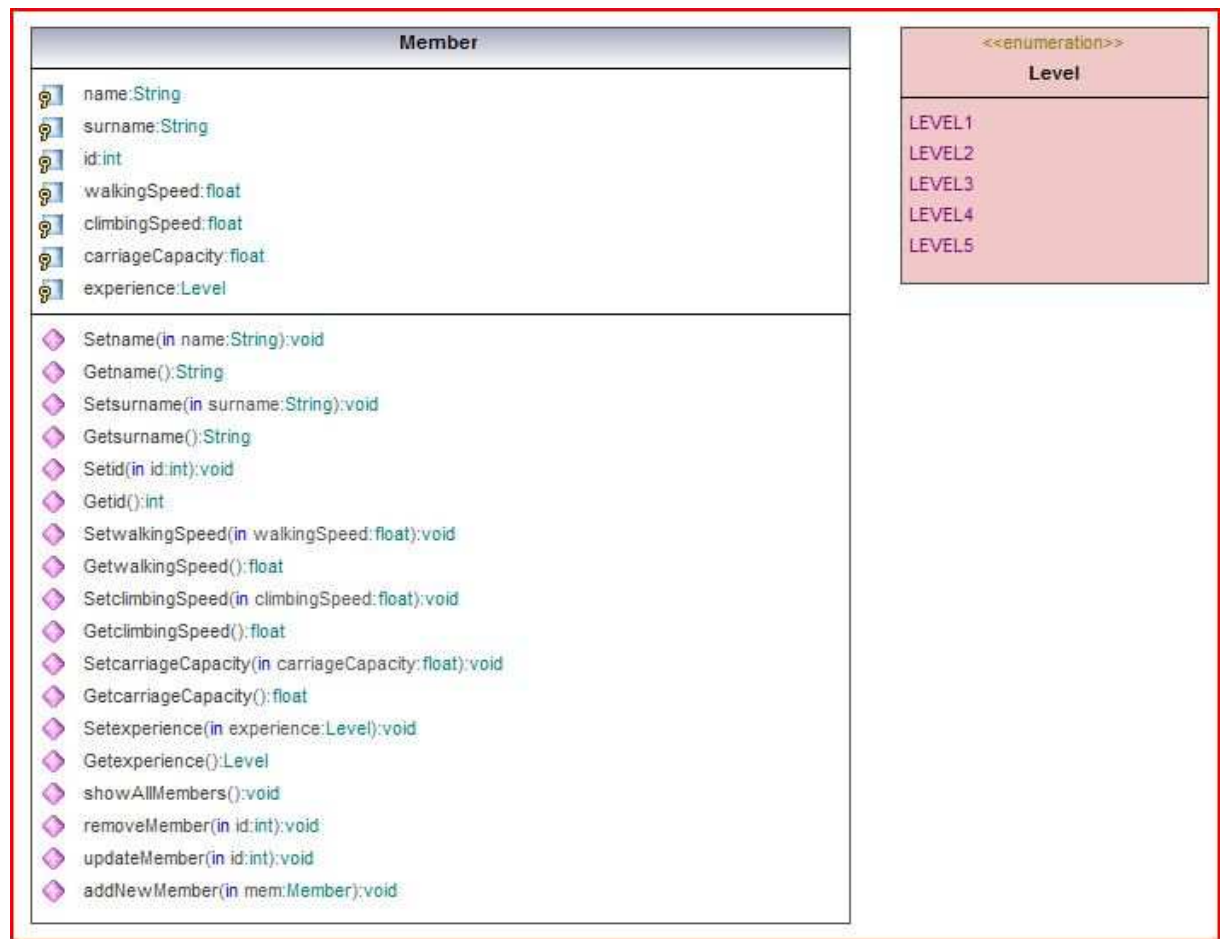


Figure 6.10. Member Class

#### 6.2.4. EquipmentPackage

EquipmentPackage has only one class named Equipment. It is for getting data from the database of equipments.





Figure 6.11. Equipment Class

#### 6.2.5. PlannerPackage

PlannerPackage is the package that the plan of the activity is prepared. There are two classes in this package: Camping class and Planner class. Planner class has the method to prepare the plan, Camping is a helper class.

The method makePlan() in the Planner class prepares the activity plan. Our algorithm will work in makePlan() function. Since we do not have a well-defined algorithm to prepare plan at this step, we will only give some information about the working principals of our planner algorithm.

Activity plan consists of a route, a camping plan including camping regions and camping duration, the optimized list of equipments that are required for the climbing and an optimized food list. Among these members of the plan, route is the one that dominates the others. So the algorithm should first find an optimized route for the activity meeting the constraints specified by the user.

The algorithm will make decision on paths considering the least experienced, slowest and weakest members of the group as a basis. The paths that require more experience than the least experienced member of the group will not be chosen. Duration of the activity will be decided according to the speed of slowest member in climbing or walking.

Walking speed of the members will be reduced at nights, at snowy and rainy weathers. At rainy and snowy weathers walking will be preferred climbing, actually climbing will not be an option.

Safest route is the route that there is less or no avalanche and falling rocks risks. Also walking will be taken as safer than climbing. When minimizing time of the activity is a constraint, safest route will be considered as shortest route in duration, because the risky activities like climbing, passing rivers, are time consuming activities even if they are on the shortest path. This is one of the mountaineering knowledge obtained from reliable sources.

Camping is not mandatory. However, if necessary, the duration of the camping will always be minimized. Equipment list will be prepared according to the requirements of the path and will be optimized. Food list will be prepared according to the duration of the activity. The load of the group should be minimized.

Below you can find the class diagram of PlannerPackage.

