Design and Implementation of Human – Computer Interfaces

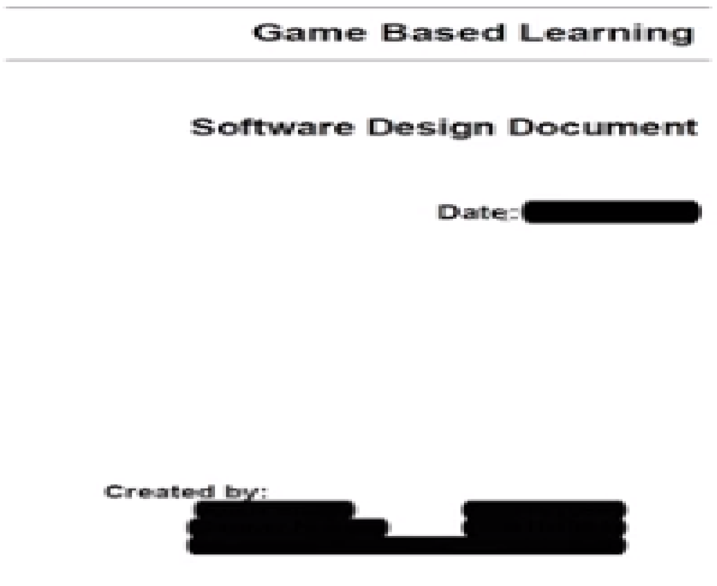
Prof. Dr. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module No # 05
Lecture No # 26
UML Case Study

Hello and welcome to the NPTEL MOOCs course on design and implementation of human computer interfaces lecture number 23 where we will discuss about a case study on creating a design document based on object oriented design approach using UML. In the previous lectures we have learned about the object oriented design approach we have also gone through the basic concepts of creating a design using the object-oriented design approach and representing the design using a language which is UML or unified modeling language.

In this lecture we are going to see a document that is created using UML for a particular system so the objective of this lecture is to demonstrate how to create a design document? Where we are using object oriented approach and representing the design using the language UML.

(Refer Slide Time: 01:50)



Game Based Learning

Software Design Document

Date:[redacted]

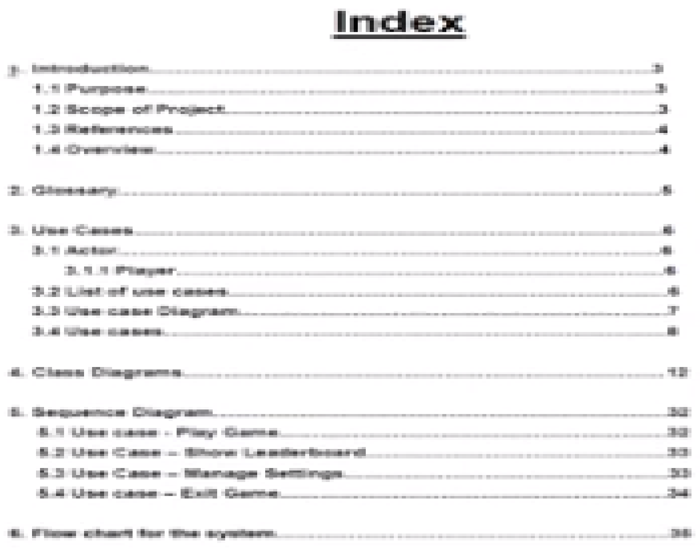
Created by:
[redacted]

This is about developing a game for learning. So, the name of the system is game based learning so this is the cover page of the design document this is a software design document with a creation date mentioned. So ideally the date should be mentioned when the document is created.

And the creators that are created by the name of the designers who have created this design document. The date need not be a single date instead here it can be a history of fast creation then revision, then refinement and final document creation date.

So this whole historical phases can be recorded or the first creation date can be recorded ideally the historical evolution should be recorded in the form of different dates.

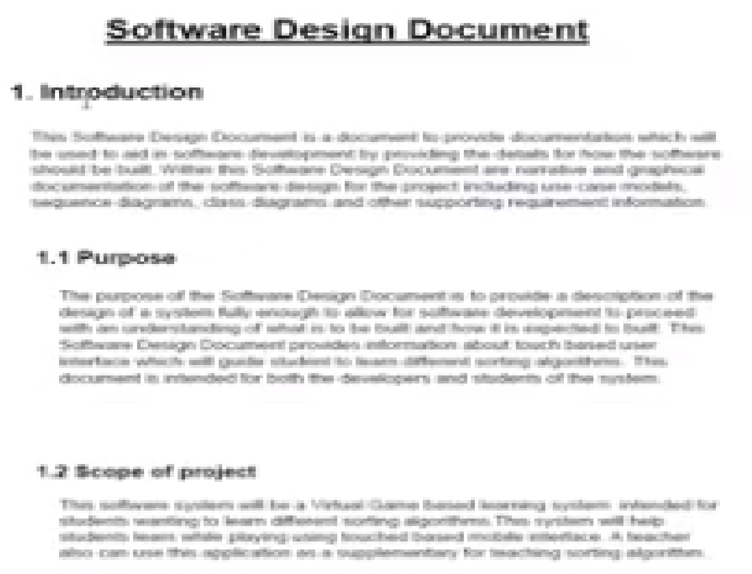
(Refer Slide Time: 02:52)



Index

The cover page is followed by a table of content where different sections that are present in the document are mentioned as shown here. Introduction, glossary of terms, use cases, class diagrams, interaction diagram in the form of a sequence diagram and optionally flow chart for the system. In the introduction if you may recollect earlier also we have seen 1 case study on SRS document similarly here in the introduction it is desirable that we keep these sections.

(Refer Slide Time: 03:33)



Software Design Document

1. Introduction

This Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within this Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, class diagrams and other supporting requirement information.

1.1 Purpose

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. This Software Design Document provides information about touch based user interface which will guide student to learn different sorting algorithms. This document is intended for both the developers and students of the system.

1.2 Scope of project

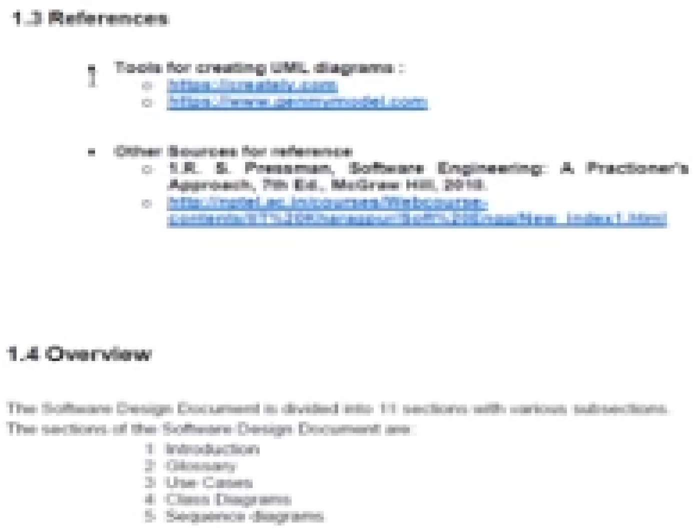
This software system will be a Virtual Game based learning system intended for students wanting to learn different sorting algorithms.This system will help students learn while playing using touched based mobile interface. A teacher also can use this application as a supplementary for teaching sorting algorithm.

General introduction this software design document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built? Some general statements are written within this design document there are narrative and graphical documentation of the software design for the project including use case models sequence diagram class diagram and other supporting requirement information.

This is followed by purpose statements what is the purpose? The purpose of the document is to provide a description of the design of a system fully enough to allow for software development is to proceed with an understanding of what is to be built and how it is expected to be built. This document provides information about touch based user interface which will guide student to learn different sorting algorithms this is the objective for this particular system that is mentioned under this purpose section.

This document is intended for both the developers and students or other users of the system. Scope of project another important section ideally should be there in the design document this software system will be a virtual game based learning system intended for students wanting to learn different sorting algorithms. The system will help students learn while playing using touch based mobile interface a teacher also can use this application as a supplementary for teaching sorting algorithms. So the introduction section contains the basic objective as well as intended users for the system.

(Refer Slide Time: 05:27)



1.3 References

- Tools for creating UML diagrams :
  - https://creately.com
  - https://www.genmymodel.com
- Other Sources for reference
  - 1.R. S. Pressman, Software Engineering: A Practioner's Approach, 7th Ed., McGraw Hill, 2010.
  - http://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Soft%20Engg/New_index1.html

1.4 Overview

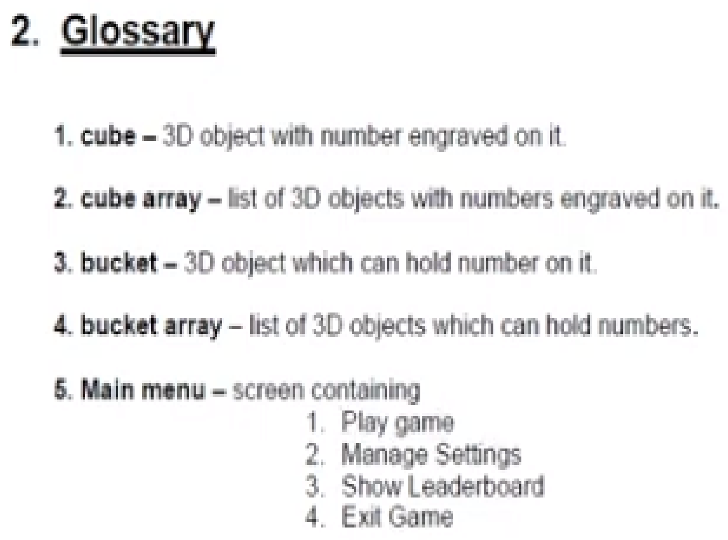
The Software Design Document is divided into 11 sections with various subsections.
The sections of the Software Design Document are:

1. Introduction
2. Glossary
3. Use Cases
4. Class Diagrams
5. Sequence diagrams

Some references can optionally be mentioned ideally it should be there like tools that are used for creating the diagrams other sources that are used to create the document these things can be listed under the reference section as shown here. And there can be an overview of the document the document is divided into 11 sections with various subsection. The major sections are introduction, glossary, use cases class diagrams and sequence diagrams.

