Project Ivanhoe

Group 16
Rebecca Everett
Tobias Hassellöf
Henrik Törnvall
Johan Renner
Martin Waara-Grape

Table of Contents

1. Introduction

The intended readers of this document are the group of people involved in designing, developing, testing and managing the release of this program. These also include the teachers and other students of the course (DD1363). This and all future appendices will be written in English.

This document will act as a reference for programming the Ivanhoe system. Its purpose is specially aimed at providing a framework for the detailed design considerations regarding the project. The scope of this document ranges from presenting the overall architecture to the details of every class to be programmed.

1.2 Brief contents description

The main contents of this document consist of:

- System Overview

  Provides a general description of the software system including its functionality and overall architecture. This section also includes a more detailed description of the architecture.

- Design Considerations

  This section describes many of the assumptions, dependencies and general constraints that apply to the design of this system.

- Graphical User Interface

  This section provides an overview of the Graphical User Interface (GUI) of the system.

- Design Details

  This section provides detailed class-level design information.

- Functional Test Cases

  Finally, this last section provides actual test cases that a system tester can follow to verify a specified functionality of the system.

1.3 Related documents

For more background documentation please see the following related documents:

- Project Overview Document
- Requirements Document
- Implementation Plan

1.4 Document history

- 2008-02-20 – The first draft of this document.

1.5 Important terms and abbreviations

GUI – Graphical User Interface
Ivanhoe – The application that this document descripes.
MVC – Model-View-Controller

For an extended and more detailed list of terminology please see the Ivanhoe Requirements Document.

2. System Overview

2.1 General Description

Project Ivanhoe is a software project for developing a Home Finance System (HFS) that lets the users input their daily expenses and expenditures. This in turn lets the users get a practical overview of his or her economical situation to make financial decisions. The users of the system are people with a home computer and a need for monitoring their personal finances. He/she is a person with income and expenses, who wants to be able to manage the money spent and earned in a more effective way. The system is designed for everyday use as well as special occasions. Some main functions of the program are to:

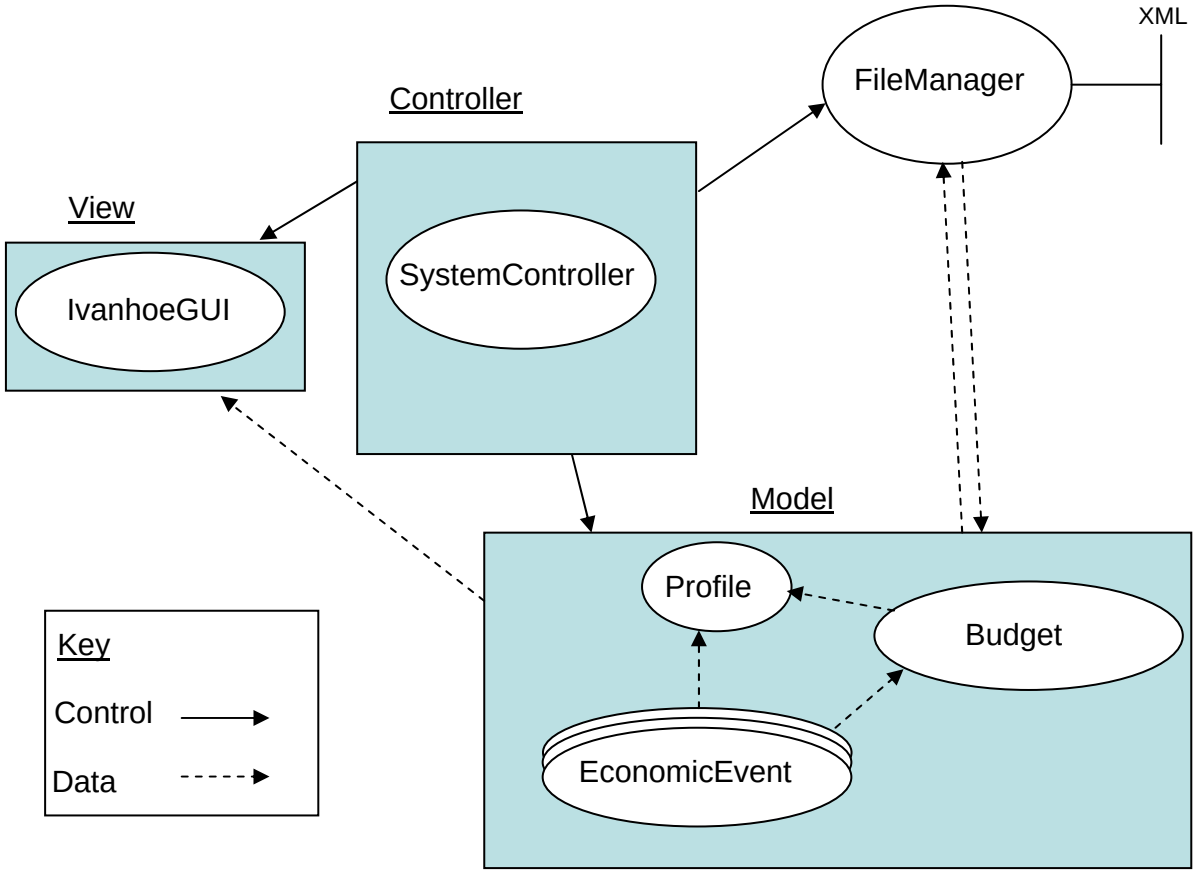
- create budgets
- monitor income and expenses.
- provide a graphical overview of the personal or family economy.
- compare expenses with the budget.
- monitor expenses, such as bills, with a reminder function

2.1.1 General design description

The program is implemented using a Model-View-Controller (MVC) design pattern. This helps the different presentation functions of the system to draw their data from a single data source. For more detailed information regarding the details of the architecture see section 2.2 of this document.

2.2 Overall Architecture Description

The Ivanhoe system is a Model-View-Controller system. The Model consists of a Profile, a Budget and EconomicEvents. The Controller consists of SystemController and the View consists of an IvanhoeGUI as pictured below.



The user interacts with the IvanhoeGUI, which shows the graphical user interface (GUI). The IvanhoeGUI shows the data stored in the profile such as the registered expenses and incomes. It also communicates with the SystemController when a new Profile, EventCollection or Budget should be created or modified.

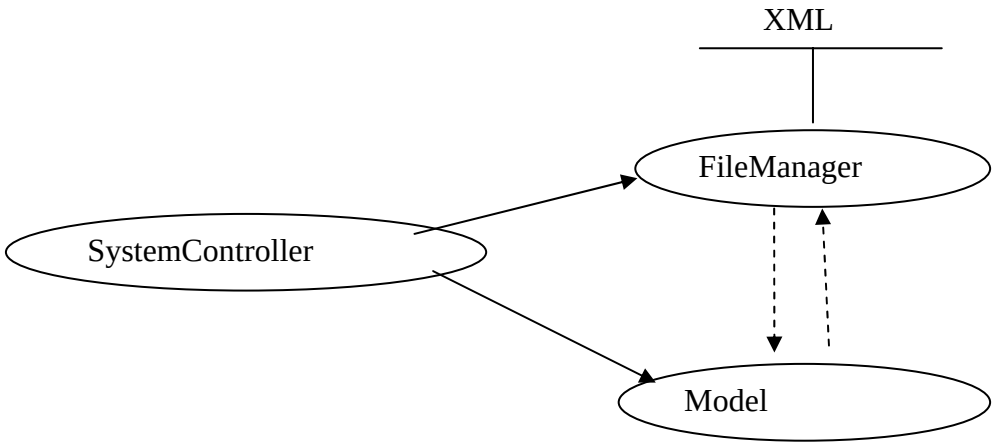
The SystemController communicates with the FileManager when information should be saved or retrieved. It handles the dataflow needed for the spreadsheets and graphs created by the IvanhoeGUI.

The Model performs calculations and other activities for creation and modifying the different parts of the Model.

The FileManager reads and writes eXtensible Markup Language-files (XML) stored on the computer. There is a link between the model and the FileManager to handle save status of the profile.

2.3 Detailed Architecture

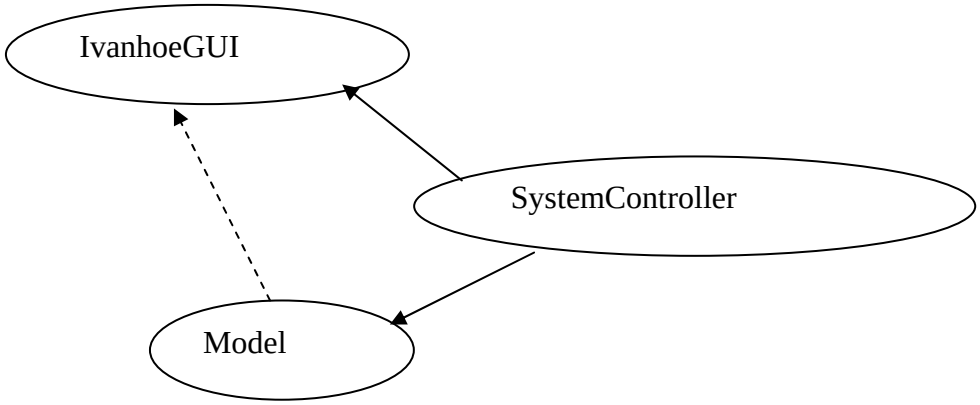
2.3.1 File manager:



The SystemController controls both the Model and the FileManager in a one-way control channel, deciding *when* data will be fetched or saved.

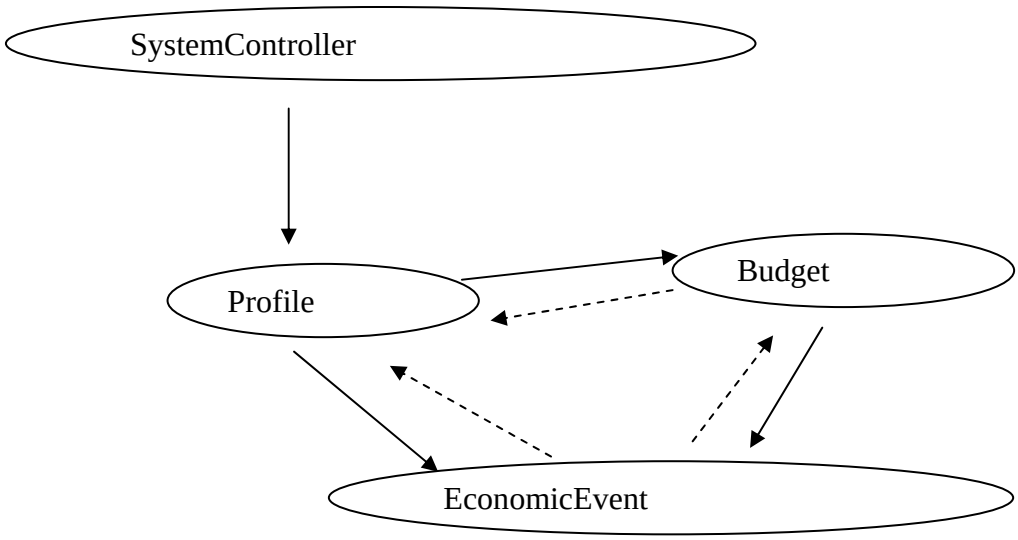
If data is to be fetched, it will be collected by the the Model, and if it is to be saved, the data will flow from the Model to the FileManager. The FileManager will interact with an XML file stored on the harddrive. The Model will parse the XML code into usable information.