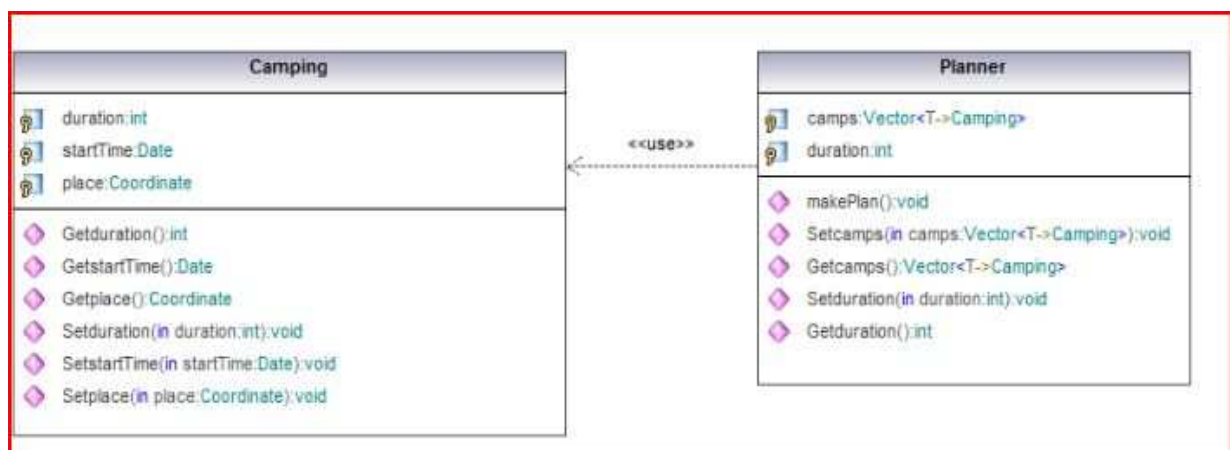
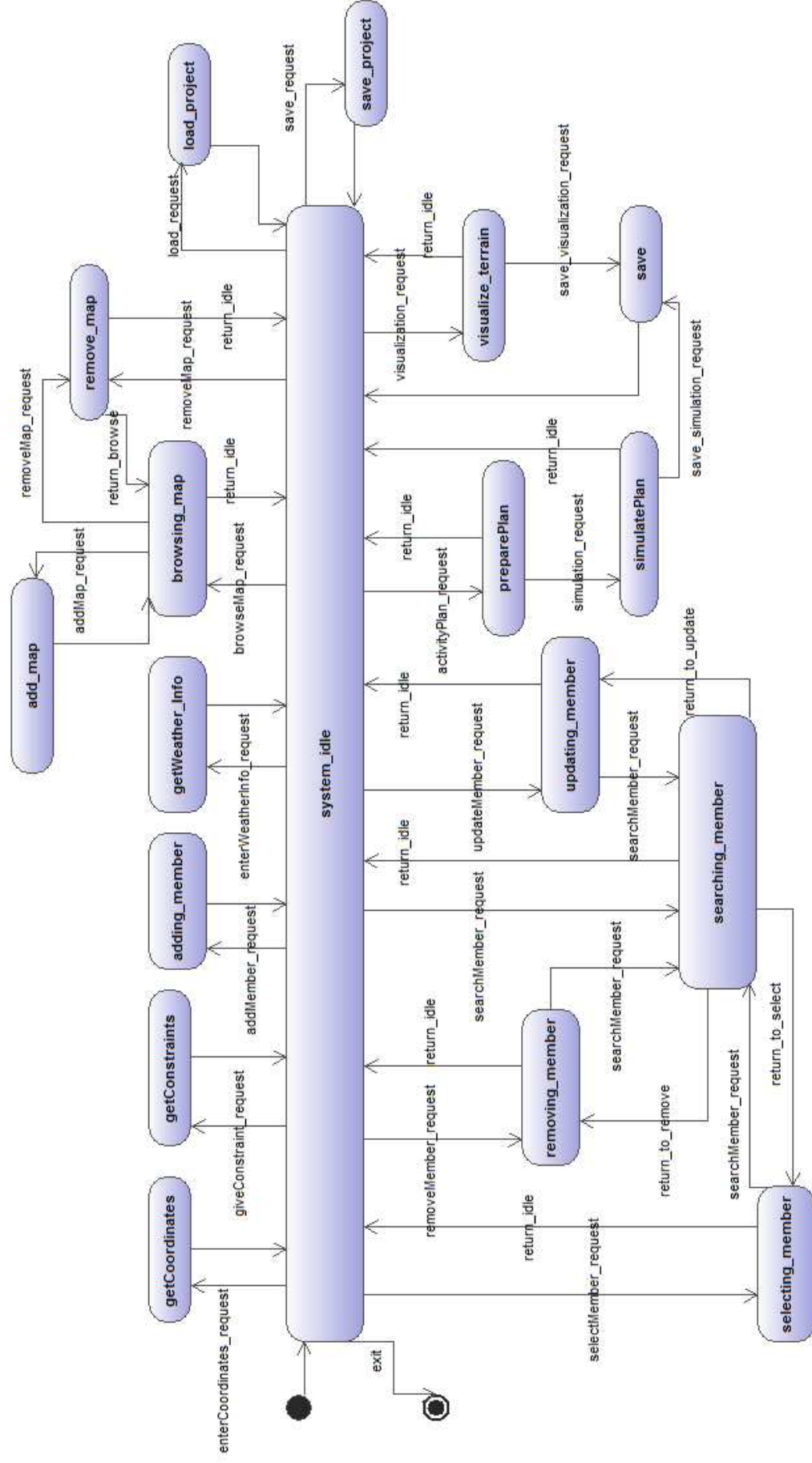


Figure 6.12. PlannerPackage Class Diagram

### 6.3. Behavioral Design



*Figure*

6.13.