This is an overview of the whole document which can be there in the introduction section the next section is the glossary of terms that are used in the subsequent document.

(Refer Slide Time: 06:20)

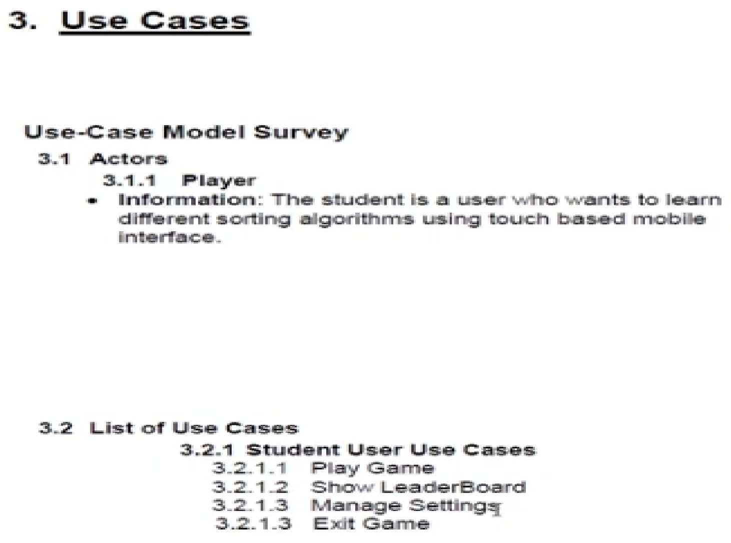


2. Glossary

1. **cube –** 3D object with number engraved on it.
2. **cube array –** list of 3D objects with numbers engraved on it.
3. **bucket –** 3D object which can hold number on it.
4. **bucket array –** list of 3D objects which can hold numbers.
5. **Main menu –** screen containing
   1. Play game
   2. Manage Settings
   3. Show Leaderboard
   4. Exit Game

For example in this document there are 5 important terms that are used cube which is a 3D object with number engraved on it. Cube array list of 3D objects with numbers engraved on each of these objects. Bucket 3D object which can hold numbers on it. Bucket array list of buckets or 3D objects which can hold numbers and main menu which is the screen containing these options play game manage setting show leaderboard and exit games so what each of these mean will be clear in subsequent part of the document.

(Refer Slide Time: 07:11)



3. Use Cases

Use-Case Model Survey

3.1 Actors

3.1.1 Player

- Information: The student is a user who wants to learn different sorting algorithms using touch based mobile interface.

3.2 List of Use Cases

3.2.1 Student User Use Cases

3.2.1.1 Play Game

3.2.1.2 Show LeaderBoard

3.2.1.3 Manage Settings

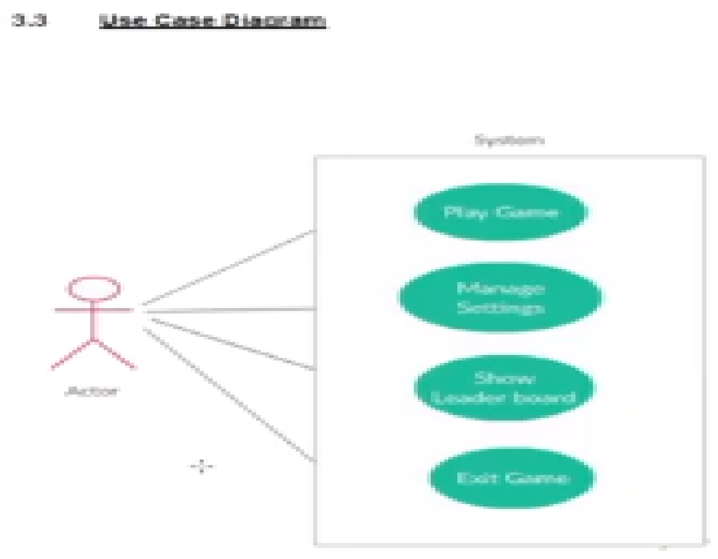
3.2.1.3 Exit Game

Then, comes the main portion of the document namely use cases remember that in object oriented design we focus on objects. And while representing the design there are 3 major views that we discussed the use case view, the behavioral view, and the structural view. The use case view relies on use cases so for this particular application that is game based learning let us see what are the use cases mentioned in the document?

Actors or player now who is a player a student is a user or player who wants to learn different sorting algorithms using touch based mobile interface this is how a player is explained in the design document itself. Now remember that here we are just mentioning student but the same system can be used by teachers as well so teacher can be another player however in this particular design document will not cover the design aspects for teacher.

We will focus only on student as users. So if that is the case that is students are the users what are the use cases for a student play games there are 4 use cases mentioned one is play games, then leaderboard, manage settings and exit game.

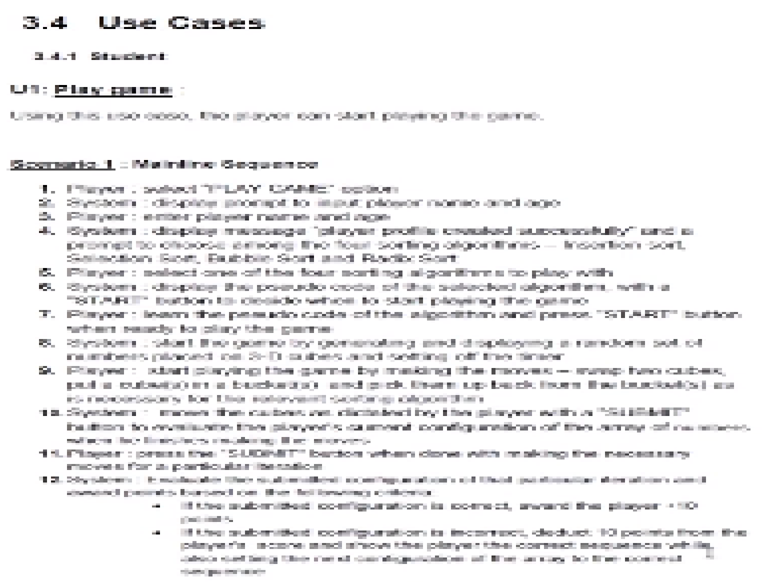
(Refer Slide Time: 08:56)


3.3 Use Case Diagram

The use cases are listed now we have to create the use case diagram if you recollect there are several components in the diagram namely the stick figure which represents the actor. The ellipses which represent the use case the system boundary and the lines. So this 4 use cases are captured in this diagram use case diagram here is the actor represented with the stick figure then these are the ellipses there are 4 such ellipse play game, manage setting, show leader board, and exit game.

Note that there need not be any order in which these 4 are placed it can be of any order. Because this placement does not indicate any; ordering between the different use cases. Now these 4 use cases are enclosed within this rectangle which represents the whole system for a student user.

(Refer Slide Time: 10:02)



3.4 Use Cases

3.4.1 Student

U1: Play game :

Using this use case, the player can start playing the game.

Scenario 1 : Mainline Sequence

1. Player : select "PLAY GAME" option
2. System : display prompt to input player name and age
3. Player : enter player name and age
4. System : display message "player profile created successfully" and a prompt to choose among the four sorting algorithms – Insertion sort, Selection Sort, Bubble Sort and Radix Sort
5. Player : select one of the four sorting algorithms to play with
6. System : display the pseudo code of the selected algorithm, with a "START" button to decide when to start playing the game
7. Player : learn the pseudo code of the algorithm and press "START" button when ready to play the game
8. System : start the game by generating and displaying a random set of numbers placed on 3-D cubes and setting off the timer
9. Player : start playing the game by making the moves – swap two cubes, put a cube(s) in a bucket(s) and pick them up back from the bucket(s) as is necessary for the relevant sorting algorithm
10. System : move the cubes as dictated by the player with a "SUBMIT" button to evaluate the player's current configuration of the array of numbers when he finishes making the moves
11. Player : press the "SUBMIT" button when done with making the necessary moves for a particular iteration
12. System : Evaluate the submitted configuration of that particular iteration and award points based on the following criteria:
    - If the submitted configuration is correct, award the player +10 points
    - If the submitted configuration is incorrect, deduct 10 points from the player's score and show the player the correct sequence while also setting the next configuration of the array to the correct sequence

Next thing is for each use case we are supposed to come up with the main line sequence and if required alternative sequence. Let us start with the first use case that is play game now in this use case using this use case the player can start playing the game so that is the use case so for this the main line sequence how it looks like according to the design proposed. One thing we have to keep in mind is that this sequence may be different depending on the designer. So whatever the designer feels that is captured within this sequence it needs not be unique.

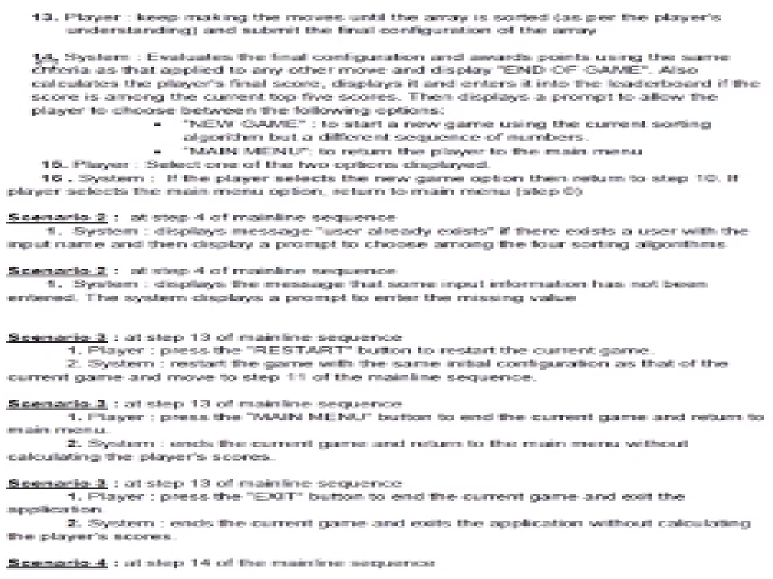
So, designer one can have 1 mainline sequence whereas designer 2 can come up with a different mainline sequence so here in the main line sequence according to this particular design there are several steps first player or the actor selects the play game option then the system displays prompt for the player to input name and age. Then the player enters the name and age then the system displays a message that player profile created successfully.

