

Bilkent University

Engineering Faculty

Department of Computer Engineering

**SNAKE - THE EXTRAORDINARY ADVENTURE**

Analysis Report

CS 319

Object-Oriented Software Engineering

Course Instructor: Hüseyin Özgür TAN

Teaching Assistant: İstemi Bahçeci

Semester: 2015-2016 Fall

Section 2 - Group 5

Cihan ERYONUCU

Mehmet Çağrıbey GÜVENDİK

Serhat ÇETİNKAYA

Zülal BİNGÖL

**CONTENTS**

1. Introduction.................................................................................................................3
2. Requirement Analysis..................................................................................................4
   1. Overview..............................................................................................................4
   2. Functional Requirements......................................................................................5
   3. Non-Functional Requirements..............................................................................6
   4. Constraints............................................................................................................7
   5. Scenarios...............................................................................................................8
   6. Use Cases and Game's Use Case Model .............................................................11
   7. User Interface.......................................................................................................18
3. Analysis........................................................................................................................21
   1. Object Model........................................................................................................21
      1. Domain Lexicon.........................................................................................21
      2. Class Diagrams...........................................................................................23
   2. Dynamic Models...................................................................................................24
      1. State Chart Diagrams..................................................................................25
      2. Sequence Diagrams.....................................................................................25
4. Conclusion....................................................................................................................29
5. Softwares Used............................................................................................................30

**1. INTRODUCTION**

We will implement a game called Snake - The Extraordinary Adventure. There are a lot of snake games with common scheme. In well-known snake game there is snake and the baits. User controls snake which is supposed to eat baits. With eating each bait, snakes’ length gets longer. If snakes’ head touches any part of its own, snake dies. Our game is inspired by Snake game but with different aspects, various events and more fun. Therefore, our Snake game will be different and difficult than any others.

Instead of just normal baits, there will be six different baits with different colors. Each bait has different effects which are dying, extending, bonus point, swapping head and tail, invulnerability to death. Hence, variations of baits will affect the entire gameplay significantly. Since every bait has different effects, users’ strategy should be depend on deciding what bait should be eaten for snakes’ length. In addition, scoreboard will be available in game. After snake dies, user will be able to see his/her score.

There will be also walls which appear in the game. Shape of the walls will be straight. Walls will complicate snakes’ movement and also affect game play radically. With combination of various baits and walls, game complexity and difficulty will be much higher than regular Snake game.

Our game will be implemented in Java programming language. We will avoid any operating system specific dependencies to be able to run our game in every platform with a Java virtual machine.

**2. REQUIREMENT ANALYSIS**

**2. 1. OVERVIEW**

At the base, the philosophy of the game is the same with all its ancestors: there is a snake which tries to eat the treats as much as possible and grows without biting himself. However, in this game, it is planned to add different treats that include shortening, poisonous, super snake, bonus point making, extending, upside down. The snake will follow various types of behaviors corresponding to the baits. The effects of the different baits on the snake are explained below;

**Regular Bait**: Snake follows usual behaviors; score gain is the regular point.

**Shortening Bait:** Snake's length decreases.

**Poisonous Bait:** Snake dies, game finishes.

**SuperSnake Bait**: The snake eats the baits within a given period of time without dying.

**BonusPoint Bait:** Within a period of time, the snake scores 5 times more points with each bait it eats.

**Extending Bait:** Snake's length increases.

**UpsideDown Bait:** Snake's head and tail change places, it starts move upside down.

The borders of the game panels will act as barriers for snake's movements. In regular state, the snake's life will end when its head crashes the walls.

**2. 2. FUNCTIONAL REQUIREMENTS**

**2.2.1. Playing the Game (Snake)**

When the player starts the game, the system initializes a game based on Snake’ rules. The game ends when the player quits or death condition is satisfied.

**2.2.2. Controlling Snake**

In the game, the player will control the movements of the Snake by using left and right buttons in the keyboard.

**2.2.3. Displaying Help**

Information about the game play, names of different colored baits and their function will be included in helpsection; the player can access the help section.

Help section consists of two sections:

• Control of Snake

• Information about entire game

With Displaying Help feature, the player will be able to read the game rules, learn the controls and learn about scoring and ending conditions of the game.