## State

## Transition

### Diagram

## 6.4. Functional Modeling

### 6.4.1. Data Flow Diagrams

- **Context Level Data Flow Diagram**

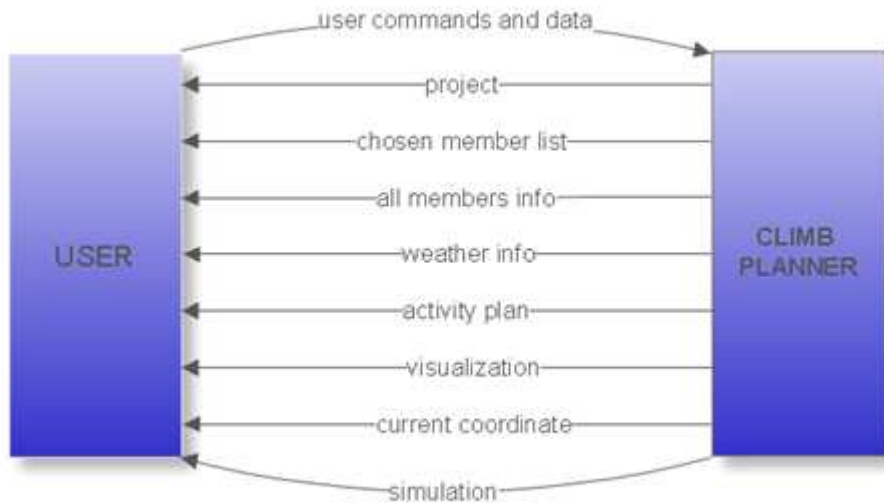


Figure 6.14. Context Level DFD