And, a prompt to choose among; the 4 sorting algorithms insertion sort, selection sort, bubble sort and radix sort. So, in this game there are these 4 algorithms sorting algorithms using which a player can learn different sorting algorithms. After that the player selects one of the 4 algorithms then system displays the pseudo code of the selection algorithm with a start option to decide when to start playing the game.

Next player launch the pseudo code of the algorithm and press or select start button when ready to play the game then the system then the system starts the game by generating and displaying a random set of numbers placed on 3D cubes and setting of the timer. So, there is 1 timer which is set up at this point player then start playing the game by making the moves swap 2 cubes put a cube or a set of cubes in a bucket and pick them up back from the bucket as is necessary for the relevant sorting algorithm.

System then moves the cubes as dictated by the player with a submit button to evaluate the player's current configuration of the array of numbers when the player finishes making the moves. Player then selects the submit button when done with making the necessary moves for a particular iteration. System then evaluates the submitted configuration of that particular iteration and award points based on the following criteria that is if the submitted configuration is correct award the player plus 10 points.

If the configuration is incorrect deduct 10 points from the player score and show the player the correct sequence while also setting the next configuration of the array to the correct sequence.
(Refer Slide Time: 13:26)



13. Player : keep making the moves until the array is sorted (as per the player's understanding) and submit the final configuration of the array

14. System : Evaluates the final configuration and awards points using the same criteria as that applied to any other move and display "END OF GAME". Also calculates the player's final score, displays it and enters it into the leaderboard if the score is among the current top five scores. Then displays a prompt to allow the player to choose between the following options:
   - "NEW GAME" : to start a new game using the current sorting algorithm but a different sequence of numbers.
   - "MAIN MENU": to return the player to the main menu

15. Player : Select one of the two options displayed.

16 . System : If the player selects the new game option then return to step 10. If player selects the main menu option, return to main menu (step 6)

Scenario 2 : at step 4 of mainline sequence
1. System : displays message "user already exists" if there exists a user with the input name and then display a prompt to choose among the four sorting algorithms

Scenario 2 : at step 4 of mainline sequence
1. System : displays the message that some input information has not been entered. The system displays a prompt to enter the missing value

Scenario 3 : at step 13 of mainline sequence
1. Player : press the "RESTART" button to restart the current game.
2. System : restart the game with the same initial configuration as that of the current game and move to step 11 of the mainline sequence.

Scenario 3 : at step 13 of mainline sequence
1. Player : press the "MAIN MENU" button to end the current game and return to main menu.
2. System : ends the current game and return to the main menu without calculating the player's scores.

Scenario 3 : at step 13 of mainline sequence
1. Player : press the "EXIT" button to end the current game and exit the application
2. System : ends the current game and exits the application without calculating the player's scores.

Scenario 4 : at step 14 of the mainline sequence

Then player keeps making the moves until the array is sorted as per the player's understanding and submit the final configuration of the array. Next the system evaluates the final configuration and awards points using the same criteria as that applied to any other move and display end of game also calculates the player's final score displays it and enters it into the leaderboard if the score is among the current top 5 scores.

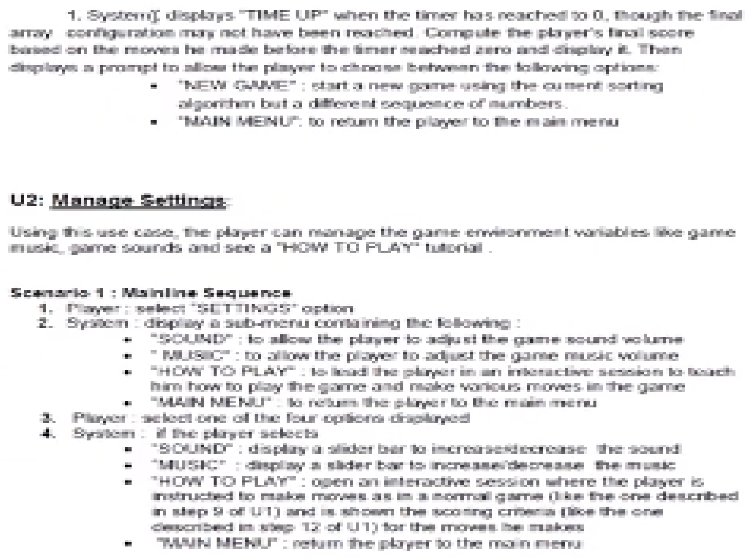
Then, displays a prompt to allow the player to; choose between the following options new game to start a new game, main menu to return to the main menu. Player then selects one of the 2 options displayed system then does the following if the player selects the new game option then return to step 10 if player selects the main menu option return to step 0 or at the beginning of this sequence that is the main line sequence as per this particular design document.

Now, there are some alternative sequences mentioned as well at step 4 of mainline sequence. The system displays message user already exists if there exists a user with the input name and then display a prompt to choose among the 4 sorting algorithms. In another scenario at step 4 of mainline sequence the system displays the message that some input information has not been entered the system then displays a prompt to enter the missing value.

Another scenario is at step 13 of mainline sequence here the player selects restart option to restart the current game and the system restarts the game with the same initial configuration as that of the current game and move to step 11 of the main line sequence. Another alternative sequence at the same step 13 can be the player selects the main menu option to end the current game and return to the main menu. Then in that case system ends the current game and return to the main menu without calculating player's scores.

Again at step 13 another alternative sequence can be player selects the exit option and the system ends the current game and exits the application without calculating player score.

(Refer Slide Time: 15:54)



1. System[] displays "TIME UP" when the timer has reached to 0, though the final array configuration may not have been reached. Compute the player's final score based on the moves he made before the timer reached zero and display it. Then displays a prompt to allow the player to choose between the following options:
   - "NEW GAME" : start a new game using the current sorting algorithm but a different sequence of numbers.
   - "MAIN MENU": to return the player to the main menu

U2: <u>Manage Settings</u>:

Using this use case, the player can manage the game environment variables like game music, game sounds and see a "HOW TO PLAY" tutorial .

Scenario 1 : Mainline Sequence

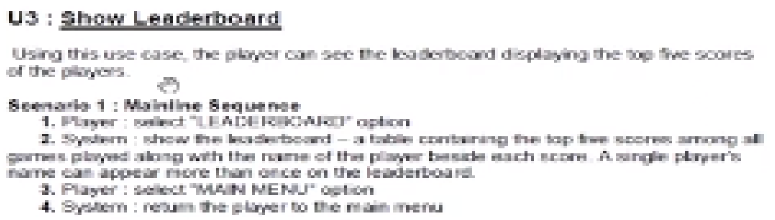
1. Player : select "SETTINGS" option
2. System : display a sub-menu containing the following :
   - "SOUND" : to allow the player to adjust the game sound volume
   - " MUSIC" : to allow the player to adjust the game music volume
   - "HOW TO PLAY" : to lead the player in an interactive session to teach him how to play the game and make various moves in the game
   - "MAIN MENU" : to return the player to the main menu
3. Player : select one of the four options displayed
4. System : if the player selects
   - "SOUND" : display a slider bar to increase/decrease the sound
   - "MUSIC" : display a slider bar to increase/decrease the music
   - "HOW TO PLAY" : open an interactive session where the player is instructed to make moves as in a normal game (like the one described in step 9 of U1) and is shown the scoring criteria (like the one described in step 12 of U1) for the moves he makes
   - "MAIN MENU" : return the player to the main menu

Then at step 14 of the main line sequence there can be some alternative sequence the system displays time up when the timer has reached to 0. Though the final array configuration may not have been reached at that point then computes the player's final scores based on the moves he made before the timer is 0 and display it .then displays a prompt to allow the player to choose between following options new game or main menu.

So, these are some of the alternative sequences that are mentioned for different steps the next use case is manage settings. In this use case the player can manage the game environment variables such as game music game sounds and see a how to play tutorial. Here the main line sequence is designed as follows first player selects setting option system then displays a sub menu containing sound music how to play and main menu options.

Player then selects one of the 4 options system then does the appropriate task depending on the option selected if sound is selected then displays a slider bar to increase decrease sound. If music is selected then displays a slider but to increase decrease music. If how to play is selected opens an interactive session where the player is instructed to make moves as in a normal game and if main menu is selected returns the player to the main menu.

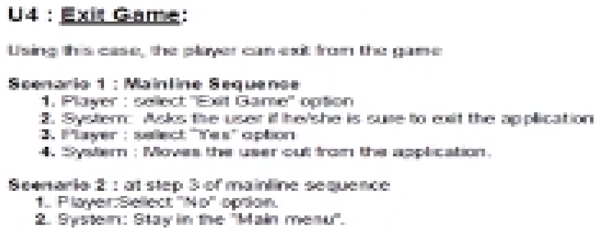
(Refer Slide Time: 17:28)


U3 : Show Leaderboard

Using this use case, the player can see the leaderboard displaying the top five scores of the players.

Scenario 1 : Mainline Sequence

1. Player : select "LEADERBOARD" option
2. System : show the leaderboard – a table containing the top five scores among all games played along with the name of the player beside each score. A single player's name can appear more than once on the leaderboard.
3. Player : select "MAIN MENU" option
4. System : return the player to the main menu


U4 : Exit Game:

Using this case, the player can exit from the game

Scenario 1 : Mainline Sequence

1. Player : select "Exit Game" option
2. System: Asks the user if he/she is sure to exit the application
3. Player : select "Yes" option
4. System : Moves the user out from the application.