**2.3. NON-FUNCTIONAL REQUIREMENTS**

**2.3.1. Graphics**

The graphics of the game will provide a fun game play. There will not be flickering and other graphical problems to create a smooth gaming.

**2.3.2. Controls and Response Time**

The response time of the game will be reduced to a minimum level in order to provide the players a smooth controlling.

**2.3.3. Capacity**

The game supports only one player at a time. The proposed system has not a multiplayer base; therefore one player can play the game.

**2.3.4. Availability**

The target audience of the proposed game is not be specified, so all of the age groups will have access to the game if the game is published publicly.

**2.3.5. Menus**

The menus will include selecting game mode, start, exiting from the game.

**2.4. CONSTRAINTS**

**2.4.1. Implementation Language**

The implementation language of the Snake will be Java.

**2.4.2. Application Platform**

The program will be desktop application.

**2.4.3. Operating System**

The program will run on every operating system having JRE.

**2.5. SCENARIOS**

1. **Regular Movement Scenario (MoveTheSnake):** Game is running normally. Let’s assume snake is moving to upwards. Player presses the right button. Snake turn his face to its right and continue its path. After a while Player presses left button this time, Snake faces its left and continues there. Player uses right and left buttons to direct the Snake.
2. **Eating Bait Scenarios (EatBait):** Player makes the Snake to move to the bait. When the Snake’s head finally meets with bait, Snake eats the bait and bait disappears. Bait affects the Snake according to its type.
   1. **Regular Bait Scenario (EatRegularBait)**

If the Snake encounters the regularly spawned black bait, Snake will grow longer and Player gains normal points.

* 1. **Shortening Bait Scenario (EatShorteningBait)**

While Eating Baits scenario is active there are regular baits and there are special baits with each has different color. When player eats purple colored special bait which is called “shortening”, the length of the snake is reduced by 5.

* 1. **Bonus Point Bait Scenario (EatBonusPointBait)**

While Eating Baits scenario is active there are regular baits and there are

special baits with each has different color. When player eats a cyan colored

special bait which is called “bonus point”, number of seconds appears on the

screen for countdown. Until countdown finishes each eaten bait makes 5 times

points of the normal bait.

* 1. **Poisonous Bait Scenario (EatPoisonousBait)**

While Eating Baits scenario is active there are regular baits and there are

special baits with each has different color. When player eats a green colored

special bait which is called “poisonous”, it causes a death scenario which is the

end of the game.

* 1. **Extending Bait Scenario (EatExtendingBait)**

In this scenario, there are normal baits and there are special baits with each has different color. If snake eats the special bait with blue color which is called “extending bait”, its length increases by 5 points which is muk.

* 1. **Upside Down Bait Scenario (EatUpsideDownBait)**

Snake has a tail and a head. This scenario includes the special bait which is ‘upside-down bait’. If snake eats the yellow bait which is ‘upside-down bait’, its head will become its tail and its tail will become its head. Therefore its head and tail will be swapped and direction of snake will be the way which former tail has directed.

* 1. **SuperSnake Bait Scenario (EatSuperSnakeBait)**

This scenario has the special bait which is ‘Super Snake Bait’ with red colored bait. If snake eats this red colored bait, it will become invulnerable any deathful events like tossing a wall, eating poisonous bait. However, SuperSnake bait does not prevent self destruction if snake eats himself.

**3. Death Scenarios**

When the following scenarios occur, the Snake dies and the Game is over. The TotalScore is calculated by the latest score of the Snake.

* 1. **Self-Destruction Scenario (SelfDestruct)**

The Snake expresses its regular movements in the game. Player controls its movements by the buttons. When the control buttons causes Snake’s head to touch its body, it dies. Game finishes.

* 1. **Crashing the Wall Scenario (CrashBorder)**

The Snake expresses his regular movements. Player controls its movements by Left & Right buttons. When the Snake does not turn and its head reaches the borders of the game or when the Snake turns to the corresponding direction while moving near the borders; Snake’s head crashes the borders and it dies. Game finishes.