- **Level 1 Data Flow Diagram**

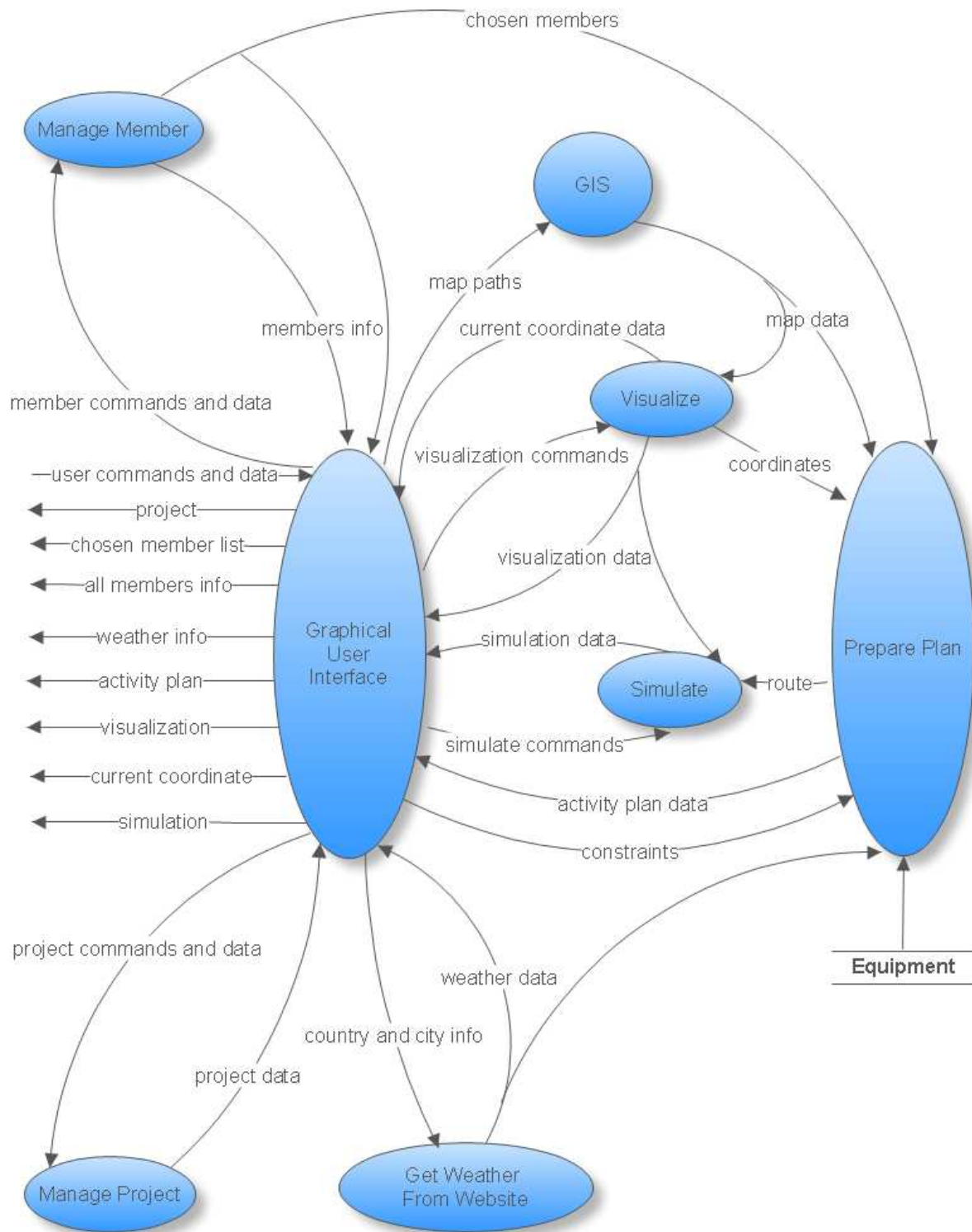


Figure 6.15. Level 1 DFD

- **Level 2 Data Flow Diagram : Visualize**

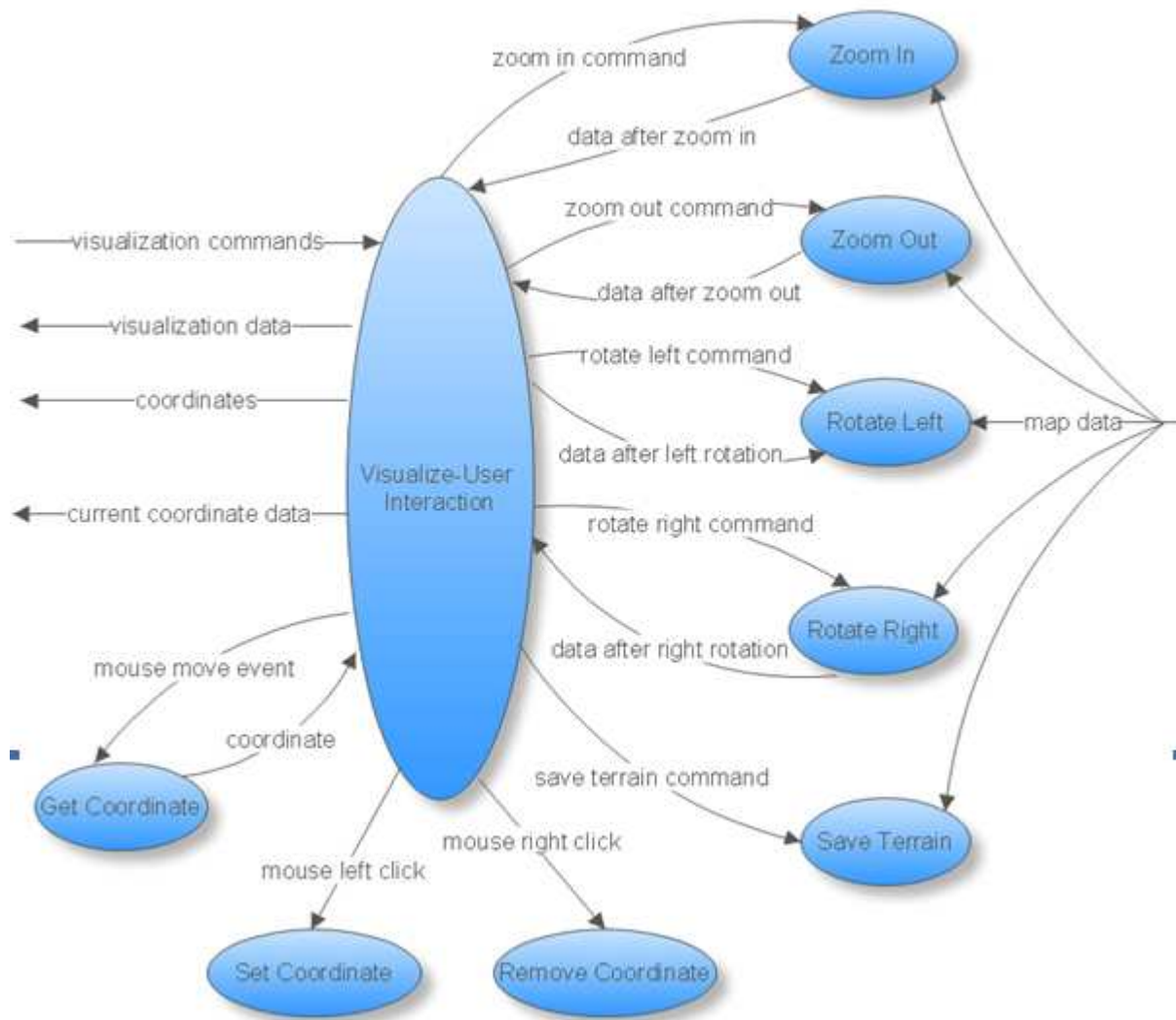
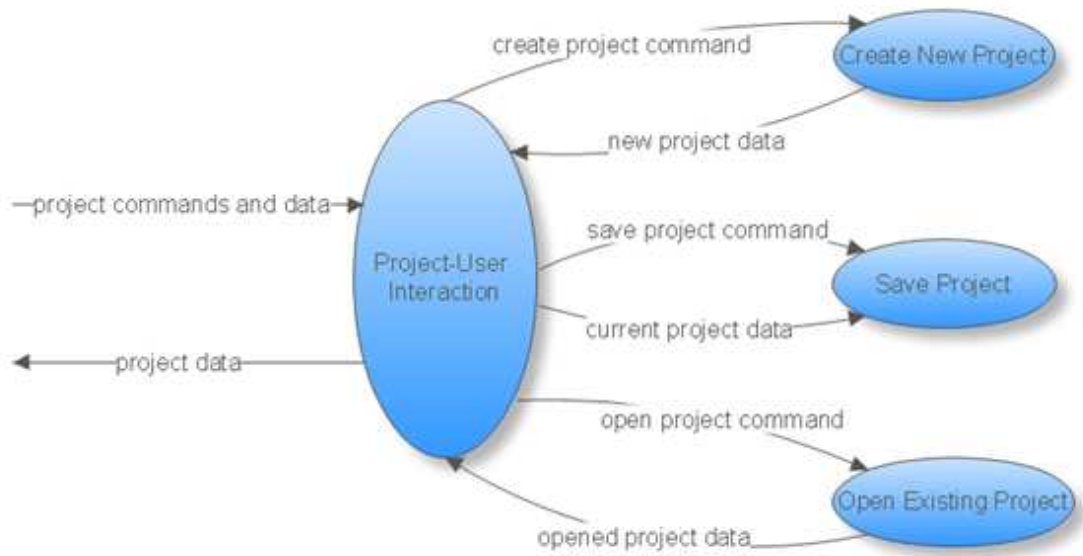