2.3.2 GUI:



The control flow is one-way from the SystemController to the Model and GUI. The SystemController will decide what action to take when there is a user signal from the GUI. The GUI is an observer of the Model, always reflecting an up-to-date version of the Model.

2.3.3 Model:



The SystemController initiates an action concerning Profile, Budget, or EconomicEvent, through Profile, and Profile in turn tells the appropriate class to perform the action.

The Profile will keep track of Budget and EconomicEvent objects, and data thus flows from these objects to Profile.

3. Design Considerations

3.1 Assumptions and Dependencies

The design of this system is based on the following assumptions:

- The user is running Windows XP
- The user has the Java JRE software installed

4. Graphical User Interface

4.1 Graphical User Interface Overview

Ivanhoe uses Java Swing to implement the graphical user interface. The main view consists of a window with a menu bar and a few fields with access to the main functionality of the program.

The tabs to the right provide access to multiple functionalities of the program, including tools for creating, displaying and analyzing the user's finances.

To assist the user in navigating their budgeted posts, a tree structure is shown in the left part of the main frame. The tree will show the user's financial categories and the corresponding posts. The user can use the tree to view only the categories that he/she is interested of by clicking in the tree structure.

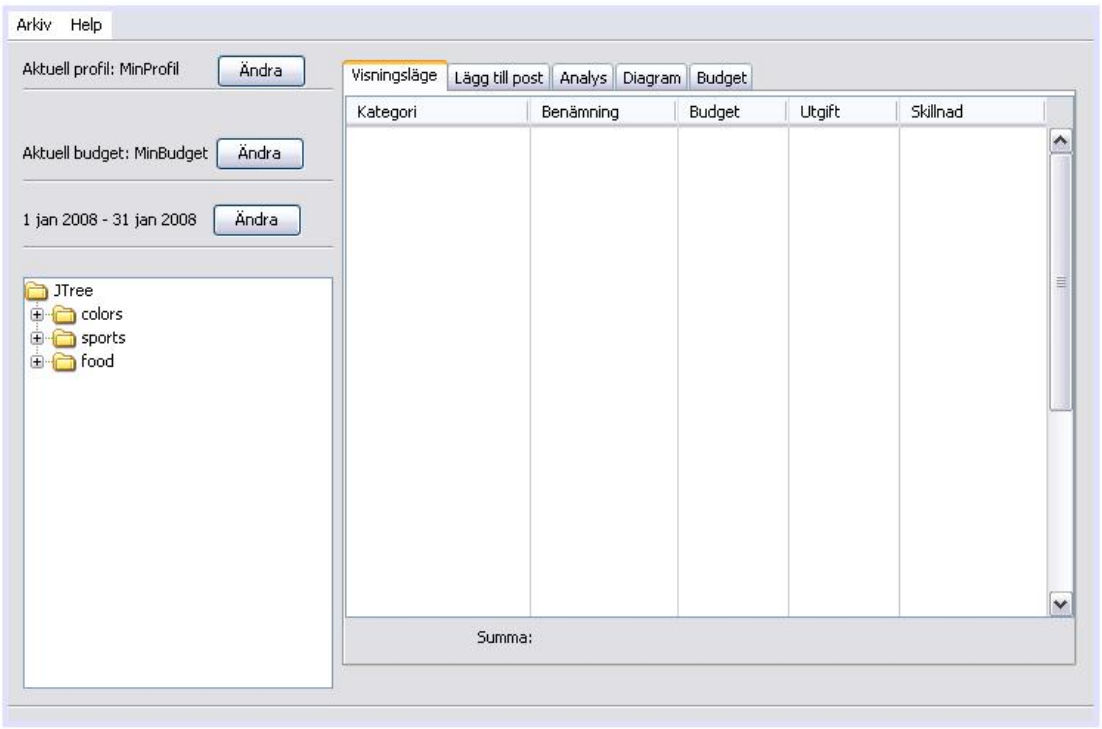
In the "Visningsläge" tab, a table is displayed, showing a simple analysis of a budget.

In the "Lägg till post" tab, the user can add their expenses and incomes to their profile.

In the "Analys" tab, the user can set a number of parameters for "advanced" analysis of their finances. Here the user can choose to analyze depending on dates, categories and/or subcategory.

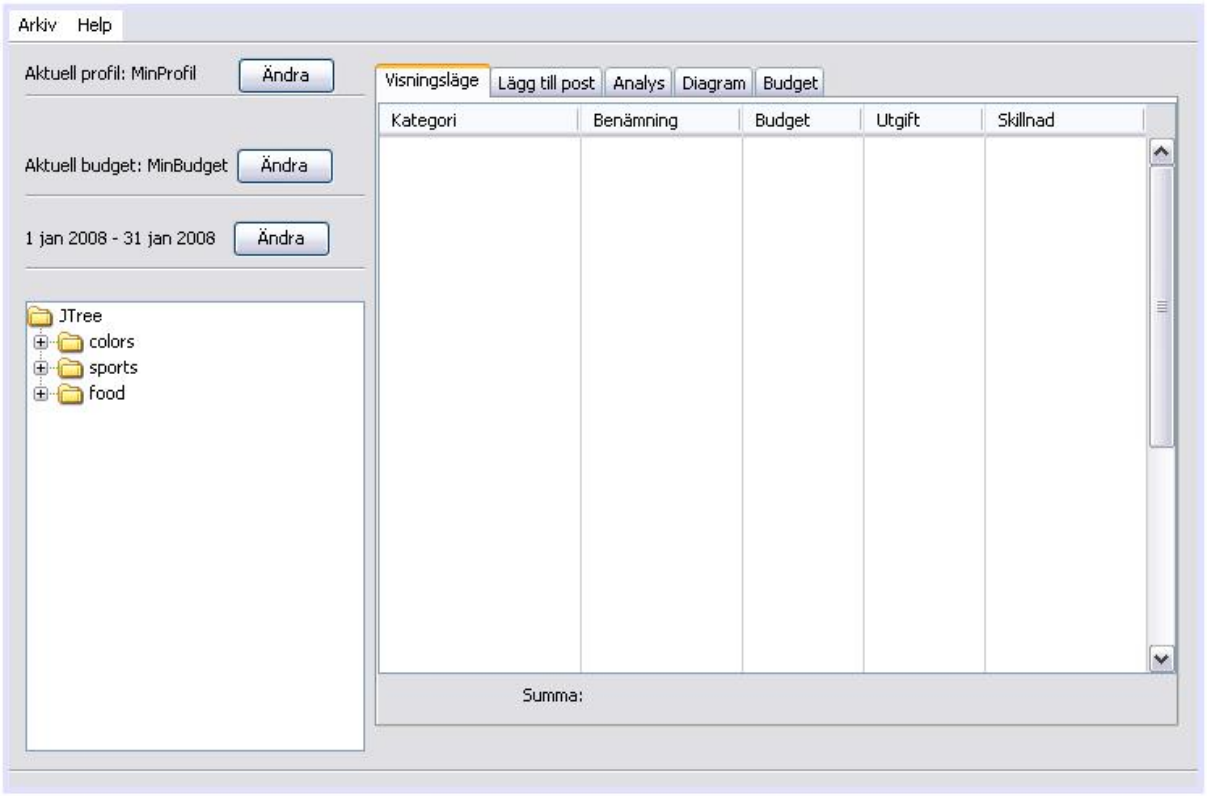
In the "Budget" tab, the user can create, modify and manage their budgets.

The time setting to the left in the main frame will define the time period for display in "Visningsläge".



4.2 Screen shots

JPanel *mainPanel*
title: ”Ivanhoe”



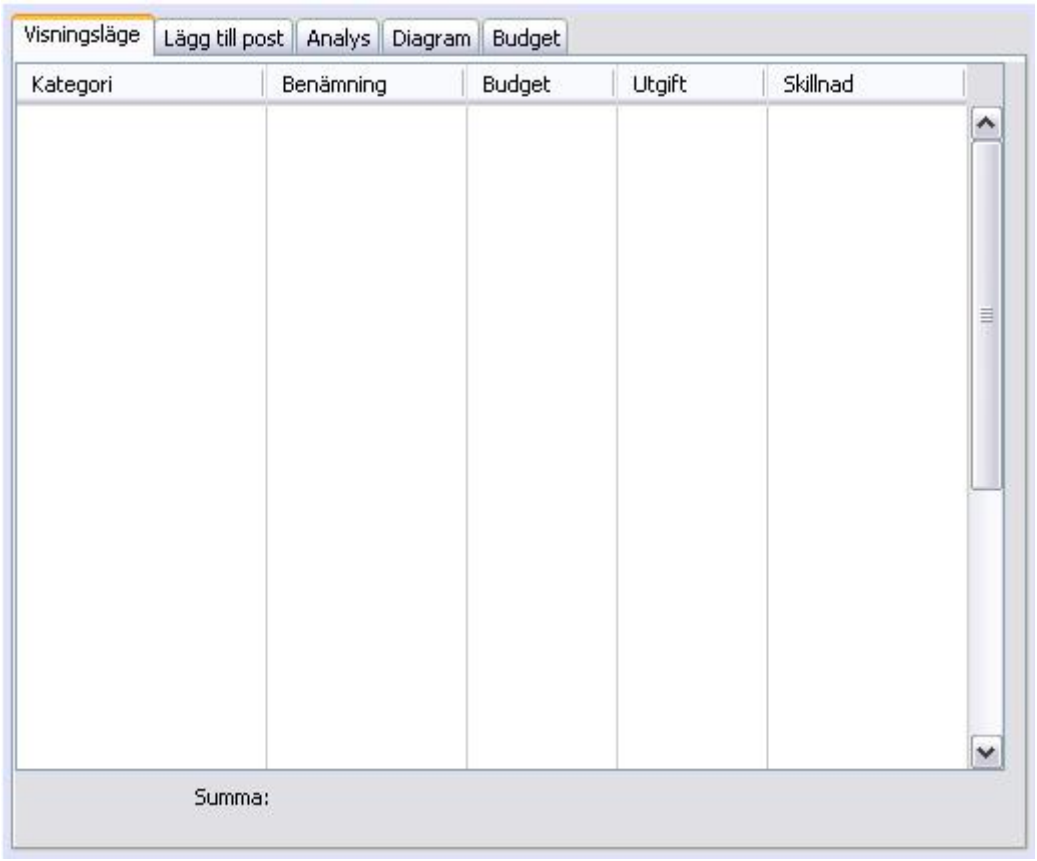
The main view of the program and is called upon system startup.