Scenario 2 : at step 3 of mainline sequence

1. Player:Select "No" option.
2. System: Stay in the "Main menu".

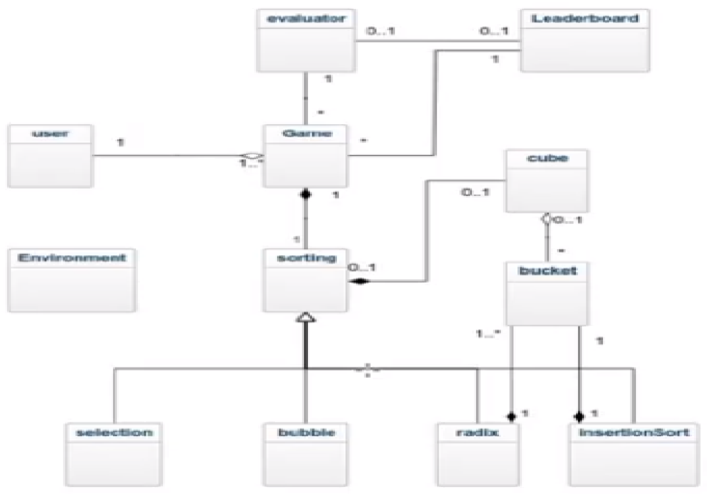
The third use case is show leaderboard; here the player can see the leaderboard displaying the top 5 scores of the players. Here there is only 1 mainline sequence that is player selects the leaderboard option and system displays the leaderboard which is nothing but a table containing top 5 scores among all games played along with the name of the player besides each score a single player's name can appear more than once in the leaderboard.

Player then selects main menu option and system returns the player to the main menu so there is no alternative sequence. And finally exit game so using this the player can exit from the game here there is a main line sequence which says that player selects exit game option system then asks user if user is sure to exit the application so there is some confirmatory dialog player selects yes option and system moves the user out from the application.

Now, there can be an alternative sequence at step 3 that means here player select no option and system stays in the main menu these are the 4 use cases with mainline and alternative sequences. Our next task is to create a behavioral view for these use cases. In the behavioral view we have to identify the objects and create interaction diagram for the objects.

(Refer Slide Time: 19:03)

4. Class Diagram

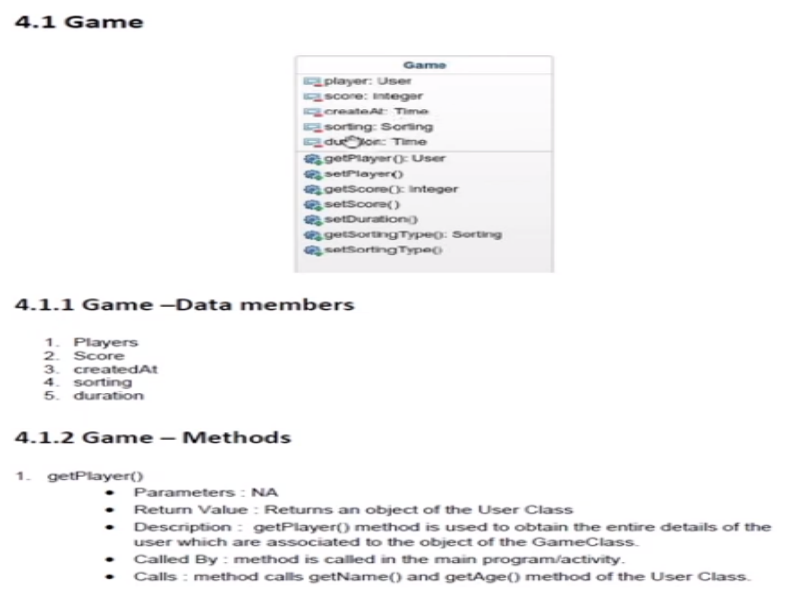


Now, that behavioral view will come later before that let us see the structural view with the classes and the relationship between the classes. This figure shows the structural view these are the classes evaluator, leaderboard, game, user, cube, bucket, sorting, environment, selection, bubble, radix, insertion sort these are the class names given now here we are not trying to figure out whether the class names or the classes or the number of classes are optimum, perfect, good anything.

So this is just a case study on design of the classes and design of the class diagram which is the structural view of the system design. So these classes are connected with arrows indicate the kind of relationship association or aggregation or similar such relationships as we have discussed in the previous lecture. As you can see different types of notations are used the field symbols here simple arrows to indicate the type of relationship along with the numerical values indicating the details of the relationships as discussed in the earlier lectures.

So along with this diagram we also need to show in detail the classes their details namely in terms of their attributes and member functions that is done separately because in this diagram everything cannot be shown as that will create a very complex diagram.

(Refer Slide Time: 21:10)



4.1 Game

```
Game
player: User
score: Integer
createAt: Time
sorting: Sorting
du[illegible]ion: Time
getPlayer(): User
setPlayer()
getScore(): Integer
setScore()
setDuration()
getSortingType(): Sorting
setSortingType()
```

4.1.1 Game –Data members

1. Players
2. Score
3. createdAt
4. sorting
5. duration

4.1.2 Game – Methods

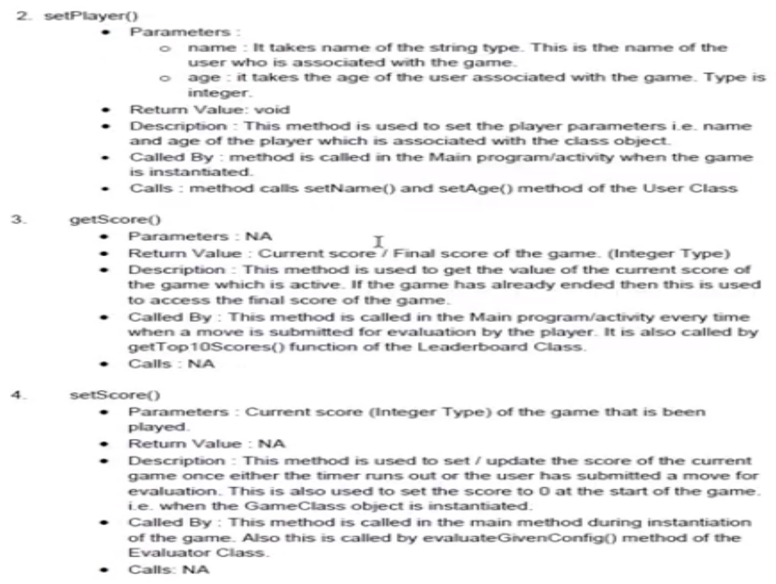
1. getPlayer()
    - Parameters : NA
    - Return Value : Returns an object of the User Class
    - Description : getPlayer() method is used to obtain the entire details of the user which are associated to the object of the GameClass.
    - Called By : method is called in the main program/activity.
    - Calls : method calls getName() and getAge() method of the User Class.

So, the class definitions are added separately in the document for example the game class shown here. So, for this game this is the details having attributes there are 5 attributes or data values and there are 7 member functions. So that data members are player score created at sorting and duration whereas methods are get player, set player, get score, set score, set duration get sorting type and set sorting type.

For each method some more details are given like say for example get player, whether it takes any argument no argument return value returns an object of the user class description get player method is used to obtain the entire details of the user which are associated to the object of the game class called by method is called by the main program or activity. It calls get name and gate age methods of the user class.

So, all these details are mentioned separately for game class also in the actual definition you can see the type of data that can be there in the attributes so player is user type data, score integer type data, created at time data, sorting data, duration another time data.

(Refer Slide Time: 23:16)



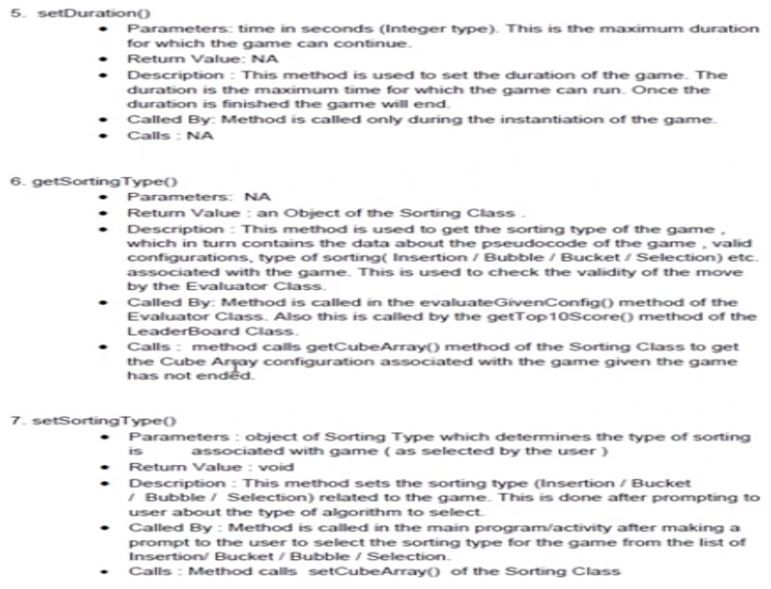
2. setPlayer()
   - Parameters :
     - name : It takes name of the string type. This is the name of the user who is associated with the game.
     - age : it takes the age of the user associated with the game. Type is integer.
   - Return Value: void
   - Description : This method is used to set the player parameters i.e. name and age of the player which is associated with the class object.
   - Called By : method is called in the Main program/activity when the game is instantiated.
   - Calls : method calls setName() and setAge() method of the User Class

3. getScore()
   - Parameters : NA
   - Return Value : Current score / Final score of the game. (Integer Type)
   - Description : This method is used to get the value of the current score of the game which is active. If the game has already ended then this is used to access the final score of the game.
   - Called By : This method is called in the Main program/activity every time when a move is submitted for evaluation by the player. It is also called by getTop10Scores() function of the Leaderboard Class.
   - Calls : NA

4. setScore()
   - Parameters : Current score (Integer Type) of the game that is been played.
   - Return Value : NA
   - Description : This method is used to set / update the score of the current game once either the timer runs out or the user has submitted a move for evaluation. This is also used to set the score to 0 at the start of the game. i.e. when the GameClass object is instantiated.
   - Called By : This method is called in the main method during instantiation of the game. Also this is called by evaluateGivenConfig() method of the Evaluator Class.
   - Calls: NA

Similarly for this class other member functions are described in details in terms of parameters, written value, description called by and calls which are the function that it calls. So, for all such all functions or all member functions those details are provided.