Figure 6.16. Level 2 DFD: Visualize

- **Level 2 DataFlowDiagram : Manage Project**



*Figure 6.17. Level 2 Data Flow Diagram: Manage Project*

- **Level 2 Data Flow Diagram : Manage Member**

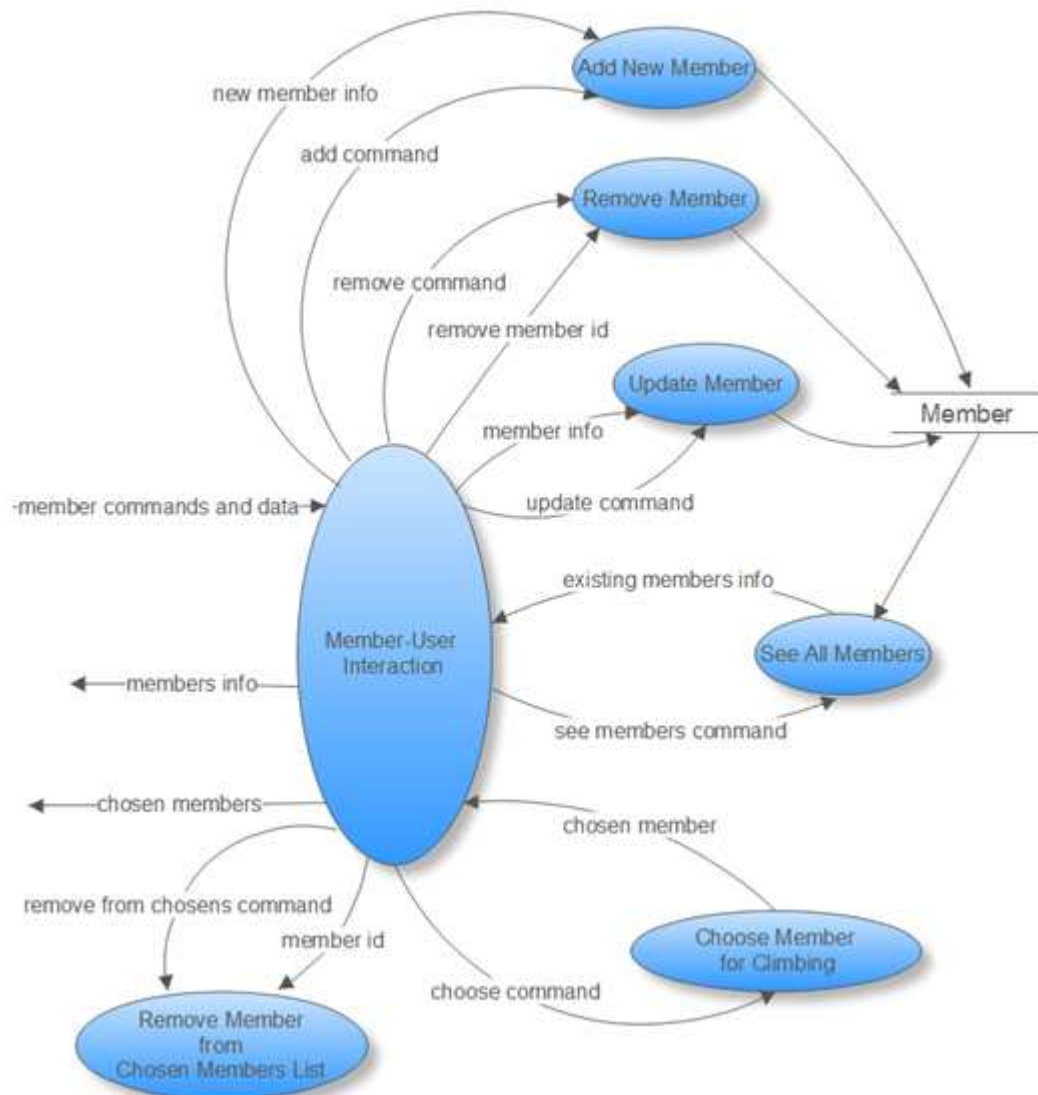


Figure 6.18. Level 2 DFD: Manage Member



#### 6.4.2. *Data Dictionary*

<b>Name:</b>	<b>user commands and data</b>
<b>From:</b>	USER
<b>To:</b>	Graphical User Interface
<b>Description:</b>	Contains all commands and data taken from user

<b>Name:</b>	<b>member commands and data</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Member-User Interaction
<b>Description:</b>	Contains all commands and data related to member

<b>Name:</b>	<b>add command</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Add New Member
<b>Description:</b>	The command for adding a new member to the system

<b>Name:</b>	<b>new member info</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Add New Member
<b>Description:</b>	Holds the information about the member that will be added to the system

<b>Name:</b>	<b>remove command</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Remove Member
<b>Description:</b>	The command for removing a member from the system

<b>Name:</b>	<b>remove member id</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Remove Member
<b>Description:</b>	The id of the member who will be removed from the system permanently

<b>Name:</b>	<b>update command</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Update Member
<b>Description:</b>	The command for updating an existing member in the system

<b>Name:</b>	<b>member info</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Update Member
<b>Description:</b>	The member whose information will be updated

<b>Name:</b>	<b>remove from chosens command</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Remove Member from Chosen Members List
<b>Description:</b>	The command for removing a member from the list that contains the members who are going to participate in the activity (chosen member list)

<b>Name:</b>	<b>member id</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Remove Member from Chosen Members List
<b>Description:</b>	The member id of the person who will be removed from the chosen members list (not from the system)

<b>Name:</b>	<b>choose command</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Choose Member for Climbing
<b>Description:</b>	The command for choosing a member to participate in the activity

<b>Name:</b>	<b>chosen member</b>
<b>From:</b>	Choose Member for Climbing
<b>To:</b>	Member-User Interaction
<b>Description:</b>	Holds information about the recently chosen member for climbing

<b>Name:</b>	<b>see members command</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	See All Members
<b>Description:</b>	The command for showing information about all members registered in the system

<b>Name:</b>	<b>existing members info</b>
<b>From:</b>	See All Members
<b>To:</b>	Member-User Interaction
<b>Description:</b>	Holds information about all members registered in the system

<b>Name:</b>	<b>members info</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Graphical User Interface
<b>Description:</b>	Holds information about all members to show them to the user

<b>Name:</b>	<b>chosen members</b>
<b>From:</b>	Member-User Interaction
<b>To:</b>	Graphical User Interface, Prepare Plan
<b>Description:</b>	Holds the information about the members chosen for climbing to show those members to the user and to help preparing activity plan

<b>Name:</b>	<b>chosen member list</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	Holds the members chosen for the climbing to show those members to the user

<b>Name:</b>	<b>all members info</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	Holds the information of all members existing in the system to show those members information to the user

<b>Name:</b>	<b>map paths</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	GIS
<b>Description:</b>	Contains all given map locations given by the user (it includes "dted", "dem", "geotiff" and "shp" file paths)

<b>Name:</b>	<b>map data</b>
<b>From:</b>	GIS
<b>To:</b>	Visualize, Prepare Plan
<b>Description:</b>	The map data gathered after reading the map files by GIS.

<b>Name:</b>	<b>visualization commands</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Visualize-User Interaction
<b>Description:</b>	commands to change the settings of visualization

<b>Name:</b>	coordinates
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Graphical User Interface
<b>Description:</b>	Holds the start and end coordinates, checkpoints and the points that has avalanche or falling rock risks to mark those point in the visualization of the terrain

<b>Name:</b>	<b>current coordinate data</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Graphical User Interface
<b>Description:</b>	As mouse moves on the terrain, the corresponding coordinate is shown. Current coordinate data holds longitude, latitude and elevation of the corresponding coordinate

<b>Name:</b>	<b>current coordinate</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	Current coordinate shown to the user

<b>Name:</b>	<b>zoom in command</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Zoom In
<b>Description:</b>	The command for zooming in the terrain

<b>Name:</b>	<b>visualization data</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Graphical User Interface, Simulation
<b>Description:</b>	Holds the necessary information to visualize the terrain