Fields:
JMenuBar *vMenuBar*
JTree *vMainTree*
JTabbedPane *vMainTabs*

Controls:
JButton *vChangeProfile* “Ändra” calls *chooseProfileDialog*
JButton *vChangeBudget* “Ändra” calls *chooseBudgetDialog*
JButton *vChangeDate* “Ändra” calls *chooseDateDialog*

References to Requirements Document: All points.

JPanel *showPanel*
title: "Visningsläge"



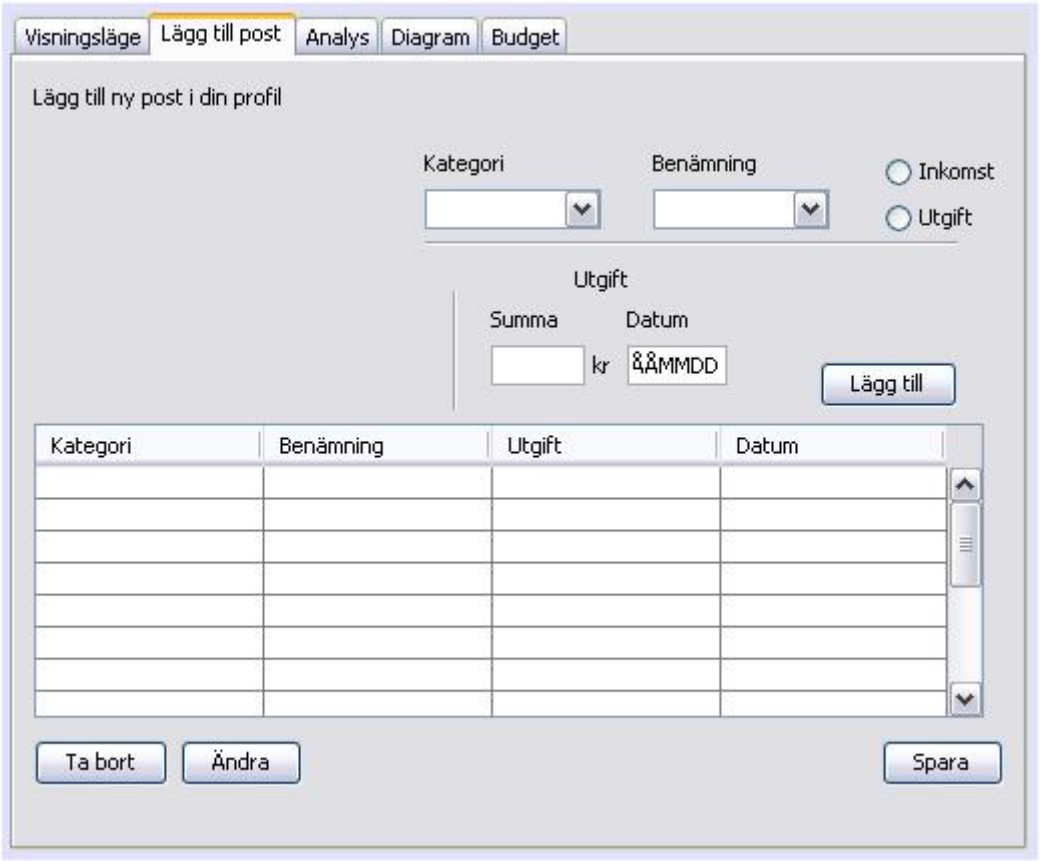
Choosen from the mainPanel

Fields:
JTable *vShowTable*

Controls:
-

References to Requirements Document: 4.1.1.6, 4.1.1.7

JPanel *addPostPanel*
title*:* "Lägg till post"



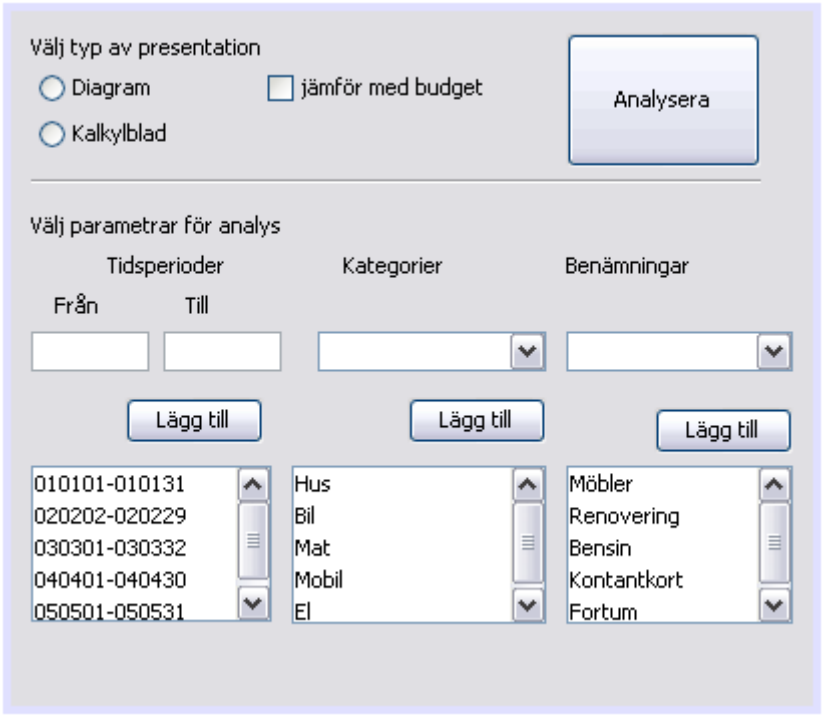
Chosen in the mainPanel.

Fields:
JComboBox *pSubCategory* “Benämning”
JComboBox *pCategory* “Kategori”
JRadioButton *pIncome* “Inkomst”
JRadioButton *pExpense* “Utgift”
JTable *pTable*

Controls:
JButton *pAdd* "Lägg till" calls *addPost*()
JButton *pSave* "Spara" calls *save*()
JButton *pRemote* "Ta bort" calls *remove*()
JButton *pChange* "Ändra" calls *change*()

References to Requirements Document: 4.1.1.4, 4.1.1.5

JPanel *analysisPanel*
title: "Analys"



Chosen in the mainPanel,

Fields:
JTextField *aTimePeriodStart* “Från”
JTextField *aTimePeriodStop* “Till”
JComboBox *aCategory* “Kategorier”
JComboBox *aSubCategory* “Benämningar”
JRadioButton *aDiagram* “Diagram”
JRadioButton *aCalculus* “Kalkylblad”
JCheckBox *aCompare* “Jämför med budget”
JList *aTimePeriodList*
JList *aCategoryList*
JList *aSubCategoryList*

Controls:
JButton *aAnalysis* “Analysera” calls *dataAnalysis()*
JButton *aCatergoryButton* “Lägg till” calls *addCategory*()
JButton *aSubCategoryButton* “Lägg till” calls *addSubCategory*()
JButton *aTimePeriodButton* “Lägg till” calls *addTimePeriod*()

References to Requirements Document: 4.1.1.7

JPanel *showDiagramPanel*
title: "Diagram"

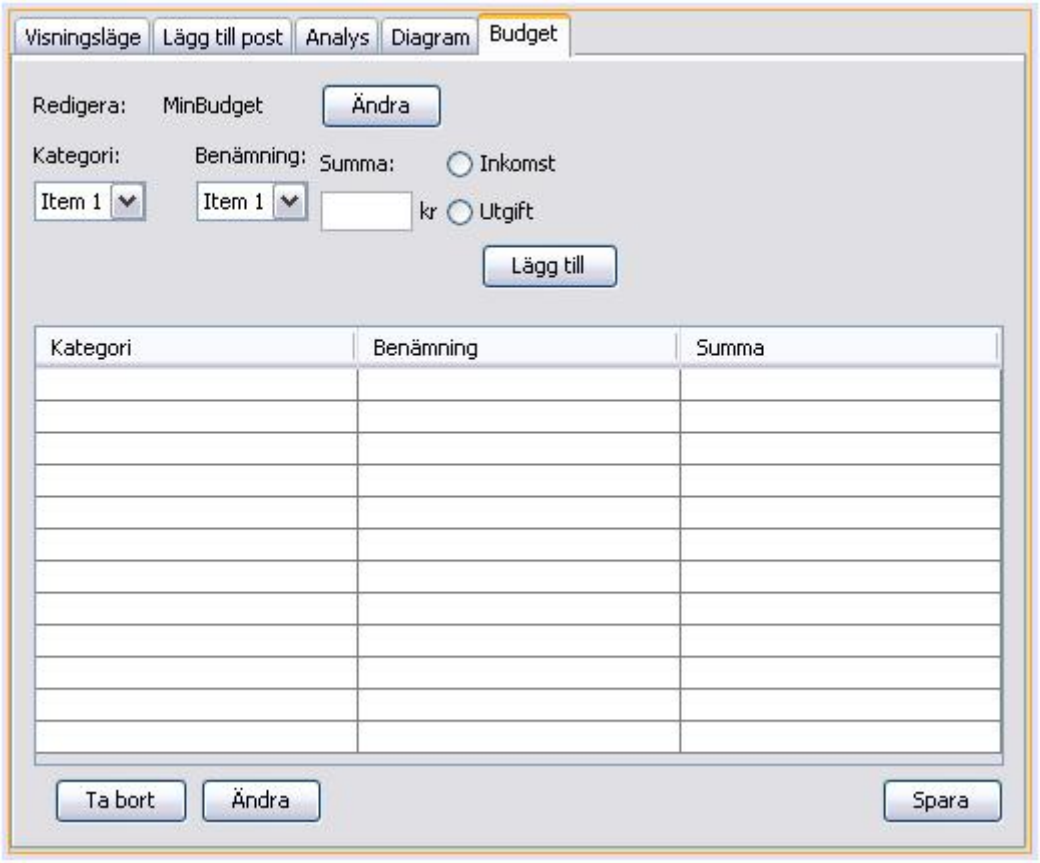
Choosen in the mainPanel.

Fields:
JPanel *dDiagramPanel*

Controls:
-

References to Requirements Document: 4.1.1.7.1

JPanel *createBudgetPanel*
title: "Redigera"



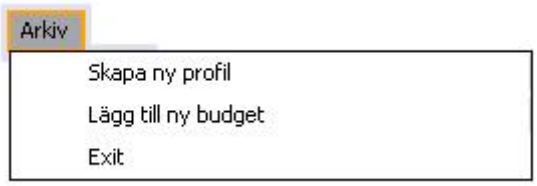
Choosen in the mainPanel.