(Refer Slide Time: 23:37)



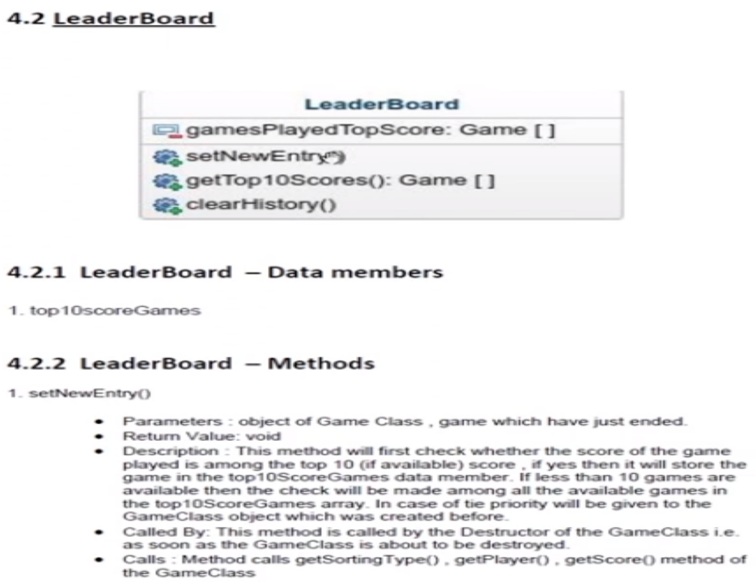
5. setDuration()
   - Parameters: time in seconds (Integer type). This is the maximum duration for which the game can continue.
   - Return Value: NA
   - Description : This method is used to set the duration of the game. The duration is the maximum time for which the game can run. Once the duration is finished the game will end.
   - Called By: Method is called only during the instantiation of the game.
   - Calls : NA

6. getSortingType()
   - Parameters: NA
   - Return Value : an Object of the Sorting Class .
   - Description : This method is used to get the sorting type of the game , which in turn contains the data about the pseudocode of the game , valid configurations, type of sorting(Insertion / Bubble / Bucket / Selection) etc. associated with the game. This is used to check the validity of the move by the Evaluator Class.
   - Called By: Method is called in the evaluateGivenConfig() method of the Evaluator Class. Also this is called by the getTop10Score() method of the LeaderBoard Class.
   - Calls : method calls getCubeArray() method of the Sorting Class to get the Cube Array configuration associated with the game given the game has not ended.

7. setSortingType()
   - Parameters : object of Sorting Type which determines the type of sorting is associated with game (as selected by the user)
   - Return Value : void
   - Description : This method sets the sorting type (Insertion / Bucket / Bubble / Selection) related to the game. This is done after prompting to user about the type of algorithm to select.
   - Called By : Method is called in the main program/activity after making a prompt to the user to select the sorting type for the game from the list of Insertion/ Bucket / Bubble / Selection.
   - Calls : Method calls setCubeArray() of the Sorting Class

So there are 7 member functions so 7 such set of details are provided.

(Refer Slide Time: 23:47)



4.2 LeaderBoard

LeaderBoard

gamesPlayedTopScore: Game [ ]

setNewEntry()

getTop10Scores(): Game [ ]

clearHistory()

4.2.1 LeaderBoard – Data members

1. top10scoreGames

4.2.2 LeaderBoard – Methods

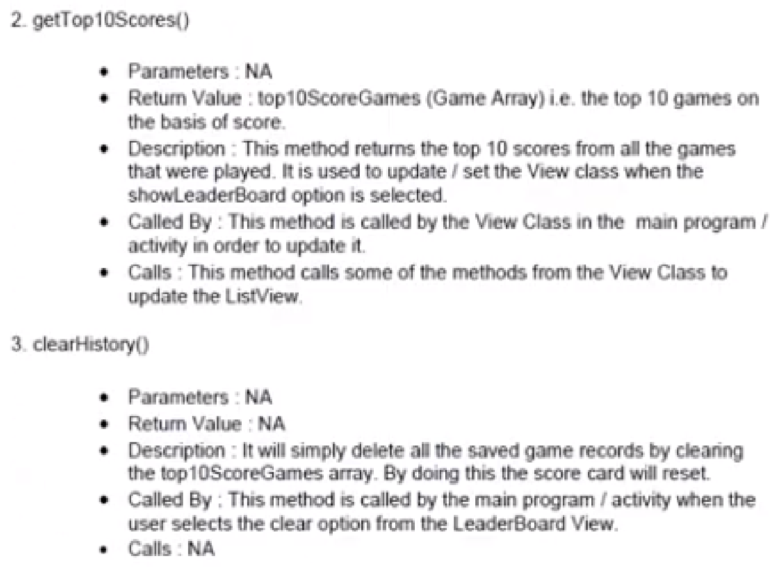
1. setNewEntry()

- Parameters : object of Game Class , game which have just ended.
- Return Value: void
- Description : This method will first check whether the score of the game played is among the top 10 (if available) score , if yes then it will store the game in the top10ScoreGames data member. If less than 10 games are available then the check will be made among all the available games in the top10ScoreGames array. In case of tie priority will be given to the GameClass object which was created before.
- Called By: This method is called by the Destructor of the GameClass i.e. as soon as the GameClass is about to be destroyed.
- Calls : Method calls getSortingType() , getPlayer() , getScore() method of the GameClass

Next is leaderboard class it has 1 attribute games played top score and it has 3 member functions set new entry, get top 10 cores and clear history. Like before for each method or member function the details are provided for example set new entry function parameters objects of game class game which have just ended return value null or void. Description this method will first check whether the score of the game played is among the top 10.

If available score if yes then it will store the game in the top 10 score games data member. If less than 10 games are available then the check will be made among all the available games in the top 10 score games array. In case of tie priority will be given to the game class object which was created before. So, this summarizes the things that this member function does it is called by the destructor of the game class. And it calls other functions such as get sorting type get player gate score method of the game class so the dependencies are clearly specified under the description.

(Refer Slide Time: 25:24)



2. getTop10Scores()

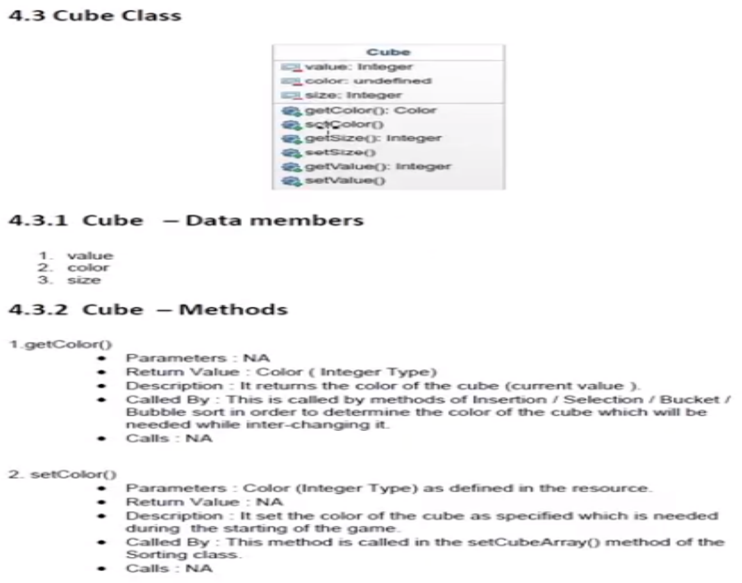
- Parameters : NA
- Return Value : top10ScoreGames (Game Array) i.e. the top 10 games on the basis of score.
- Description : This method returns the top 10 scores from all the games that were played. It is used to update / set the View class when the showLeaderBoard option is selected.
- Called By : This method is called by the View Class in the main program / activity in order to update it.
- Calls : This method calls some of the methods from the View Class to update the ListView.

3. clearHistory()

- Parameters : NA
- Return Value : NA
- Description : It will simply delete all the saved game records by clearing the top10ScoreGames array. By doing this the score card will reset.
- Called By : This method is called by the main program / activity when the user selects the clear option from the LeaderBoard View.
- Calls : NA

Similarly, for other member functions of the class similar descriptions are provided for all the other classes.

(Refer Slide Time: 25:36)



4.3 Cube Class

Cube
value: Integer
color: undefined
size: Integer
getColor(): Color
setColor()
getSize(): Integer
setSize()
getValue(): Integer
setValue()

4.3.1 Cube – Data members

1. value
2. color
3. size

4.3.2 Cube – Methods

1.getColor()

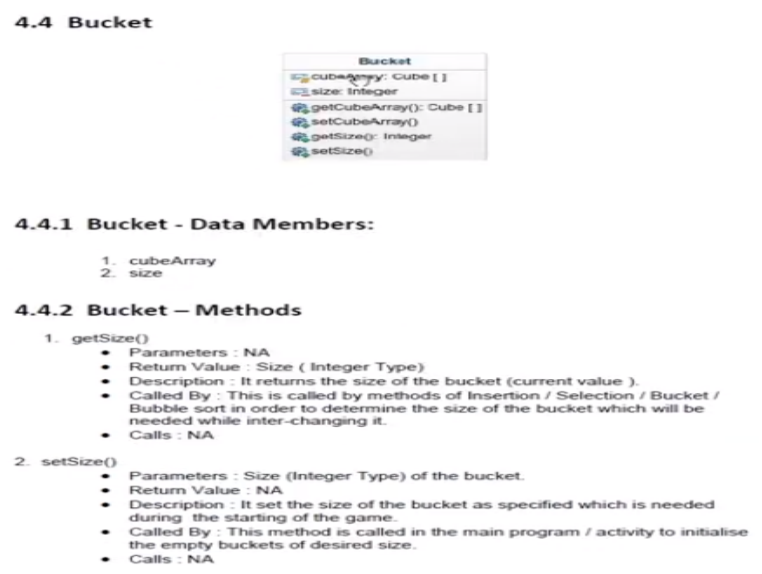
- Parameters : NA
- Return Value : Color (Integer Type)
- Description : It returns the color of the cube (current value).
- Called By : This is called by methods of Insertion / Selection / Bucket / Bubble sort in order to determine the color of the cube which will be needed while inter-changing it.
- Calls : NA

2. setColor()

- Parameters : Color (Integer Type) as defined in the resource.
- Return Value : NA
- Description : It set the color of the cube as specified which is needed during the starting of the game.
- Called By : This method is called in the setCubeArray() method of the Sorting class.
- Calls : NA

Cube class having 3 attributes and 6 member functions.