<b>Name:</b>	<b>data after zoom in</b>
<b>From:</b>	Zoom In
<b>To:</b>	Visualize-User Interaction
<b>Description:</b>	Holds the visualization data after zooming in the terrain to show the effect of zooming in to the user

<b>Name:</b>	<b>zoom out command</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Zoom Out
<b>Description:</b>	The command for zooming out from the terrain

<b>Name:</b>	data after zoom out
<b>From:</b>	Zoom Out
<b>To:</b>	Visualize-User Interaction
<b>Description:</b>	Holds the visualization data after zooming out from the terrain to show the terrain with zoomed out to the user

<b>Name:</b>	<b>rotate left command</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Rotate Left
<b>Description:</b>	The command for rotating left in the terrain

<b>Name:</b>	<b>data after rotate left</b>
<b>From:</b>	Rotate Left
<b>To:</b>	Visualize-User Interaction
<b>Description:</b>	Holds the visualization data after rotating left in the terrain

<b>Name:</b>	<b>rotate right command</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Rotate Right
<b>Description:</b>	The command for rotating right in the terrain

<b>Name:</b>	<b>data after rotate right</b>
<b>From:</b>	Rotate Right
<b>To:</b>	Visualize-User Interaction
<b>Description:</b>	Holds the visualization data after rotating right in the terrain

<b>Name:</b>	<b>save terrain command</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Save Terrain
<b>Description:</b>	The command for saving the terrain in a user defined location

<b>Name:</b>	<b>mouse move event</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Get Coordinate
<b>Description:</b>	Action of mouse movement in the terrain to get the corresponding coordinate

<b>Name:</b>	<b>coordinate</b>
<b>From:</b>	Get Coordinate
<b>To:</b>	Visualize-User Interaction
<b>Description:</b>	Contains the coordinate of the corresponding point in the terrain

<b>Name:</b>	<b>mouse left click</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Set Coordinate
<b>Description:</b>	If user clicks the left button of the mouse on the terrain, the corresponding coordinate is set as one of the followings : start, end, checkpoint, avalanche risk, falling rock risk

<b>Name:</b>	<b>mouse right click</b>
<b>From:</b>	Visualize-User Interaction
<b>To:</b>	Remove Coordinate
<b>Description:</b>	If user clicks the right button of the mouse on the terrain, if there exists a predefined coordinate (start, end, etc.), that coordinate will be removed.

<b>Name:</b>	<b>project commands and data</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Project-User Interaction
<b>Description:</b>	Contains all commands and data related to project

<b>Name:</b>	<b>project data</b>
<b>From:</b>	Project-User Interaction
<b>To:</b>	Graphical User Interface
<b>Description:</b>	Holds the current project data that are necessary when loading a project

<b>Name:</b>	<b>create project command</b>
<b>From:</b>	Project-User Interaction
<b>To:</b>	Create New Project
<b>Description:</b>	The command for creating a new project

<b>Name:</b>	<b>new project data</b>
<b>From:</b>	Create New Project
<b>To:</b>	Project-User Interaction
<b>Description:</b>	Holds the data of the newly created project

<b>Name:</b>	save project command
<b>From:</b>	Project-User Interaction
<b>To:</b>	Save Project
<b>Description:</b>	The command for saving the project

<b>Name:</b>	<b>current project data</b>
<b>From:</b>	Project-User Interaction
<b>To:</b>	Save Project
<b>Description:</b>	Holds the current project data to be saved

<b>Name:</b>	<b>open project command</b>
<b>From:</b>	Project-User Interaction
<b>To:</b>	Open Existing Project
<b>Description:</b>	The command for opening an existing project

<b>Name:</b>	<b>opened project data</b>
<b>From:</b>	Open Existing Project
<b>To:</b>	Project-User Interaction
<b>Description:</b>	Holds the data of the opened project

<b>Name:</b>	<b>project</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	The data to show the current settings of the project to the user

<b>Name:</b>	<b>simulation data</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Simulate
<b>Description:</b>	The data combined with route and terrain

<b>Name:</b>	<b>simulate commands</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Simulate
<b>Description:</b>	Contains playing and saving simulation commands

<b>Name:</b>	<b>country and city info</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Get Weather From Website
<b>Description:</b>	Country and city of the mountain to get the weather conditions from a website

<b>Name:</b>	<b>weather data</b>
<b>From:</b>	Get Weather From Website
<b>To:</b>	Graphical User Interface, Prepare Plan
<b>Description:</b>	Weather data taken from the website to show the user and to use in preparing plan

<b>Name:</b>	<b>weather info</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	Weather information shown to the user

<b>Name:</b>	<b>constraints</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	Prepare Plan
<b>Description:</b>	Holds all given constraints by user

<b>Name:</b>	<b>route</b>
<b>From:</b>	Prepare Plan
<b>To:</b>	Simulate
<b>Description:</b>	The route of the climbing that is shown to the user in simulation

<b>Name:</b>	<b>activity plan data</b>
<b>From:</b>	Prepare Plan
<b>To:</b>	Graphical User Interface
<b>Description:</b>	Generated activity plan data to show the user and save the activity plan report

<b>Name:</b>	<b>activity plan</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	The activity plan shown to the user including the duration of the climbing, camping location and times, food, equipment and emergency equipment list

<b>Name:</b>	<b>visualization</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	Visualization of the terrain shown to the user



<b>Name:</b>	<b>simulation</b>
<b>From:</b>	Graphical User Interface
<b>To:</b>	USER
<b>Description:</b>	Simulation of the climbing shown to the user

## 6.5. Database Design

Since the GIS tool that we are going to use (GeoTools) is capable of storing the map information in its own data store, we don't need to create any table for storing maps. So, the data that we will store is not complex. We have two basic tables for storing member information and equipment information.

### 6.5.1. Tables

- **Member Table**

Member Table holds the basic information about a member. As user adds, removes or updates a member, his information is saved to database by using this table. In the next steps of the project, new fields can be added to this table when needed.