Fields:
JTextField *bSum* "Summa"
JCombobox *bCategory* "Kategori"
JCombobox *bSubCategory* "Benämning"
JTable *bTable*
RadioButton *bIncome* "Inkomst"
RadioButton *bExpense* "Utgift"

Controls:
JButton *bChange* “Ändra” calls *chooseBudgetDialog()*
JButton *bSave* “Spara” calls *saveBudget()*
JButton *bRemove* "Ta bort" calls *removeEntry()*
JButton *bEdit* "Ändra" calls *updateEntry()*

References to Requirements Document: 4.1.1.2, 4.1.1.3, 4.1.1.5

JMenu *FileMenu*
title*: "*Arkiv*"*



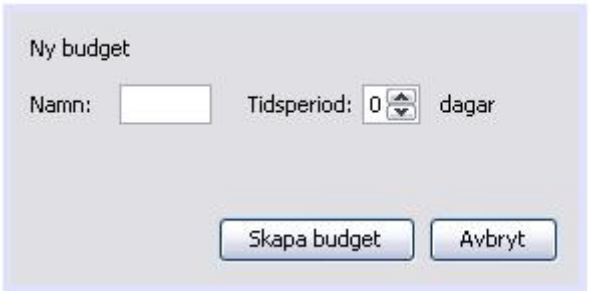
Choosen from the menuBar

Fields:
JMenu *fileMenu "*Arkiv*"*

Controls:
JMenuItem *createNewProfile* "Skapa ny profil" calls *createNewProfileDialog()*
JMenuItem *addNewBudget* "Lägg till ny budget" calls *addNewBudget()*
JMenuItem *exit* "Exit" calls the *exit(),* quits the application

References to Requirements Document: 4.1.1.1, 4.1.1.2

JDialog C*reateNewBudgetDialog*
title*:* "Skapa ny budget"



Choosen from the mainPanel

Fields:
JTextField *budgetName*
JSpinner *daySpinner* “Tidsperiod”

Controls:
JButton *createBudget* "Skapa budget" calls *createBudget()*
JButton *cancel* "Avbryt" quits the dialog

References to Requirements Document: 4.1.1.2

JDialog C*reateProfileDialog*
Title: ”Skapa ny profil”

Called from createNewProfile

Fields:
JTextField *profileNameField*

Controls:
JButton *ok* “Ok” calls *createProfile()* and closes the dialog.
JButton *cancel* “Avbryt” closes the dialog.

Reference to Requirements Document: 4.1.1.1

JDialog *chooseBudgetDialog*
title: "Välj budget"

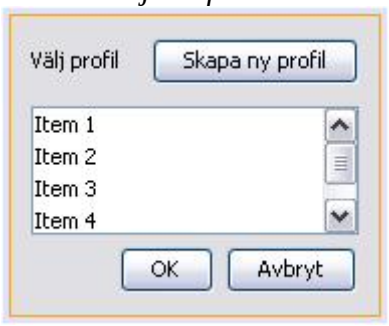
Choosen from the menuBar, and bBudget

Fields:
JList *budgetList*

Controls:
JButton *newBudget* "Skapa ny budget" calls *createNewBudgetDialog()*
JButton *ok* "Ok" calls *createBudgetPanel()*
JButton *cancel* "Avbryt" exits dialog

References to Requirements Document: 4.1.1.2, 4.1.1.3

JDialog *chooseProfileDialog*
Title: *"Välj Profil"*



Called from the button createNewProfile.

Fields:
JList *profileList*

Controls:
JButton *createProfile* "Skapa ny profil" calls *createProfileDialog()*
JButton *ok* "Ok" calls *createNewProfile()*
JButton *cancel* "Avbryt" closes the dialog

Reference to Requirements Document: 4.1.1.1

JDialog C*hooseDateDialog*
title: "*Välj tidsperiod för visning"*

Called from JPane "visningsläge" button "ändra"

Fields:
JTextField *startDate* "Från"
JTextField *endDate* "Till"

Controls:
JButton *ok* "Ok" calls *changeTimePeriod()*
JButton *cancel* "Avbryt" quits the dialog

References to Requirements Document: 4.1.1.6, 4.1.1.7

4.2.1 Requirements Cross reference

Cross reference between the GUI components and the User Requirements from the Ivanhoe Requirements Document.

Requirement:	**Reference:**
4.1.1.1	**4.2 "Skapa ny profil". "Välj profil"**
4.1.1.2	**4.2 "Skapa ny budget"**
4.1.1.3	**4.2 "Redigera"**
4.1.1.4	**4.2 "Lägg till post"**
4.1.1.5	**4.2 "Lägg till post", "Redigera"**
4.1.1.6	**4.2 "Visningsläge", "Välj tidsperiod för visning"**
4.1.1.7	**4.2 "Analys"**
4.1.1.7.1	**4.2 "Diagram"**
4.1.1.7.2	**4.2 "Diagram"**

5. Design Details

5.1 Class Responsibility Collaborator (CRC) Cards

Class Profile

Responsibilities	Collaborators
Create Budgets	Budget
Create DatedEvents	DatedEvent
Keep track of Budgets	Budget
Keep track of DatedEvents	DatedEvent
Initiate analysis	AnalyseData

Class AnalyseData

Responsibilities	Collaborators
Create list of EconomicEvents	Profile
Create list of DatedEvents	Profile
Calculate difference between EconomiEvents and DatedEvents	Profile

Class Budget

Responsibilities	Collaborators
Keep track of EconomicEvent	EconomicEvent
Create EconomicEvents	EconomicEvent
Check budget deficit	

Class EconomicEvent

Responsibilities	Collaborators
Keep track of data for EconomicEvents: sum, category, name and Boolean for income/expense	

Class DatedEvent

Responsibilities	Collaborators
Keep track of data for a date and all the data in an EconomicEvent.	

Class FileSystem

Responsibilities	Collaborators
Read data from file given a filename	
Write data to a file	

Class SystemController

Responsibilities	Collaborators
Allow user to create a Profile	Profile
Allow user to create a Budget	Budget
Allow user to add EconomicEvent to a Budget	Budget, EconomicEvent
Allow user to add DatedEvents to Profile	Profile, DatedEvent
Allow user to delete EconomicEvents from a Budget	Budget
Allow user to edit EconomicEvents in a Budget	Budget, EconomicEvent
Allow user to edit DatedEvent in a Profile	Profile, DatedEvent
Allow user to analyse data	Profile, GUI

Allow user to save Profile to file	FileSystem, Profile
Allow user to read Profile from file	FileSystem, Profile
Allow user to change active Budget	GUI, Budget, Profile
Allow user to change active Profile	GUI, FileSystem, Profile

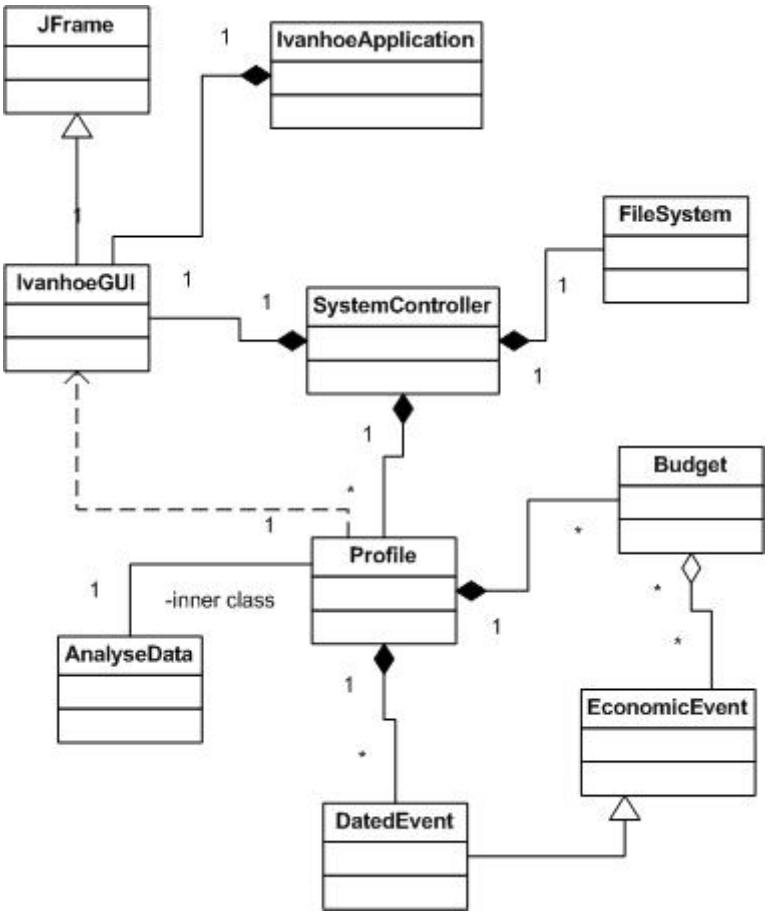
Class IvanhoeGUI

Responsibilities	**Collaborators**
Present data in the form of graphs and tables	AnalyseData
Always present an updated view of Profile	Profile

Class IvanhoeApplication

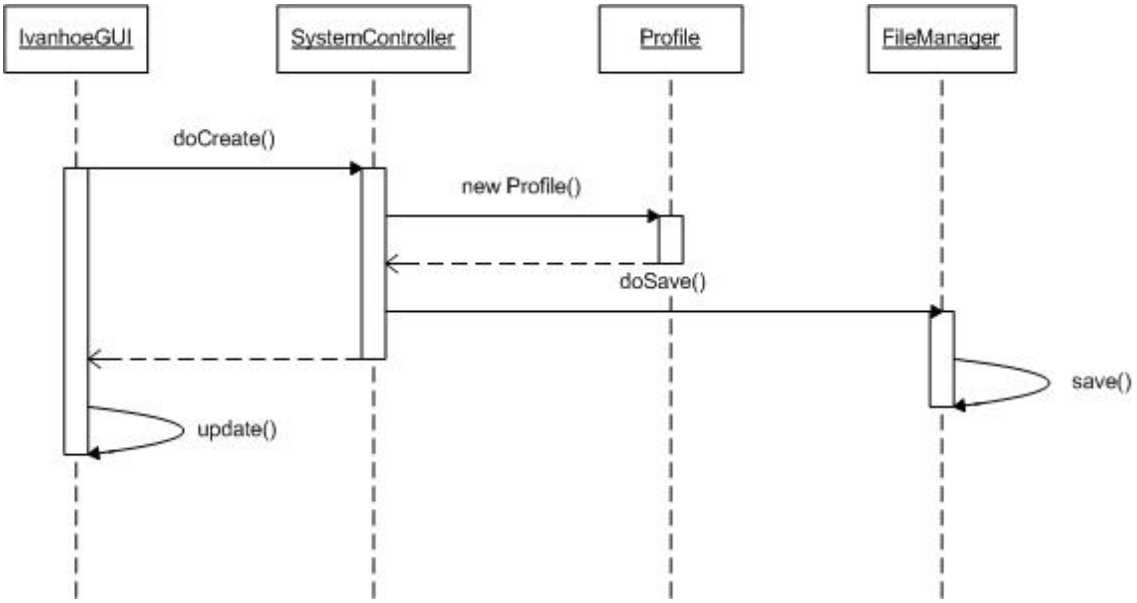
Responsibilities	**Collaborators**
Initiate system	GUI
Exit system	