* 1. **Poisonous Bait Scenario (EatPoisonousBait)**

The Snake expresses its regular movements in the game. EatBait scenario is active in the game. A green bait (poisonous) is on the game screen. It eats the green bait. It dies. Game finishes.

**2.6. USE-CASE MODELS**

**Use Case Model for Regular Movement:**

**Name**: MoveTheSnake

**Actors**: User (Player in the scenario)

**Entry condition:** None

**Flow of Events:** Player presses the left or right button**.** Snake turn his heads according to the pressed button

**Exit condition:** Turning process is completed

**Exceptions:** None

**Special Requirements:** None

**Use Case Model for Eating Regular Bait:**

**Name**: EatRegularBait

**Actors**: Snake, Player

**Entry condition:** Snake’s head is over the black Regular bait. In other words, Snake eats the black bait.

**Flow of Events:** Snake'slength will increasein normal state.Normal points will be given.

**Exit condition**: One move after eating the bait.

**Exceptions:** None

**Special Requirements:** None

**Use Case Model for Eating Shortening Bait:**

**Name:** EatShorteningBait

**Actors:** Snake

**Entry Condition:** Snake’s head touches the purple bait.

**Flow of Events:** The Snake moves regularly on the screen, then snake approaches to the purple bait.

**Exit Condition:** One move after from touching the bait The Snake’s length is reduced by 5.

**Exceptions:** If the length of The Snake is less than (starting length + then the effect of the shortening bait is ignored. If the snake has eaten “SuperSnakeBait” before, then the effect of the shortening bait is ignored.

**Special Requirements:** None

**Use Case Model for Eating Bonus Point Bait:**

**Name:** EatBonusPointBait

**Actors:** Snake

**Entry Condition:** Snake’s head touches the cyan bait.

**Flow of Events:** The snake moves regularly on the screen, then snake approaches to the cyan bait and eats it.

**Exit Condition:** Number of seconds appears on the screen for countdown.

**Exceptions:** Until countdown finishes, the gain of every bait is raised 5 times.

**Special Requirements:** None

**Use Case Model for Eating Poisonous Bait:**

**Name:** EatPoisonousBait

**Actors:** Snake

**Entry Condition:** Snake’s head touches the green bait.

**Flow of Events:** The Snake moves regularly on the screen. Then Snake’s head approaches to the green bait and touches. The snake dies. Game is over. Total Score is calculated and printed on the screen.

**Exit Condition:** The Snake dies.

**Exceptions:** If the snake has eaten “super snake bait” before, then the effect of the shortening bait is ignored.

**Special Requirements:** None

**Use Case Model for Eating Extending Bait:**

**Name:** EatExtendingBait

**Actors:** Snake, Player

**Entry Condition:** Snake eats the blue bait which is extending bait

**Flow of Events:** If player eats the blue bait, snake’s length will be extended 5 points more compared to regular bait

**Exit Condition:** One move after the eating extending bait

**Exception:** None

**Special Requirements:** None

**Use Case Model for Eating Upside Down Bait:**

**Name:** EatUpsideDownBait

**Actors:** Snake, Player

**Entry Condition:** Snake eats the yellow bait which is upside-down bait

**Flow of Events:** If player eats the yellow bait, snake’s tail and its head will be replaced. After that, direction of snake will be direction of former tail.

**Exit Condition:** One move after the eating upside-down bait

**Exception:** None

**Special Requirements:** None

**Use Case Model for Eating SuperSnake Bait:**

**Name:** EatSuperSnakeBait

**Actors:** Snake, Player

**Entry Condition:** Snake eats the red bait which is SuperSnakeBait.

**Flow of Events:** While playing the game, if snake eats the red bait which is ‘SuperSnakeBait’.

**Exit Condition:** After ten second, snake will be no more invulnerable or against facing self destruction.

**Exception:** None

**Special Requirements:** None

**Use Case Model for Self Destruction:**

**Name:** SelfDestructSnake

**Actors:** Player

**Entry Condition:** Snake’s headtouches its body.

**Flow of Events:** The Snake moves regularly on the game screen. The Player makes the Snake’s head touch its body by controlling it with the left and right buttons of the keyboard. The Snake dies, game is over. TotalScore is calculated by the latest score of the Snake. TotalScore is printed on the screen.