Field Name	Data Type	Description
<u>Id</u>	INTEGER	Unique id of the member
Name	VARCHAR(15)	Name of the member
Surname	VARCHAR(20)	Surname of the member
ClimbingSpeed	FLOAT	Climbing speed of the member in m/h.
WalkingSpeed	FLOAT	Walking speed of the member in km/h.
CarriageCapacity	FLOAT	Carriage capacity of the member in kg.
ExperienceLevel	INTEGER	Experience level of the member. (Level1 – Level 5)
PRIMARY_KEY(Id)		

Table 6.1. Member Table

- **Equipment Table**

Equipment Table holds the basic information about an equipment. It also includes the equipments that are needed in case of emergency. This table is needed for preparing the activity plan. User will not be able to access Equipment Table. In the next steps of the project, new fields can be added to this table as needed.

Field Name	Data Type	Description
Id	INTEGER	Unique id of the equipment
Name	VARCHAR(30)	Name of the equipment
Weight	FLOAT	Weight of the equipment
NeededforClimbing	BOOLEAN	True if the equipment is needed for climbing
NeededforWalking	BOOLEAN	True if the equipment is needed for walking
NeededforCamping	BOOLEAN	True if the equipment is needed for camping
Season	INTEGER	season in which the equipment is used
Emergency	BOOLEAN	True if the equipment is an emergency equipment
PRIMARY_KEY(Id)		

*Table 6.2. Equipment Table*

### 6.5.2. ER Diagrams

Since we don't store any complex data and the data that we store are not related to each other, the ER diagrams are not complex, also. We have ER diagrams of only our database tables.

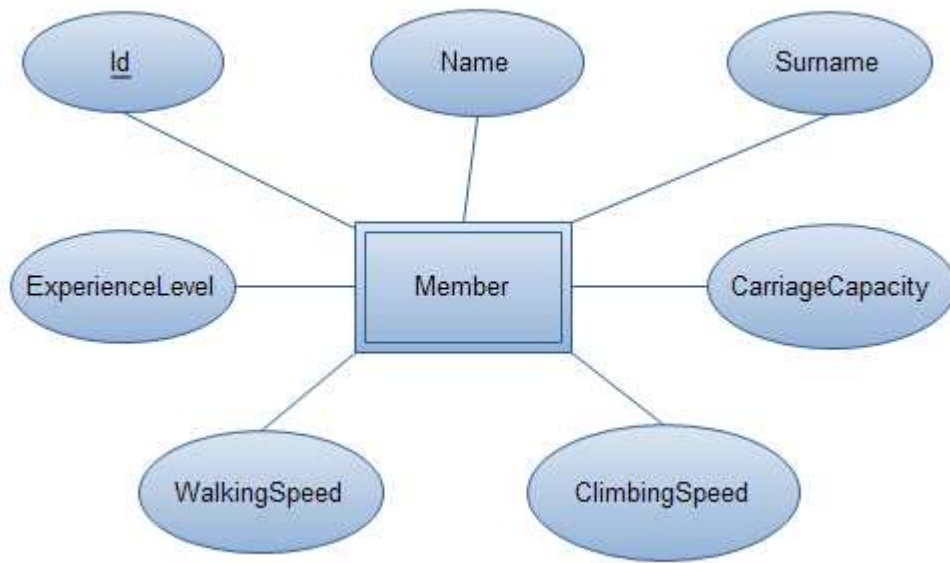


Figure 6.19. ER Diagram of Member

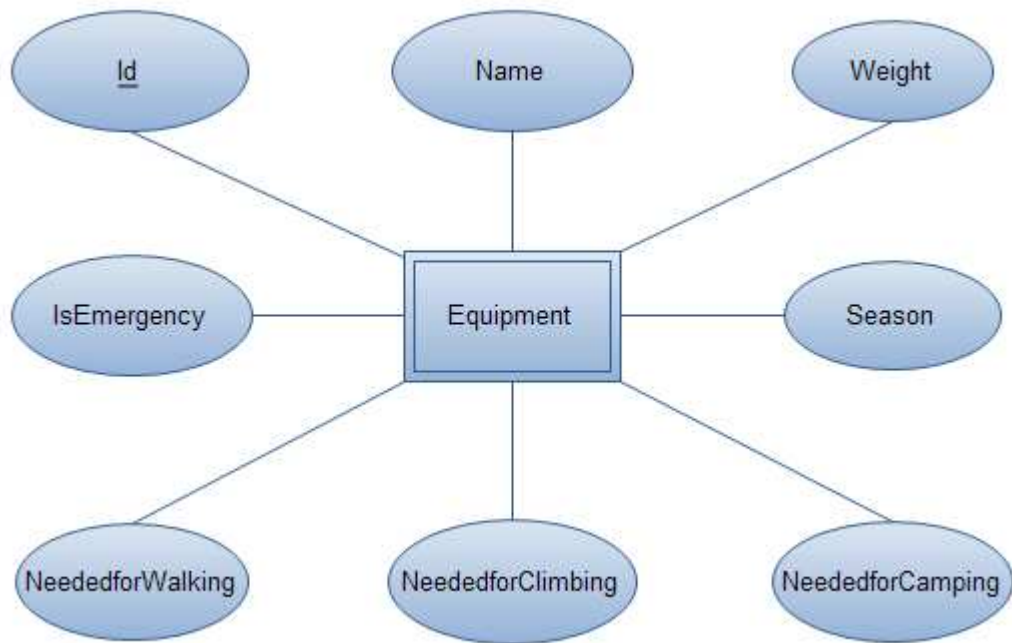


Figure 6.20. ER Diagram of Equipment

# 7. IMPLEMENTATION

## 7.1. Current State

We have started to use tools and libraries. To determine the GIS tool capabilities we have played with Geotools and seen some extra features more than that we had hoped like rendering terrain in 3D and lots of 3D graphics methods. On the other hand, when we designed the packages and classes of the project in Architectural Design part, we have determined the methods of classes as good as possible and defined some of their contents like database connections, graphical user interface interactions etc.

## 7.2. Prototype Implementation

The prototype of ClimbPlanner should contain an understandable graphical user interface with its main features. The software main features are opening a new project, saving and loading it. In addition to these, gui should be able to get user inputs and show a 3D visualization of the terrain. There should be a database connection for choosing climber for current plan and actions on climbers who exist in database. There can be a network connection for getting weather conditions.

Of course all these considerations will become final with respect to Aselsan's demands. We will do our implementation on prototype in this direction.