5.2 Class Diagram



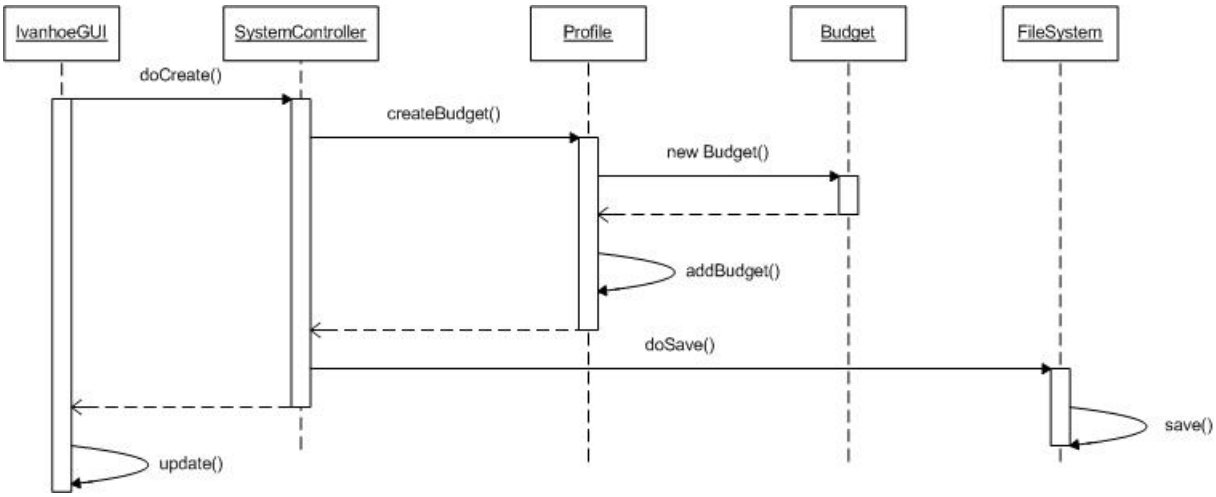
5.3 Interaction Diagrams

Diagram 1 – Create Profile



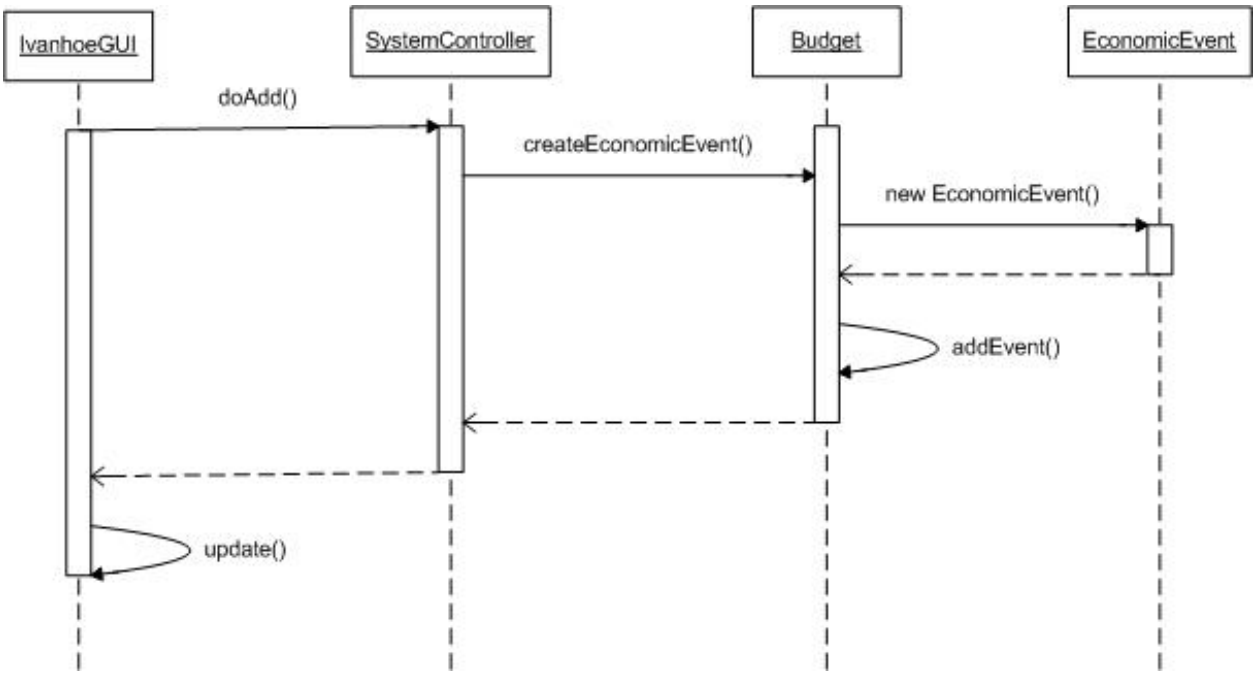
- The user clicks a button on the GUI provided by IvanhoeGUI and the command is sent to SystemController.
- SystemController creates a new Profile and saves the profile by calling the save-method in FileManager.
- The FileManager saves the Profile by rewriting the XML-file.
- When the Profile has been saved, the IvanhoeGUI is updated and shows the Profile to the user.

Diagram 2 – Create Budget



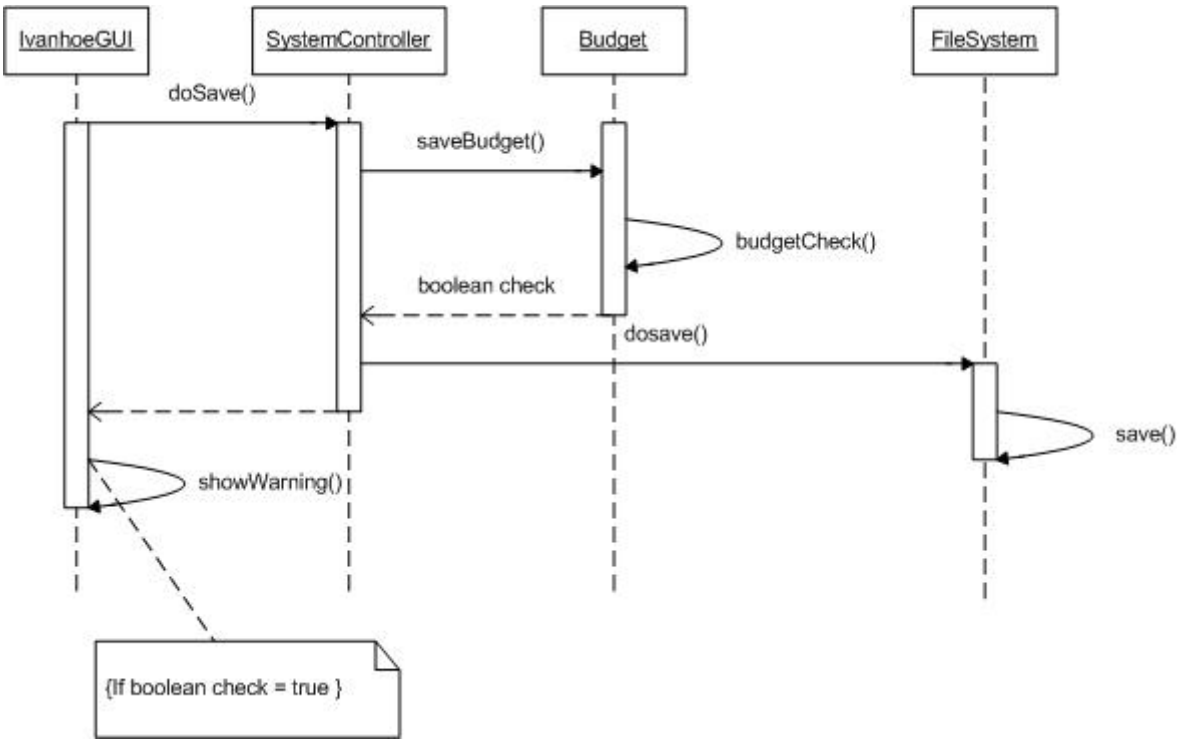
- The user clicks a button to create a new Budget and inputs the data needed.
- The command is sent to SystemController that calls the method createBudget() in Profile.
- The Profile creates a new Budget.
- When the Budget is created, it is added to the Profile.
- The SystemController saves the edited Profile by calling the save-method in FileManager.
- The FileManager saves the Profile by rewriting the XML-file.
- When the Profile has been saved, the IvanhoeGUI is updated and shows the Budget to the user.

Diagram 3 – Edit Active Budget



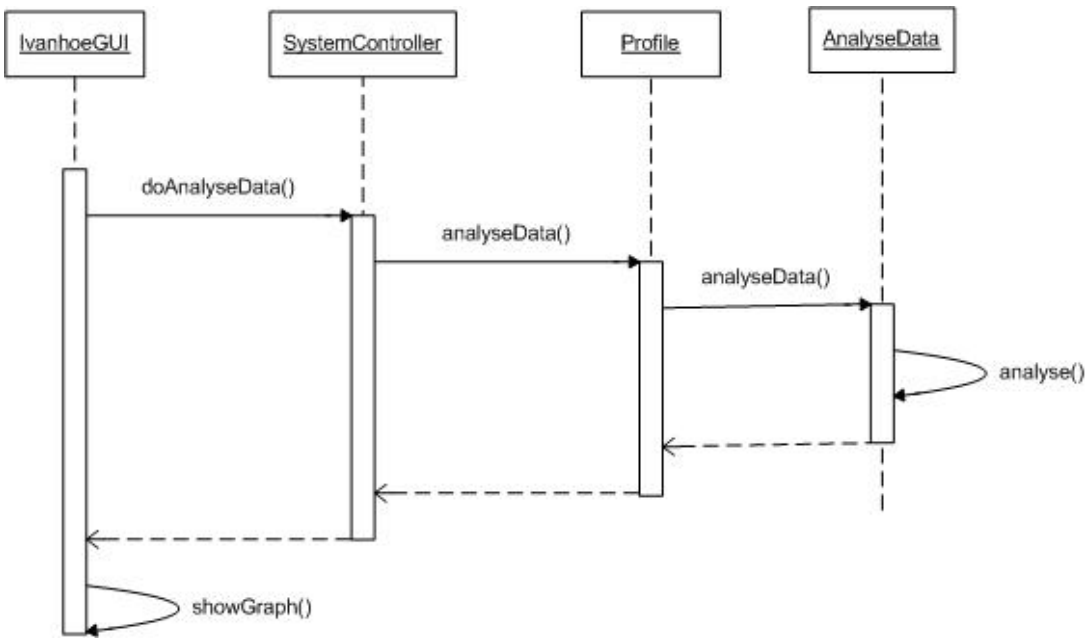
- The user clicks a button to add an EconomicEvent to the active Budget after inputting the required data.
- The command is sent to SystemController that calls the method createEconomicEvent() in Budget.
- The Budget creates a new EconomicEvent.
- When the EconomicEvent is created, it is added to the Budget.
- When the EconomicEvent has been added to the Budget, the IvanhoeGUI is updated and shows the Budget to the user.