(Refer Slide Time: 25:44)



4.4 Bucket

Bucket
cubeArray: Cube [ ]
size: Integer
getCubeArray(): Cube [ ]
setCubeArray()
getSize(): Integer
setSize()

4.4.1 Bucket - Data Members:

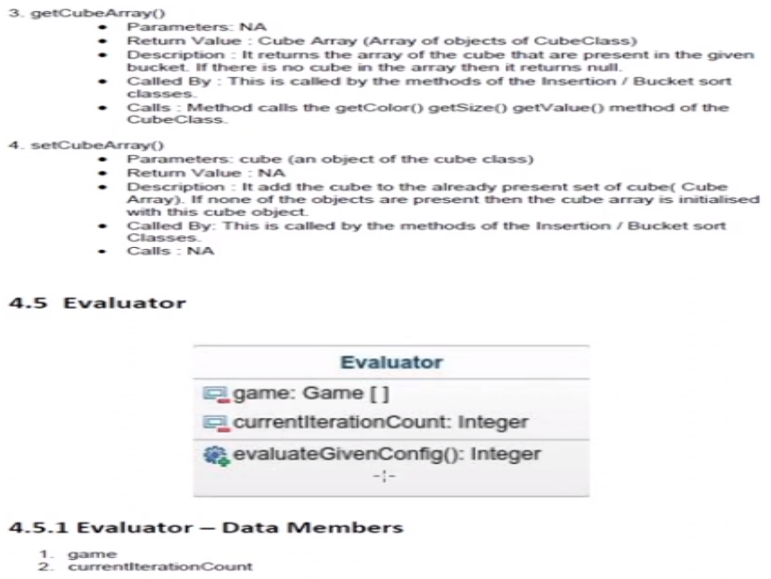
1. cubeArray
2. size

4.4.2 Bucket – Methods

1. getSize()
   - Parameters : NA
   - Return Value : Size (Integer Type)
   - Description : It returns the size of the bucket (current value).
   - Called By : This is called by methods of Insertion / Selection / Bucket / Bubble sort in order to determine the size of the bucket which will be needed while inter-changing it.
   - Calls : NA
2. setSize()
   - Parameters : Size (Integer Type) of the bucket.
   - Return Value : NA
   - Description : It set the size of the bucket as specified which is needed during the starting of the game.
   - Called By : This method is called in the main program / activity to initialise the empty buckets of desired size.
   - Calls : NA

Bucket class having 2 attributes and 4 member functions.

(Refer Slide Time: 25:53)



3. getCubeArray()
   - Parameters: NA
   - Return Value : Cube Array (Array of objects of CubeClass)
   - Description : It returns the array of the cube that are present in the given bucket. If there is no cube in the array then it returns null.
   - Called By : This is called by the methods of the Insertion / Bucket sort classes.
   - Calls : Method calls the getColor() getSize() getValue() method of the CubeClass.
4. setCubeArray()
   - Parameters: cube (an object of the cube class)
   - Return Value : NA
   - Description : It add the cube to the already present set of cube(Cube Array). If none of the objects are present then the cube array is initialised with this cube object.
   - Called By: This is called by the methods of the Insertion / Bucket sort Classes.
   - Calls : NA

4.5 Evaluator

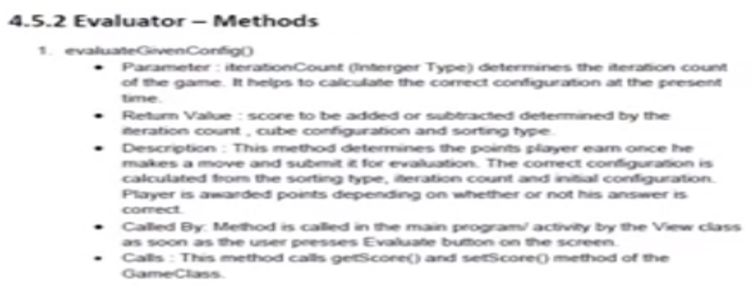
Evaluator
game: Game [ ]
currentIterationCount: Integer
evaluateGivenConfig(): Integer

4.5.1 Evaluator – Data Members

1. game
2. currentIterationCount

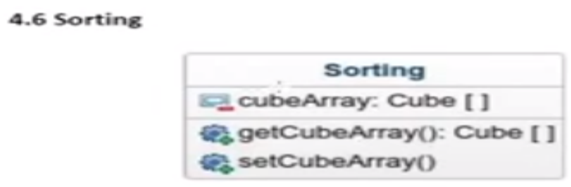
Evaluator having 2 attributes and 1 member functions.

(Refer Slide Time: 25:58)



4.5.2 Evaluator – Methods

1. evaluateGivenConfig()
   - Parameter : iterationCount (Interger Type) determines the iteration count of the game. It helps to calculate the correct configuration at the present time.
   - Return Value : score to be added or subtracted determined by the iteration count , cube configuration and sorting type.
   - Description : This method determines the points player earn once he makes a move and submit it for evaluation. The correct configuration is calculated from the sorting type, iteration count and initial configuration. Player is awarded points depending on whether or not his answer is correct.
   - Called By: Method is called in the main program/ activity by the View class as soon as the user presses Evaluate button on the screen.
   - Calls : This method calls getScore() and setScore() method of the GameClass.

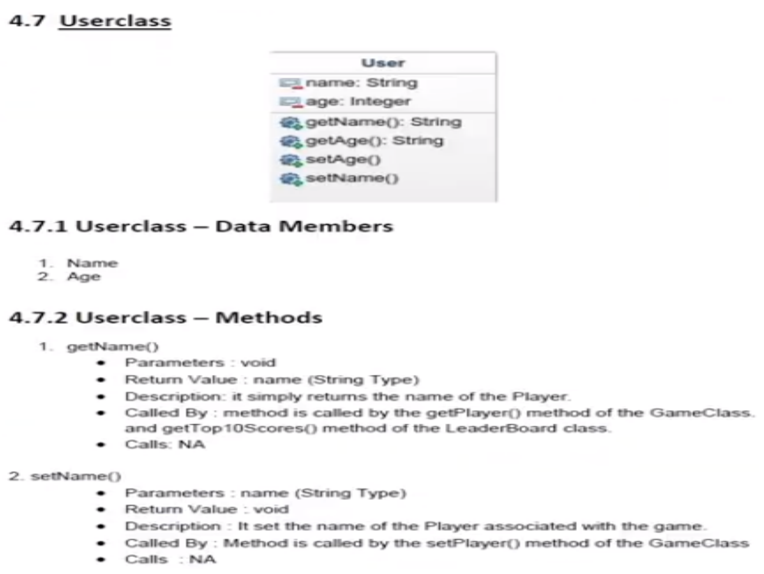


4.6 Sorting

Sorting
cubeArray: Cube [ ]
getCubeArray(): Cube [ ]
setCubeArray()

Sorting having 1 attribute and 2 member functions.

(Refer Slide Time: 26:06)



4.7 Userclass

User
name: String
age: Integer
getName(): String
getAge(): String
setAge()
setName()

4.7.1 Userclass – Data Members

1. Name
2. Age

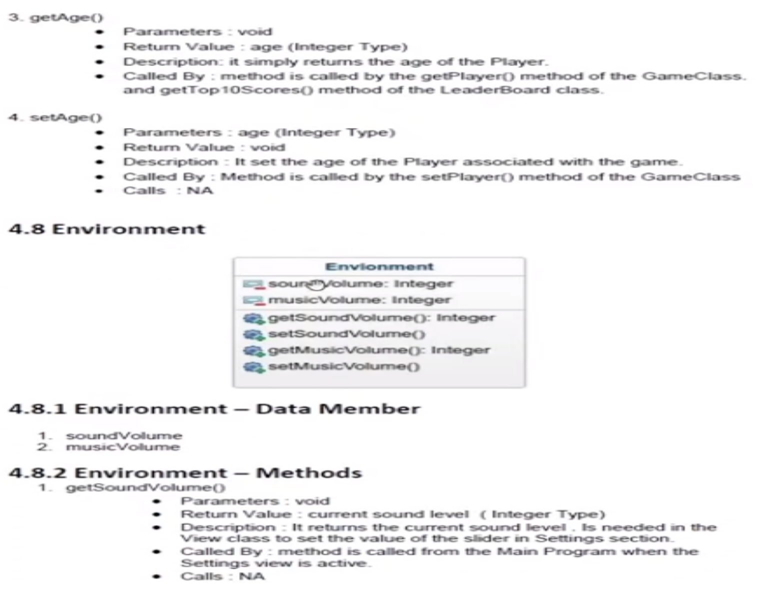
4.7.2 Userclass – Methods

1. getName()
   - Parameters : void
   - Return Value : name (String Type)
   - Description: it simply returns the name of the Player.
   - Called By : method is called by the getPlayer() method of the GameClass. and getTop10Scores() method of the LeaderBoard class.
   - Calls: NA

2. setName()
   - Parameters : name (String Type)
   - Return Value : void
   - Description : It set the name of the Player associated with the game.
   - Called By : Method is called by the setPlayer() method of the GameClass
   - Calls : NA

User class having 2 attributes and 4 member functions.