## 7.3. Future Work

In addition to these design decisions, we have thought about the Aselsan's feedback on equipment list issue. They suggested that we should think on a tool identification interface. We have discussed about it and we have realized that we can do it. We have thought it such that we will design a sheet which contains the possible features of equipments. If user wants to introduce a new equipment to the system, after asking the features of equipment particularly, the software can store it to equipment database. However, we can't add this interface to our design until now because of time shortness. In the detailed design report, we will add that interface. In addition, we will also add the equipments that will be used, to

the database in detailed design. We will also start the prototype implementation according to the specifications given by Aselsan on prototype demands.

## 8. CONCLUSION

During the preparation of this document, our team was aware of how important this stage of the project is, as initial design forms a basis for the future design works. Therefore, first we deepened our searches about the technical issues to create a realistic view. We considered every possible choice for libraries, tools and platforms and make more research on our choices. After this part, we needed to embody our abstractions in a clear, comprehensive, objective way to make this document lead us through the implementation smoothly. In the light of these beliefs, we tried to be clear during the design of hierarchy of the whole system, modules and classes. Moreover, we needed to specify and draw class, activity and ER diagrams so that any developer -including us should understand the architecture and design by analyzing the diagrams; we needed objectivity as well as realism. For the main architecture, we mostly considered consistency and performance; our team did its best for these purposes.

To summarize, Sirius Software team performed its best for this stage of project as our team is aware of the importance of the initial design mile stone. This document's aim is to enlighten us and readers through our project's development; therefore it will be upgraded after getting feedback from ASELSAN and our project advisor.

## 9. APPENDIX

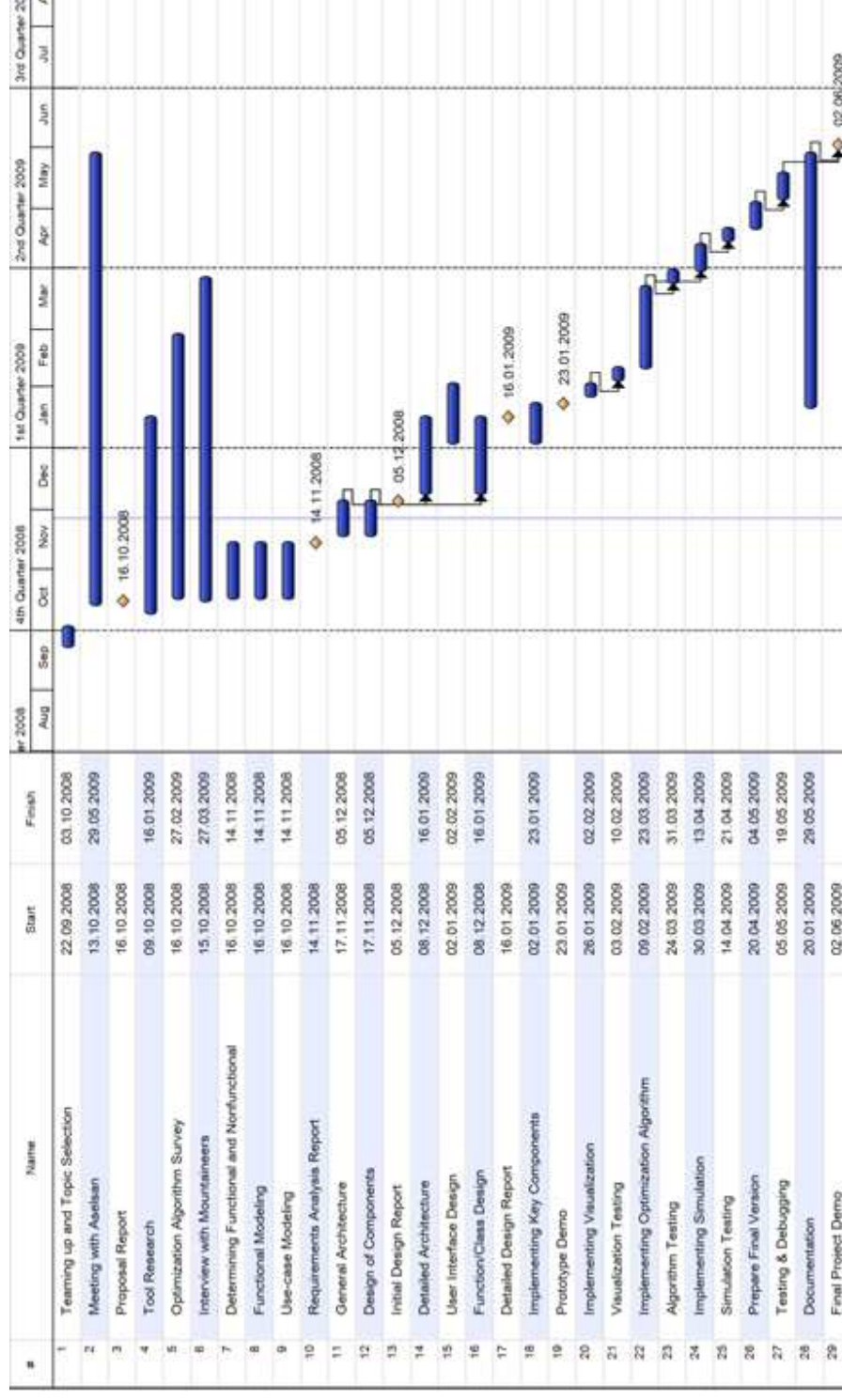


Figure 9.1. Gantt Chart

## 10. REFERENCES

- [1]. Java Programming Language, [http://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Java_(programming_language))
- [2]. Eclipse Platform, [http://en.wikipedia.org/wiki/Eclipse\\_\(software\)](http://en.wikipedia.org/wiki/Eclipse_(software))
- [3]. Netbeans IDE, <http://www.netbeans.org/features/4>
- [4]. GeoTools, Open Source Java GIS Tool, <http://geotools.codehaus.org/>
- [5] Extensible Markup Language, <http://www.xml.com/>
- [6] MySQL, <http://www.mysql.com/>
- [7] User Interfacer Desing Tips, Techniques, and Principles: <http://www.ambysoft.com/essays/userInterfaceDesign.html>