Diagram 4 – Save Active Budget



- The user clicks a button to save an edited budget.
- The command is sent to SystemController that calls the method saveBudget() in Budget.
- The Budget checks if there is a budget deficit and returns a Boolean.
- The SystemController saves the edited Profile by calling the save-method in FileManager.
- The FileManager saves the Profile by rewriting the XML-file.
- If there is a budget deficit, the IvanhoeGUI shows a dialog warning the user. The dialog is closed when the user clicks the button “Ok”.
- The IvanhoeGUI shows the Budget to the user.

Diagram 5 – Create Graph



- The user clicks a button to create a graph and inputs the data needed.
- The command is sent to SystemController that calls the method analyseData() in Profile.
- The Profile, in turn, calls the method analyseData() in AnalyseData.
- The AnalyseData analyse the data stored in the Profile according to the user's input and returns the result.
- The IvanhoeGUI displays the data as a graph.

5.5 Detailed Design

The detailed design shows the classes and their variables and methods as well as a detailed description of each method in the class.

5.5.1 AnalyseData

Class AnalyseData
+compareWithBudget(Date timeperiod, Budget budget): ArrayList +produceDiagramData(ArrayList timeperiods, ArrayList categories, ArrayList subcategories, Budget budget, ArrayList datedEventCollection, Boolean compareWithBudget): ArrayList

Method: compareWithBudget()	
Parameters:	Date timeperiod, Budget budget
Return value:	ArrayList
Description:	Computes and structures data for the table in the “Visningsläge” tab in the GUI.
Called by:	Profile.processShowData()
Calls:	-
Reference to RD:	6.1.1.6, 6.1.1.6.1, 6.1.1.6.2, 6.1.1.6.2.1

Method: produceDiagramData()	
Parameters:	ArrayList timeperiods, ArrayList categories ArrayList subcategories, Budget budget, ArrayList datedEventCollection, Boolean compareWithBudget
Return value:	ArrayList
Description:	Computes and structures data for the table or diagram in the “Diagram” tab in the GUI.
Called by:	Profile.processDiagramData()
Calls:	-
Reference to RD:	6.1.1.7

5.5.2 Profile

Class Profile
-budgetCollection: ArrayList<Budget> -datedEventCollection: ArrayList<DatedEvent> -activeBudget Budget -activeDatedEvent DatedEvent -name: String -file: File -changedSinceLastSave: Boolean
+Profile(String name) +getFile(): File +setFile(File file) +createBudget(String name) +editBudget(String newName) +addEconomicEventToBudget(String category, String subCategory, nit sum, String type) +addDatedEvent(String category, String subCategory, nit sum, String type, Date timperiod) +deleteDatedEvent() +deleteEconomicEvent() +editDatedEvent(int newSum, String newType, String newCategory, String newSubCategory, Date newTiemperiod) +editEconomicEvent(int newSum, String newType, String newCategory, String newSubCategory) +getChangedSinceLastSave(): Boolean +setChangedSinceLastSave(Boolean arg) +processShowData(Date timeperiod): ArrayList +processDiagramData(ArrayList timeperiods, ArrayList categories, ArrayList subCategories, Boolean compareWithBudget ArrayList

Method: getFile()	
Parameters:	-
Return value:	File
Description:	Get the File this Profile was most recently read from or saved to.
Called by:	FileSystem.saveFile(), FileSystem.readFile()
Calls:	-
Reference to RD:	-

Method: setFile()	
Parameters:	File file
Return value:	-
Description:	Set the File this Profile was most recently read from or saved to
Called by:	FileSystem.saveFile(), FileSystem.readFile()
Calls:	-
Reference to RD:	-

Method: createBudget()	
Parameters:	String name
Return value:	-
Description:	Creates a budget and adds it to the budgetCollection
Called by:	SystemController.doCreateBudget()
Calls:	budget()
Reference to RD:	6.1.1.2, 6.1.1.2.1

Method: editBudget()	
Parameters:	String newName
Return value:	-
Description:	Changes the name of a budget
Called by:	SystemController.editBudget()
Calls:	Budget.setName()
Reference to RD:	6.1.1.3

Method: addEconomicEventToBudget()	
Parameters:	String category, String subCategory, int sum, String type
Return value:	-
Description:	Adds an EconomicEvent to a budget
Called by:	SystemController.doAddEconomicEvent()
Calls:	Budget.addEconomicEvent()
Reference to RD:	6.1.1.2.1, 6.1.1.4

Method: addDatedEvent()	
Parameters:	String category, String subCategory, nit sum, String type, Date timperiod
Return value:	-
Description:	Adds a DatedEvent to the profile
Called by:	SystemController.doAddDatedEvent()
Calls:	DatedEvent()
Reference to RD:	6.1.1.4.1

Method: deleteDatedEvent()	
Parameters:	-
Return value:	-
Description:	Deletes a DatedEvent from the profile
Called by:	SystemController.doDeleteDatedEvent()
Calls:	-
Reference to RD:	-

Method: deleteEconomicEvent()	
Parameters:	-
Return value:	-
Description:	Deletes an EconomicEvent from the profile's budget
Called by:	SystemController.doDeleteEconomicEvent()
Calls:	Budget.deleteEconomicEvent()
Reference to RD:	6.1.1.3, 6.1.1.3.1

Method: editDatedEvent()	
Parameters:	int newSum, String newType, String newCategory, String newSubCategory, Date newTimeperiod
Return value:	-
Description:	Edits a DatedEvent in the profile
Called by:	SystemController.doEditDatedEvent()
Calls:	DatedEvent.setDate(), DatedEvent.setSum(), DatedEvent.setType(), DatedEvent.setSubcategory(), DatedEvent.setCategory()
Reference to RD:	-

Method: editEconomicEvent()	
Parameters:	int newSum, String newType, String newCategory, String newSubCategory
Return value:	-
Description:	Edits an conomicEventEvent in the profile's budget
Called by:	SystemController.doEditEconomicEvent()
Calls:	Budget.editEconomicEvent()
Reference to RD:	6.1.1.3

Method: getChangedSinceLastSave()	
Parameters:	-
Return value:	Boolean
Description:	Find out whether this profile has been changed since last open / saved.
Called by:	SystemController.doSave()
Calls:	-
Reference to RD:	-

Method: setChangedSinceLastSave()	
Parameters:	Boolean changedSinceLastSaved
Return value:	-
Description:	Record a change in the saved status of this profile.
Called by:	SystemController.doSave()
Calls:	-
Reference to RD:	-

Method: processShowData()	
Parameters:	Date timeperiod
Return value:	ArrayList
Description:	Requests class AnalyseData to process data to be shown in the “Visningsläge” tab in the GUI.
Called by:	SystemController.doShowData()
Calls:	AnalyseData.compareWithBudget()
Reference to RD:	6.1.1.6.1

Method: processDiagramData()	
Parameters:	ArrayList timeperiods, ArrayList categories, ArrayList subCategories, Boolean compareWithBudget
Return value:	ArrayList
Description:	Requests class AnalyseData to process data to be shown in the “Diagram” tab in the GUI
Called by:	SystemController.doAnalyseData()
Calls:	AnalyseData.produceDiagramData()
Reference to RD:	6.1.1.7

5.5.3 Economic event

Class EconomicEvent
-type:string -category:String -subcategory:String -sum:int
+EconomicEvent(type, category, subcategory, value) +getType():String +getCategory():String +getSubcategory():String +getSum():int +setType(String type): +setCategory(String category): +setSubcategory(String subCategory): +setSum(int sum):

Method: getType()	
Parameters:	-
Return value:	String
Description:	Returns the type for the EconomicEvent
Called by:	Budget.sortByType()
Calls:	-
Reference to RD:	-

Method: getCategory()	
Parameters:	-
Return value:	String
Description:	Returns the category for the EconomicEvent
Called by:	Budget.sortByCategory()
Calls:	-
Reference to RD:	-

Method: getSubcategory()	
Parameters:	-
Return value:	String
Description:	Returns the subcategory for the EconomicEvent
Called by:	Budget.sortBySubcategory()
Calls:	-
Reference to RD:	-

Method: getSum()	
Parameters:	-
Return value:	Int
Description:	Returns the sum for the EconomicEvent
Called by:	Budget.sortBySum()
Calls:	-
Reference to RD:	-

Method: setType()	
Parameters:	String type
Return value:	-
Description:	Sets the type for an EconomicEvent.
Called by:	
Calls:	-
Reference to RD:	6.1.1.2.1.2

Method: setCategory()	
Parameters:	String category
Return value:	-
Description:	Sets the Category for an EconomicEvent.
Called by:	
Calls:	-
Reference to RD:	6.1.1.2.1.1

Method: setSubcategory()	
Parameters:	String subcategory
Return value:	-
Description:	Sets the subcategory for an EconomicEvent.
Called by:	
Calls:	-
Reference to RD:	6.1.1.2.1.1

Method: setSum()	
Parameters:	Int sum
Return value:	-
Description:	Sets the sum for an EconomicEvent.
Called by:	
Calls:	-
Reference to RD:	6.1.1.2.1.1

5.5.4 SystemController

Class SystemController
-ivanhoeGUI: IvanhoeGUI -fileSystem: FileSystem -profile: Profile
+SystemController(FileSystem filesystem): +doSave(IvanhoeGUI gui) +doQuit(IvanhoeGUI gui) +doCreateProfile(IvanhoeGUI gui) +doDeleteProfile(IvanhoeGUI gui) +doCreateBudget(IvanhoeGUI gui) +doEditBudget(IvanhoeGUI gui) +doAddEconomicEvent(IvanhoeGUI gui) +doAddDatedEvent(IvanhoeGUI gui) +doDeleteDatedEvent(IvanhoeGUI gui) +doDeleteEconomicEvent(IvanhoeGUI gui) +doEditEconomicEvent(IvanhoeGUI gui) +doEditDatedEvent(IvanhoeGUI gui) +doAnalyseData(IvanhoeGUI gui) +doOpenProfile(IvanhoeGUI gui)