**Exit Condition:**  the Snake dies.

**Exceptions:** None

**Special Requirements:** None

**Use Case Model for Crashing Border:**

**Name:** CrashBorder

**Actors:** Player

**Entry Condition:** Snake’s headtouches the border.

**Flow of Events:** The Snake moves regularly on the game screen. The Player makes the Snake’s head crush to the borders of the game screen by controlling it with the left and right buttons of the keyboard. The Snake dies, game is over. TotalScore is calculated by the latest score of the Snake. TotalScore is printed on the screen.

**Exit Condition:** The Snake dies.

**Exceptions:** When the Snake eats the red bait (SuperSnake), it enters into SuperSnakeBait Mode. In this mode, the Snake cannot die.

**Special Requirements:** None

**Use Case Model for Poisonous Bait:**

**Name:** EatPoisonousBait

**Actors:** Snake

**Entry Condition:** Snake’s headtouches green bait.

**Flow of Events:** The Snake moves regularly on the game screen. Snake’s head touches the green bait. The Snake dies, game is over. TotalScore is calculated by the latest score of the Snake. TotalScore is printed on the screen.

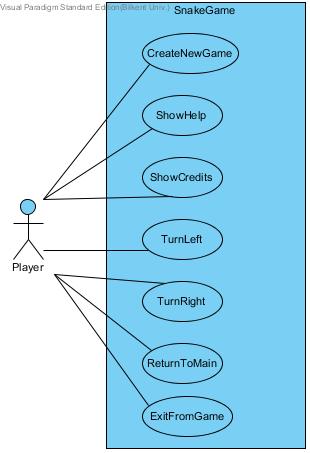
**Exit Condition:** The Snake dies.

**Exceptions:** When the Snake eats the red bait (SuperSnake), it enters into SuperSnakeBait Mode. In this mode, the Snake cannot die.

**Special Requirements:** None

**Use Case Model for the Game:**

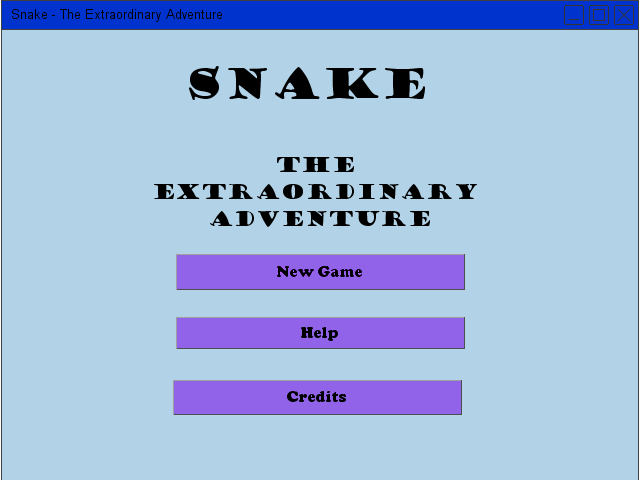
The following figure indicates the use case model for the game.



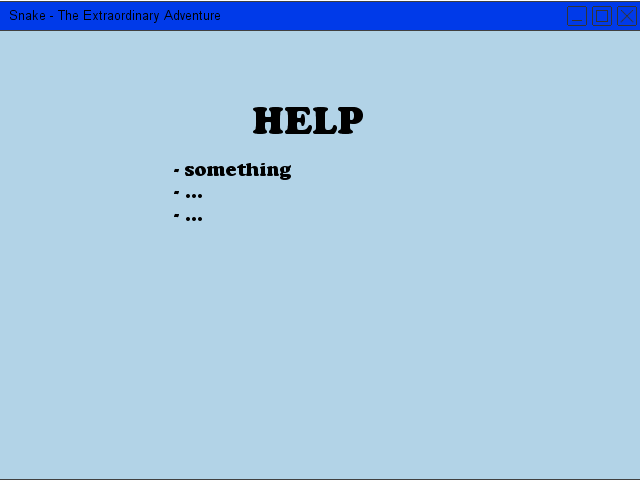
**2.7. USER INTERFACE**

The following figures represent the proposed user interface of the game, Snake - The Extraordinary Adventure.