(Refer Slide Time: 26:11)



3. getAge()
- Parameters : void
- Return Value : age (Integer Type)
- Description: it simply returns the age of the Player.
- Called By : method is called by the getPlayer() method of the GameClass. and getTop10Scores() method of the LeaderBoard class.

4. setAge()
- Parameters : age (Integer Type)
- Return Value : void
- Description : It set the age of the Player associated with the game.
- Called By : Method is called by the setPlayer() method of the GameClass
- Calls : NA

4.8 Environment

Envionment
- soundVolume: Integer
- musicVolume: Integer
- getSoundVolume(): Integer
- setSoundVolume()
- getMusicVolume(): Integer
- setMusicVolume()

4.8.1 Environment – Data Member

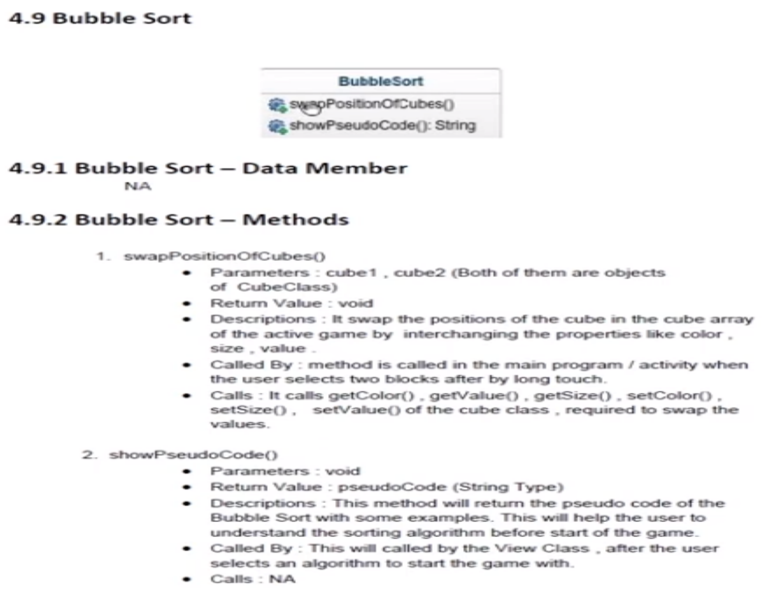
1. soundVolume
2. musicVolume

4.8.2 Environment – Methods

1. getSoundVolume()
   - Parameters : void
   - Return Value : current sound level (Integer Type)
   - Description : It returns the current sound level . Is needed in the View class to set the value of the slider in Settings section.
   - Called By : method is called from the Main Program when the Settings view is active.
   - Calls : NA

Environment having 2 attributes and 4 member functions.

(Refer Slide Time: 26:16)



4.9 Bubble Sort

BubbleSort
- swapPositionOfCubes()
- showPseudoCode(): String

4.9.1 Bubble Sort – Data Member

NA

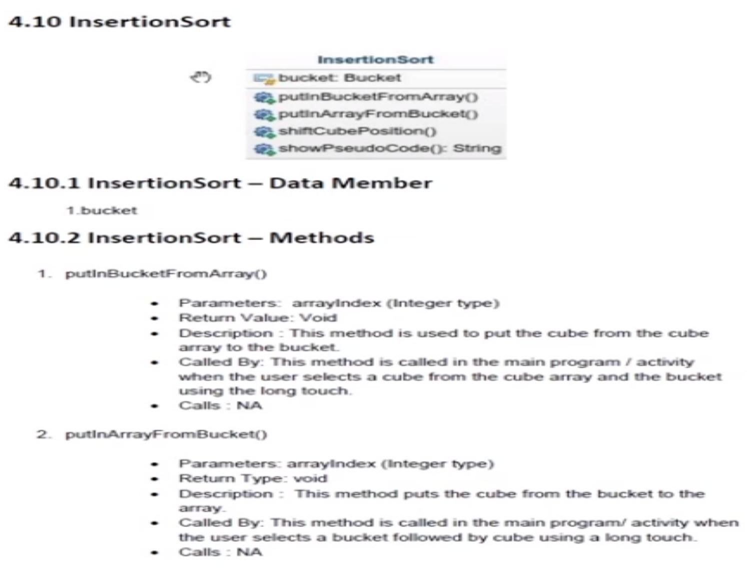
4.9.2 Bubble Sort – Methods

1. swapPositionOfCubes()
   - Parameters : cube1 , cube2 (Both of them are objects of CubeClass)
   - Return Value : void
   - Descriptions : It swap the positions of the cube in the cube array of the active game by interchanging the properties like color , size , value .
   - Called By : method is called in the main program / activity when the user selects two blocks after by long touch.
   - Calls : It calls getColor() , getValue() , getSize() , setColor() , setSize() , setValue() of the cube class , required to swap the values.

2. showPseudoCode()
   - Parameters : void
   - Return Value : pseudoCode (String Type)
   - Descriptions : This method will return the pseudo code of the Bubble Sort with some examples. This will help the user to understand the sorting algorithm before start of the game.
   - Called By : This will called by the View Class , after the user selects an algorithm to start the game with.
   - Calls : NA

Bubble chart having 2 member functions only no attribute.

(Refer Slide Time: 26:22)



4.10 InsertionSort

InsertionSort
bucket: Bucket
putInBucketFromArray()
putInArrayFromBucket()
shiftCubePosition()
showPseudoCode(): String

4.10.1 InsertionSort – Data Member

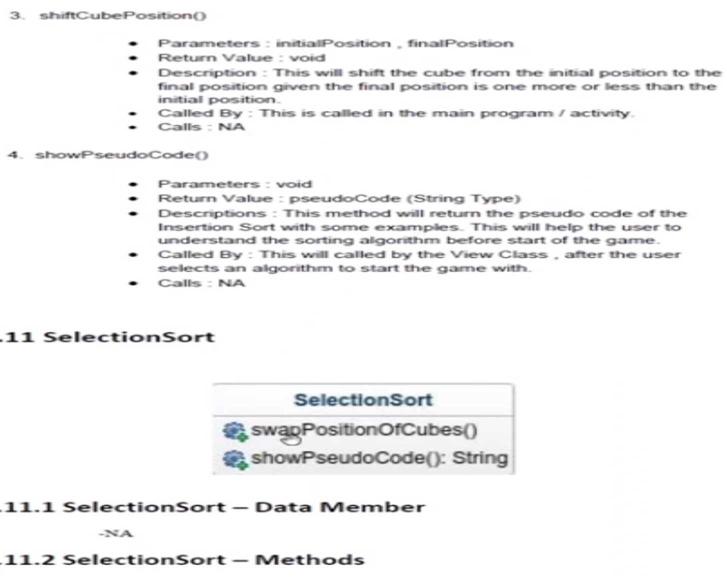
1.bucket

4.10.2 InsertionSort – Methods

1. putInBucketFromArray()
    - Parameters: arrayIndex (Integer type)
    - Return Value: Void
    - Description : This method is used to put the cube from the cube array to the bucket.
    - Called By: This method is called in the main program / activity when the user selects a cube from the cube array and the bucket using the long touch.
    - Calls : NA
2. putInArrayFromBucket()
    - Parameters: arrayIndex (Integer type)
    - Return Type: void
    - Description : This method puts the cube from the bucket to the array.
    - Called By: This method is called in the main program/ activity when the user selects a bucket followed by cube using a long touch.
    - Calls : NA

Insertions are having 1 attribute and 4 member functions.

(Refer Slide Time: 26:28)



3. shiftCubePosition()
    - Parameters : initialPosition , finalPosition
    - Return Value : void
    - Description : This will shift the cube from the initial position to the final position given the final position is one more or less than the initial position.
    - Called By : This is called in the main program / activity.
    - Calls : NA
4. showPseudoCode()
    - Parameters : void
    - Return Value : pseudoCode (String Type)
    - Descriptions : This method will return the pseudo code of the Insertion Sort with some examples. This will help the user to understand the sorting algorithm before start of the game.
    - Called By : This will called by the View Class , after the user selects an algorithm to start the game with.
    - Calls : NA

4.11 SelectionSort

SelectionSort
swapPositionOfCubes()
showPseudoCode(): String

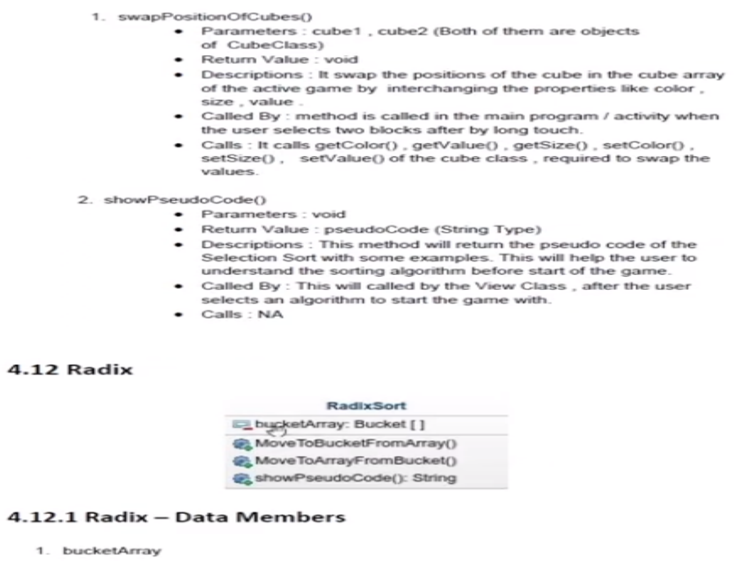
4.11.1 SelectionSort – Data Member

-NA

4.11.2 SelectionSort – Methods

Selection sort having 2 member functions no attributes.