Method: doSave()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Saves the active profile to file
Called by:	IvanhoeGUI
Calls:	FileSystem.save()
Reference to RD:	-

Method: doQuit()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Quits the application
Called by:	IvanhoeGUI
Calls:	IvanhoeApplication.quitApplication()
Reference to RD:	-

Method: doCreateProfile()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Creates a profile
Called by:	IvanhoeGUI
Calls:	profile()
Reference to RD:	-

Method: doDeleteProfile()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Deletes the active profile
Called by:	IvanhoeGUI
Calls:	FileSystem.delete()
Reference to RD:	-

Method: doCreateBudget()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Creates a Budget
Called by:	IvanhoeGUI
Calls:	Profile.createBudget()
Reference to RD:	-

Method: doEditBudget()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Edits an active Budget's name
Called by:	IvanhoeGUI
Calls:	Profile.editBudget()
Reference to RD:	-

Method: doAddEconomicEvent()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Adds an EconomicEvent to the active budget
Called by:	IvanhoeGUI
Calls:	Profile.addEconomicEventToBudget()
Reference to RD:	-

Method: doAddDatedEvent()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Adds a DatedEvent to the active profile
Called by:	IvanhoeGUI
Calls:	Profile.addDatedEvent()
Reference to RD:	-

Method: doDeleteDatedEvent()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Delets a DatedEvent from the profile
Called by:	IvanhoeGUI
Calls:	Profile.deleteDatedEvent()
Reference to RD:	-

Method: doDeleteEconomicEvent()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Edits an EconomicEvent in the profile's budget
Called by:	IvanhoeGUI
Calls:	Profile.deleteEconomicEvent()
Reference to RD:	-

Method: doEditEconomicEvent()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Edits an EconomicEvent
Called by:	IvanhoeGUI
Calls:	Profile.editEconomicEvent()
Reference to RD:	-

Method: doEditDatedEvent()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Edits a DatedEvent in the profile.
Called by:	IvanhoeGUI
Calls:	Profile.editDatedEvent()
Reference to RD:	-

Method: doAnalyseData()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Analyses the data from the given parameters and displays them as a table or diagram
Called by:	IvanhoeGUI
Calls:	Profile.processDiagramData()
Reference to RD:	-

Method: doShowData()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Shows the data from the profile in a table
Called by:	IvanhoeGUI
Calls:	Profile.processShowData()
Reference to RD:	-

Method: doOpenProfile()	
Parameters:	IvanhoeGUI gui
Return value:	-
Description:	Opens the selected profile
Called by:	IvanhoeGUI
Calls:	FileSystem.readFile()
Reference to RD:	-

5.5.5 Budget

Class Budget
-economicEventCollection: ArrayList <EconomicEvent> -name: String -timespan:int -activeEconomicEvent: EconomicEvent
+Budget(name, timespan) +addEconomicEvent(String type, String category, String subcategory, int sum) +editEconomicEvent(String type, String category, String subcategory, int sum) +deleteEconomicEvent() +setName() +sortByCateogry() +sortBySum() +sortByName() +sortByType() +sortBySum()

Method: addEconomicEvent()	
Parameters:	String type, String category, String subcategory, int sum
Return value:	-
Description:	Creates and adds an EconomicEvent to the economicEventCollection
Called by:	Profile.addEconomicEventToBudget()
Calls:	economicEvent()
Reference to RD:	6.1.1.2.1

Method: editEconomicEvent()	
Parameters:	String newType, String newCategory, String newSubcategory, int newSum
Return value:	-
Description:	Edits an EconomicEvent in the economicEventCollection
Called by:	Profile.editEconomicEvent()
Calls:	EconomicEvent.setType(), EconomicEvent.setCategory(), EconomicEvent.setSubcategory(), EconomicEvent.setSum(),
Reference to RD:	6.1.1.3, 6.1.14

Method: deleteEconomicEvent()	
Parameters:	-
Return value:	-
Description:	Deletes an EconomicEvent from the economicEventCollection
Called by:	Profile.deleteEconomicEvent()
Calls:	-
Reference to RD:	6.1.1.3

Method: setName()	
Parameters:	String newName
Return value:	-
Description:	Changes the name of the Budget
Called by:	Profile.editBudget()
Calls:	-
Reference to RD:	-

Method: sortByCategory()	
Parameters:	-
Return value:	-
Description:	Sorts the EconomicEvents in the economicEventCollection by category
Called by:	-
Calls:	EconomicEvent.getCategory()
Reference to RD:	-

Method: sortBySum()	
Parameters:	-
Return value:	-
Description:	Sorts the EconomicEvents in the economicEventCollection by sum
Called by:	-
Calls:	EconomicEvent.getSum()
Reference to RD:	-

Method: sortBySubcategory()	
Parameters:	-
Return value:	-
Description:	Sorts the EconomicEvents in the economicEventCollection by subcategory
Called by:	-
Calls:	EconomicEvent.getSubcategory()
Reference to RD:	-

Method: sortByType()	
Parameters:	-
Return value:	-
Description:	Sorts the EconomicEvents in the economicEventCollection by type
Called by:	-
Calls:	EconomicEvent.getType()
Reference to RD:	-

5.5.6 FileSystem

Class FileSystem
+FileSystem() +readFile(File file): Profile +saveFile(File file, Profile profile) +deleteFile(File file)

Method: readFile()	
Parameters:	File file
Return value:	Profile
Description:	Reads a stored file
Called by:	SystemController.doOpenProfile()
Calls:	-
Reference to RD:	-

Method: saveFile()	
Parameters:	File file, Profile profile
Return value:	Profile
Description:	Saves a Profile to a File
Called by:	SystemController.doSave()
Calls:	-
Reference to RD:	-

Method: deleteFile()	
Parameters:	File file
Return value:	-
Description:	Deletes a File
Called by:	SystemController.doDeleteProfile()
Calls:	-
Reference to RD:	-

5.5.7 IvanhoeGUI

Class IvanhoeGUI
-controller: SystemController -profile:Profile -showTable:JTable
+IvanhoeGUI(SystemController controller, Profile profile): +getProfile(): Profile +setProfile(Profile profile) +reportError(String message) +update(Observable o, Object obj) +checkIfSignedInt(Object sum):Boolean +drawDiagram(ArrayList data) +drawTable(ArrayList data)

Method: getProfile()	
Parameters:	-
Return value:	-
Description:	Accessor for the profile this GUI displays.
Called by:	-
Calls:	Profile.getProfile(), setProfile()
Reference to RD:	-

Method: setProfile()	
Parameters:	Profile profile
Return value:	-
Description:	Changes the profile this GUI displays.
Called by:	getProfile()
Calls:	-
Reference to RD:	-

Method: reportError()	
Parameters:	String message
Return value:	-
Description:	Reports an error represented by a string.
Called by:	-
Calls:	-
Reference to RD:	-

Method: update()	
Parameters:	Observable o, Object obj
Return value:	-
Description:	Method required by the Observer interface - update the display in response to any change in the profile
Called by:	-
Calls:	-
Reference to RD:	-

Method: checkIfSigned()	
Parameters:	Object sum
Return value:	Boolean
Description:	Checks whether the sum is positive or negative
Called by:	-
Calls:	-
Reference to RD:	-

Method: drawDiagram()	
Parameters:	ArrayList data
Return value:	-
Description:	Draws a diagram from the given data and displays it
Called by:	-
Calls:	-
Reference to RD:	-

Method: drawTable()	
Parameters:	ArrayList data
Return value:	-
Description:	Draws a table from the given data and displays it.
Called by:	-
Calls:	-
Reference to RD:	-

5.5.8 DatedEvent

Class DatedEvent extends EconomicEvent
-date: Date *Class also includes the variables methods as EconomicEvent.*
+DatedEvent(Date date) +getDate(): Date +setDate(Date date) *Class also includes the same methods as EconomicEvent.*

Method: getDate()	
Parameters:	-
Return value:	Date
Description:	Gets the date for a DatedEvent.
Called by:	Profile??
Calls:	-
Reference to RD:	6.1.1.2.1.1

Method: setDate()	
Parameters:	Date date
Return value:	-
Description:	Sets the date for a DatedEvent.
Called by:	Profile.editDatedEvent()
Calls:	-
Reference to RD:	-

5.5.9 IvanhoeApplication

Class IvanhoeApplication
-fileSystem: FileSystem -controller: SystemController
+main() +quitApplication()

Method: main()	
Parameters:	-
Return value:	-
Description:	Initiates the application
Called by:	-
Calls:	-
Reference to RD:	-

Method: quitApplication()	
Parameters:	-
Return value:	-
Description:	Quits the application
Called by:	SystemController.doQuit()
Calls:	-
Reference to RD:	-

5.5.10 Requirements Cross reference

Cross reference between the detailed classes and the System Requirements from the Ivanhoe Requirements Document.

Requirement:	Reference:
6.1.1.1	5.5.2 Class Profile
6.1.1.2	5.5.5 Class Budget 5.5.2 Profile.createBudget()
6.1.1.2.1	5.5.5 Budget.addEconomicEvent() 5.5.2 Profile.createBudget() 5.5.2 Profile.addEconomicEvent()
6.1.1.2.1.1	5.5.3 Class EconomicEvent, 5.5.8 Class DatedEvent 5.5.3 EconomicEvent.setCategory() 5.5.3 EconomicEvent.setSubCategory() 5.5.3 EconomicEvent.setSum() 5.5.5 Budget.addEconomicEvent()
6.1.1.2.1.2	5.5.3 EconomicEvent.setType() 5.5.8 Class DatedEvent
6.1.1.2.1.3	5.5.7 IvanhoeGUI.checkIfSignedInt()
6.1.1.3	5.5.3 Budget.editEconomicEvent() 5.5.3 Budget.deleteEconomicEvent() 5.5.2 Profile.editBudget() 5.5.2 Profile.deleteEconomicEvent() 5.5.2 Profile.editEconomicEvent()
6.1.1.3.1	5.5.7 Class IvanhoeGUI 5.5.2 Profile.deleteEconomicEvent()
6.1.1.4	5.5.3 Class EconomicEvent, 5.5.8 Class DatedEvent 5.5.2 Profile.addEconomicEvent() 5.5.5 Budget.editEconomicEvent()
6.1.1.4.1	5.5.8 Class DatedEvent 5.5.2 Profile.addDatedEvent()
6.1.1.5	5.5.3 Class EconomicEvent 5.5.8 Class DatedEvent
6.1.1.6	5.5.1 AnalyseData.compareWithBudget()
6.1.1.6.1	5.5.7 Class IvanhoeGui, 5.5.1 AnalyseData.compareWithBudget() 5.5.2 Profile.processShowData()
6.1.1.6.2	5.5.1 AnalyseData.compareWithBudget()
6.1.1.6.2.1	5.5.1 AnalyseData. compareWithBudget ()
6.1.1.7	5.5.2 AnalyseData.produceDiagramData() 5.5.2 Profile.processDiagramData()
6.1.1.7.1	5.5.8 Class DatedEvent. 5.5.1 AnalyseData.produceDiagramData()
6.1.1.7.2	5.5.1 AnalyseData.produceDiagramData()
6.1.1.7.3	5.5.7 IvanhoeGui.drawDigram()
6.1.1.7.4	5.5.X IvanhoeGui.drawTable()

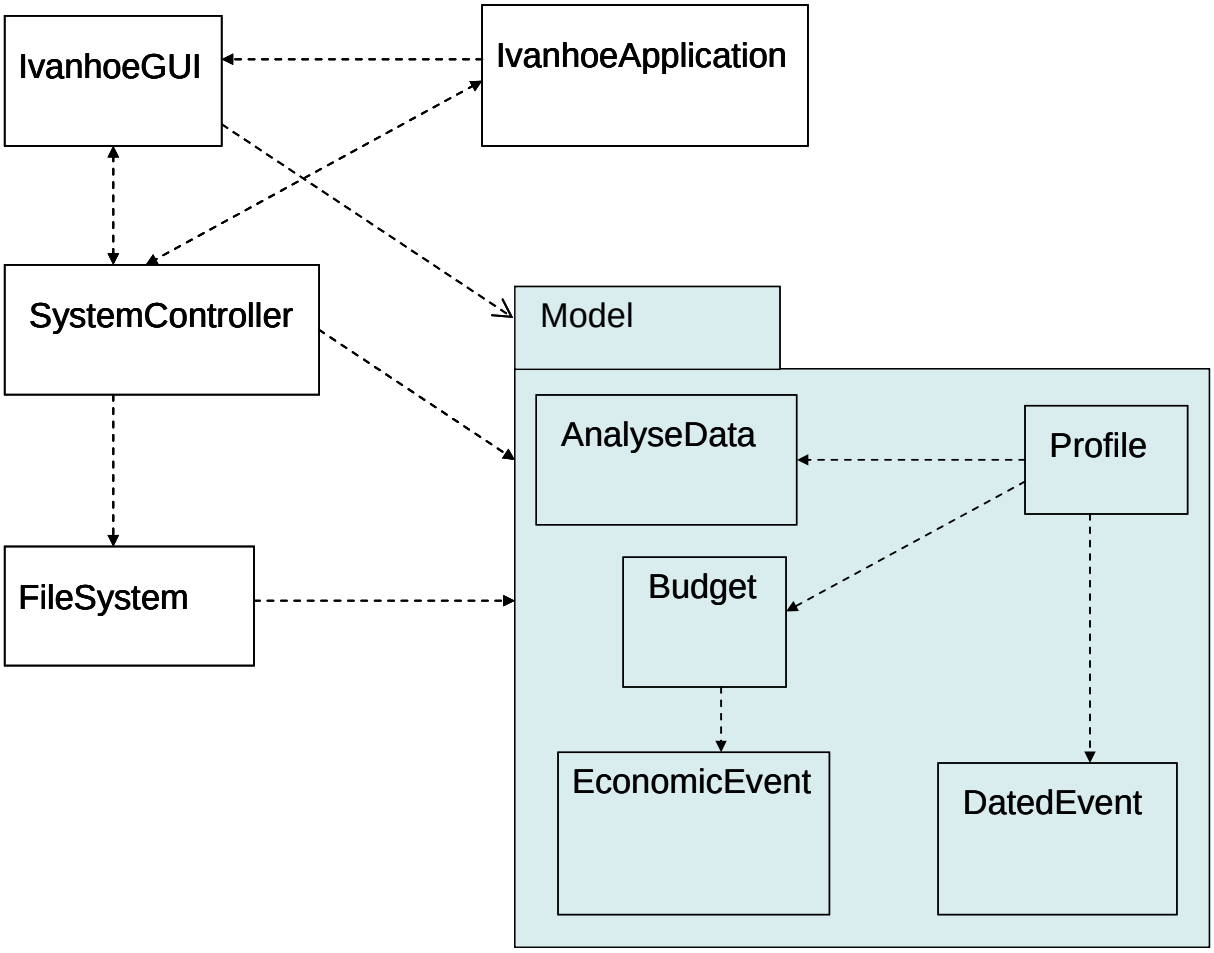
5.6 Package Diagram

There are two levels in the package diagram. The top level consists of:

- The IvanhoeApplication, which instantiates the SystemController and the IvanhoeGUI,
- The FileSystem that controls whether the Model is saved since the last change
- The SystemController, which performs the events that occur in the IvanhoeGUI, creates and manages the Profile, uses the FileSystem to read and write files and is able to quit the application by sending a command to the IvanhoeApplication.
- The IvanhoeGUI that provides the user with a graphical interface and is an observer of the Profile.

The second level is the Model package that consists of:

- The Profile, which uses the inner class AnalyseData to analyse the data and creates and keeps track of DatedEvents and Budgets
- The Budget that creates and keeps track of EconomicEvents.
- The DatedEvent and the EconomicEvent keeps track of data concerning the income or expense.
- The AnalyseData that calculates the difference between EconomiEvents and DatedEvents



6. Functional Test Cases

6.1 Create Profile

Function being tested: Creation of Profile

RD reference: 4.1.1.1

Precondition: -

Use case: UC2 RD 8.2

Input: Profile name.

Expected output: “New profile dialog” is closed. Main window of application displayed. “Aktuell profil:” equals the name of the inputted profile. The profile is saved to file.

Instructions:
1) Choose “Skapa ny profil” in the menu “Arkiv”.
2) Input name in the text field in the dialog.
3) Click the button “Ok”.

6.2 Create Budget

Function being tested: Creation of Budget

RD reference: 4.1.1.2

Precondition: Profile exists.

Use case: UC3 RD 8.3

Input: Budget name and time period.

Expected output: “Create new budget dialog” is closed. Main window of application displayed. “Aktuell budget:” equals the inputted budget. The budget is saved to file.

Instructions:
1) Choose “Skapa ny budget” in the menu “Arkiv”.
2) Input name in the text field in the dialog.
3) Choose number of days in the list in the dialog.
4) Click the button “Skapa budget”.

6.3 Change user profile

Function being tested: Change of active user Profile.

RD reference: 4.1.1.1

Precondition: There is a profile active.

Use case: -

Input: -

Expected output: “Choose profile dialog” is closed. Main window of application displayed. “Aktuell profil:” equals the new chosen profile.

Instructions:

5) Click the button “Ändra” next to “Aktuell profil” in the left main window panel.
6) Select a profile from the list in the “Choose profile dialog”.
7) Click the button “Ok”.

6.4 Analyse data in graph

Function being tested: Analyse data with a graph.

RD reference: 4.1.1.6, 4.1.1.7, 4.1.1.7.1

Precondition: There is a profile active. The profile must contain some financial data such as dated events but also a last one budget if the checkbox “Jämför med budget” is clicked.

Use case: UC4 RD 8.4

Input: Time period and one or many categories and/or subcategories.

Expected output: The tab “Diagram” is displayed. Data is presented in a graph.

Instructions:

1) Choose the “Analys” tab in the main window.
2) Select the option “Diagram”.
3) Click the checkbox “Jämför med budget”*
4) Input a period of time to analyse in the text field “Tidsperioder”.
5) Click the button “Lägg till”.

Repeat step 4 and 5 until satisfied.

6) Choose category or subcategory from the drop down lists “Kategorier” and “Benämningar”.
7) Click the button “Lägg till”.

Repeat step 6 and 7 until satisfied.

8) Click the button “Analysera”.

*Optional

6.5 Analyse data in table

Function being tested: Analyse data with a graph.

RD reference: 4.1.1.6, 4.1.1.7, 4.1.1.7.2

Precondition: There is a profile active. The profile must contain legitimate financial data.

Use case: UC4 RD 8.4

Input: Time period and one or many category and/or subcategories.

Expected output: The tab “Diagram” is displayed. Data is presented in a table.

Instructions:

1) Choose the tab “Analys” in the main window.
2) Select the option “Kalkylblad”.
3) Click the checkbox “Jämför med budget”*
4) Input a period of time to analyse in the text field “Tidsperioder”.
5) Click the button “Lägg till”.

Repeat step 4 and 5 until satisfied.

6) Choose category and subcategory from the drop down lists “Kategorier” and “Benämningar”.
7) Click the button “Lägg till”.

Repeat step 6 and 7 until satisfied.

8) Click the button “Analysera”.

*Optional

6.6 Change time period

Function being tested: Change of the time period.

RD reference: 4.1.1.7

Precondition: There is a profile active.

Use case: -

Input: Start date and end date.

Expected output: “Choose date dialog” is closed. Main window of application displayed. “Aktuell tidsperiod:” equals the inputted time period. The dated events for the chosen time period are displayed.

Instructions:
1) Click the button “Ändra” on the right side of the displayed date in the main view.
2) Input start date and end date in the text fields.
3) Click the button “Ok”.

6.7 Add economic event

Function being tested: Add economic event to budget

RD reference: 4.1.1.3., 4.1.1.5**.**

Precondition: There is a profile active and a budget exists

Use case: UC1 RD 8.1

Input: The sum for the economic event.

Expected output: The economic event has been added to the budget and the modified budget is shown by the system.

Instructions:
1) Click on the tab “Budget”.
2) Choose category from the drop down list “Kategori”.
3) Choose subcategory from the drop down list “Benämning”.
4) Enter the sum in the text field “Summa”.
5) Choose whether the economic event is an income or an expense by clicking one of the radio buttons “Inkomst” and “Utgift”.
6) Press the button “Lägg till”.

6.8 Add dated event

Function being tested: Add dated event to profile

RD reference: 4.1.1.4, 4.1.1.5.

Precondition: A profile is active

Use case: -

Input: The sum and date for the dated event.

Expected output: The dated event has been added to the profile and the modified profile is shown by the system.

Instructions:
1) Click on the tab “Lägg till post”.
2) Choose category from the drop down list “Kategori”.
3) Choose subcategory from the drop down list “Benämning”.
4) Choose whether the economic event is an income or an expense by clicking one of the radio buttons “Inkomst” and “Utgift”.
5) Enter the sum in the text field “Summa”.
6) Enter the date in the text field “Datum”.
7) Press the button “Lägg till”.

6.9 Show by category

Function being tested: Show details for the selected table by category.

RD reference: 4.1.1.5 and 4.1.1.6

Precondition: Dated events have previously been stored in the active profile. For the data to be compared with a budget, an existing budget must be chosen.

Use case: -

Input: -

Expected output: Details for the selected category is shown in the main table. If the user had another tab showing, the program will change to the tab “Visningsläge”.

Instructions:
1) Click on the category in the tree on the left side of the tab ”Visningsläge”.