(Refer Slide Time: 26:33)



1. swapPositionOfCubes()
   - Parameters : cube1 , cube2 (Both of them are objects of CubeClass)
   - Return Value : void
   - Descriptions : It swap the positions of the cube in the cube array of the active game by interchanging the properties like color , size , value .
   - Called By : method is called in the main program / activity when the user selects two blocks after by long touch.
   - Calls : It calls getColor() , getValue() , getSize() , setColor() , setSize() , setValue() of the cube class , required to swap the values.
2. showPseudoCode()
   - Parameters : void
   - Return Value : pseudoCode (String Type)
   - Descriptions : This method will return the pseudo code of the Selection Sort with some examples. This will help the user to understand the sorting algorithm before start of the game.
   - Called By : This will called by the View Class , after the user selects an algorithm to start the game with.
   - Calls : NA

4.12 Radix

RadixSort
bucketArray: Bucket [ ]
MoveToBucketFromArray()
MoveToArrayFromBucket()
showPseudoCode(): String

4.12.1 Radix – Data Members

1. bucketArray

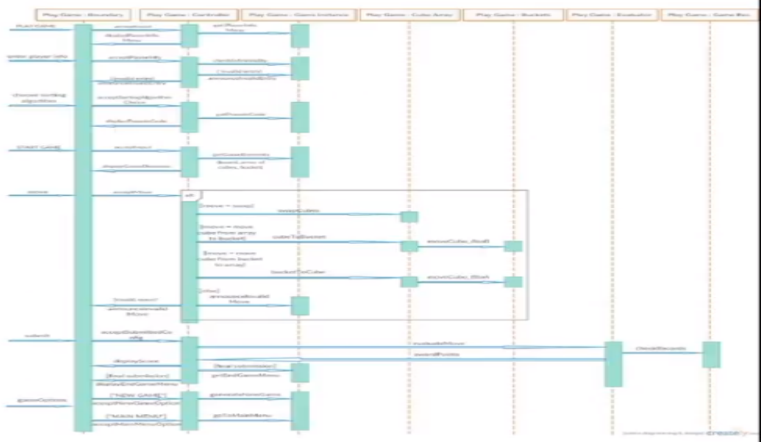
Radix sort having 1 attribute and 3 member functions. So that gives us a structural view of the system in terms of classes and their relationships with detailed description for each of the classes including description of the attributes and description of the member functions. As you can see this is how we can give the details there can be 1 diagram only mentioning the names of the classes and their relationship and separate entries for details for each class.

And in the details we have to keep give details of the attributes as well as member functions for member functions we have to give detailed information regarding its input, output, dependencies and the algorithm or what it does exactly in the form of description.

(Refer Slide Time: 27:43)

5. Sequence Diagram

5.1 Use Case : Play Game



32

The next thing that we should do is we should create a behavioral diagram that mean how the objects that are instantiation of these classes behave while the system is being executed. That we can do in terms of sequence diagram which we have learned earlier. So here what we can do is create sequence diagram for each use cases in terms of objects on top we have mentioned the object names as you can see here.

So, there are 1, 2, 3, 4, 5, 6, 7 objects for this use case as identified by the designer again these are not unique diagrams. So, for the same use case a different designer or a team of designers can choose different sets of objects and accordingly can come up with a different sequence diagram so there is nothing unique about these diagrams for each object the life span is shown by the length of the bar that represents the object.

For example, the first object has the maximum lifespan, and then the second object gets created and destroyed periodically as shown with these individual bars that are true again for the third object up to this point. The fourth object again optionally gets created and periodically gets created and destroyed 3 such are there. For fifth object there are 2 such occurrences. Sixth object gets created and destroyed within this range as shown with this bar. And the seventh object is the one with list life span gets created once and destroyed as indicated by this particular bar with its particular position. And the arrows indicate the interaction between the objects when they are alive with the direction indicating the direction of message passing so this is for play game.

(Refer Slide Time: 30:04)

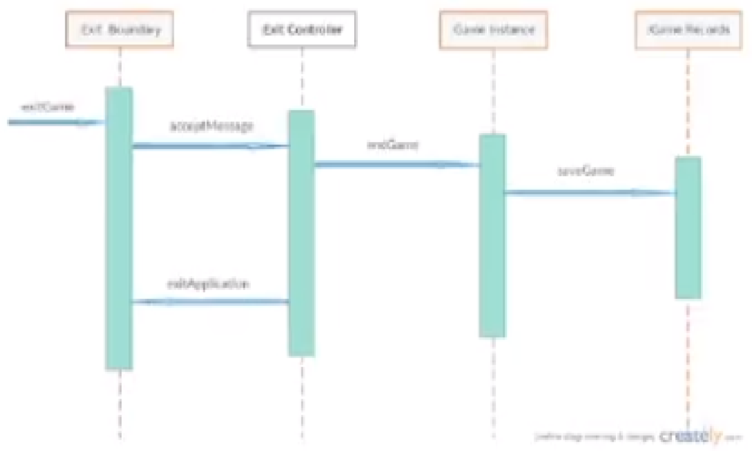
5.2 **Use Case** : Show Leaderboard

5.3 **Use Case** : Manage Settings

For show leaderboard use case this is the sequence diagram there are 3 objects the first object has the maximum life span followed by the second object followed by the third object. For use case manage setting again there are 3 objects first object with maximum life span followed by the second object which is followed by the third object note that here the objects are created based on the domain model so we have boundary objects, entity objects and controller objects.

(Refer Slide Time: 30:40)

5.4 Use Case : Exit Game



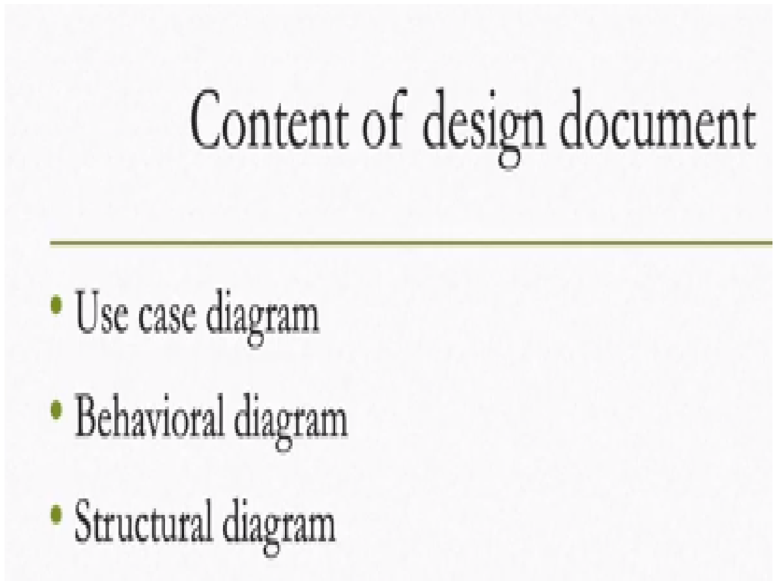
And finally we have the exit game use case for which there are 4 objects first one is having the maximum life span then the second one followed by the third object and the fourth object is having the least life span. So, what these diagrams indicate is that when this particular use case is being executed that means when suppose the user is exiting the game or the user is playing the game.

Then the particular objects belonging to the appropriate classes those get created destroyed as per the diagram and those interact with each other as per the sequence diagram or as specified in the sequence diagram. So, these diagrams indicate how the objects that are result of instantiation of the classes which we have seen in the structural diagram how these objects behave while a particular use case is being executed.

While we are learning the concepts of UML and the different views and the diagrams we mentioned that a right way to do is to first identify the use cases for each use cases create this behavioral diagram. And get the list of objects and then merge them together to come to the structural diagram however in this design document as you can see that sequence is not followed instead first the use cases are identified then structural view is created.

And then the behavioral view is created this is also all right it is not that only the first approach should be followed however it sometimes helps if we first identify the objects and from there generalize the classes. But it is up to the designer which path to follow whether first use case then structural followed by behavioral diagrams or first use case diagrams followed by behavioral diagrams and from there come to the structural diagram both are all right.

(Refer Slide Time: 33:06)



Content of design document

- Use case diagram
- Behavioral diagram
- Structural diagram

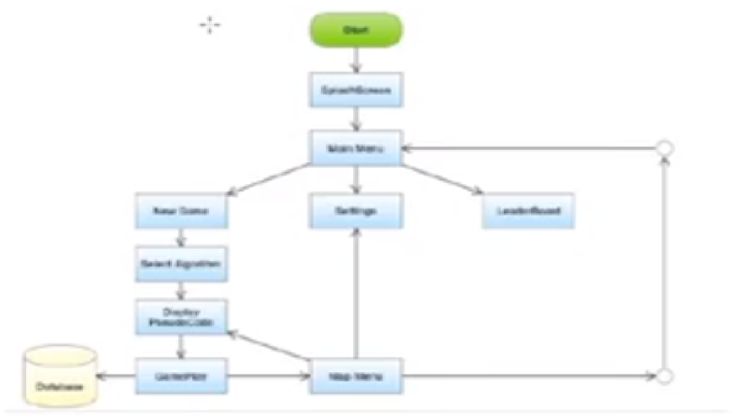
So, that is in a nutshell what should be part of the design document when we are creating a design based on the object oriented design approach and representing it using UML. So, we have to include the use cases with use case diagram mainline sequence and alternative sequence for each use case then we have to include the objects and their behavior during run time that is the behavioral diagram.

We have also to include the structural diagram that is what are the classes and how they are connected to each other how they are related to each other the particular sequence in which you are going to include these information in the document is up to you up to the designer that is you can first have the use cases followed by structural diagram or the class diagram followed by behavioral diagram.

Or you can have the use case followed by behavioral diagram followed by structural diagram either is fine optionally at the end of the document for the better understanding of the data flow in the system you can also include a flowchart as shown here.

(Refer Slide Time: 34:34)

6. Flow Chart for the System



Although it is optional and not mandatory so that is how we can conceptualize and create a design document following object oriented design approach. Earlier we have seen how to create a design document following the functional approach where we use TDF and entity relationship diagram. In this case study we have seen how to create a design document where we use object oriented design and UML as a language to express the design I hope you enjoyed this lecture and learned how to create such a document looking forward to meet you in the next lecture thank you and good bye.