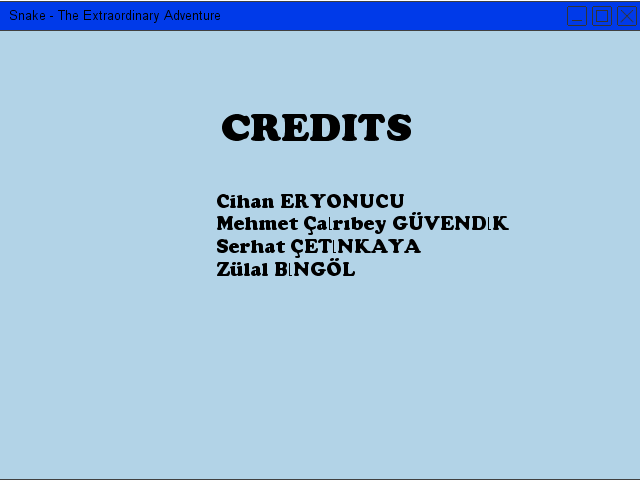
**Figure 1: GameBoard Panel**

****

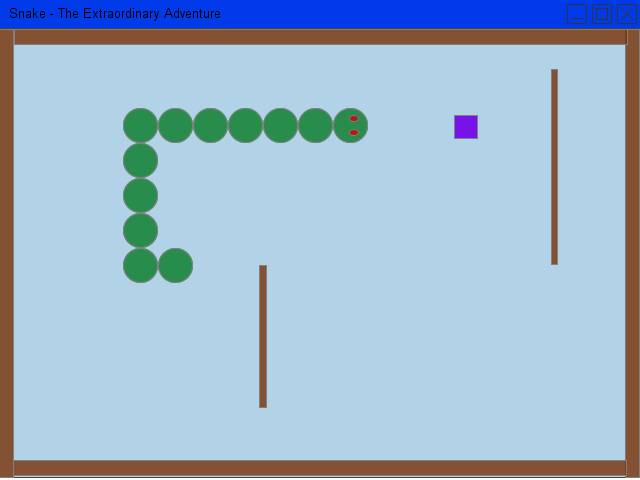
**Figure 2: Help Panel**

****

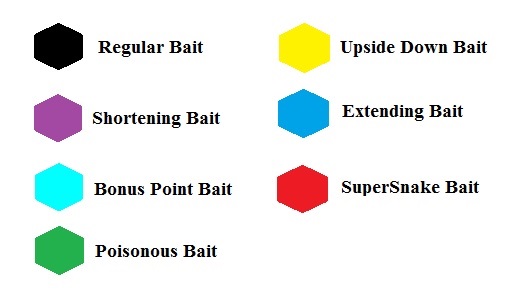
**Figure 3: Credits Panel**

****

**Figure 4: Game Panel**

****

Here are the colors of the baits which will be used in the user interface;



**3. ANALYSIS**

**3.1. OBJECT MODEL**

**3.1.1. DOMAIN LEXICON**

**Piece Class:** Piece class is one of the most important classes of the system. It is used in the body of the snake and the arena. Arena’s wall will be Piece’s. In addition snake will be consisting of Piece’s. They are both pieces because when there is collision of the piece’s snake will die whether it is because of the self-collision or collision between wall and snake.

**Arena Class:** Arena is basically the board that game will run.

**Snake Class:** Snake is consisting of Piece’s. It has the crucial methods in it such as die, eatBait and movement. It can be said that it is the core functional class of the game.

**Bait:** Bait is the food that the snake will try to eat the baits. This class is abstract it will be foundation for the baits.

**UpsideDown Bait class:** This bait will change the position of the head and tail. Score will be given normally.

**BonusPointBait Class:** This bait will give the player a 15 second time for the 5 times bonus point.

**RegularBait Class:** This will extend the snake’s length and give regular point.

**ShorteningBait Class:** This will shorten the snake’s length. Score will be given normally.

**SuperSnakeBait Class:** This will give the snake invulnerability to the wall deaths for 10 seconds. Score will be given normally.

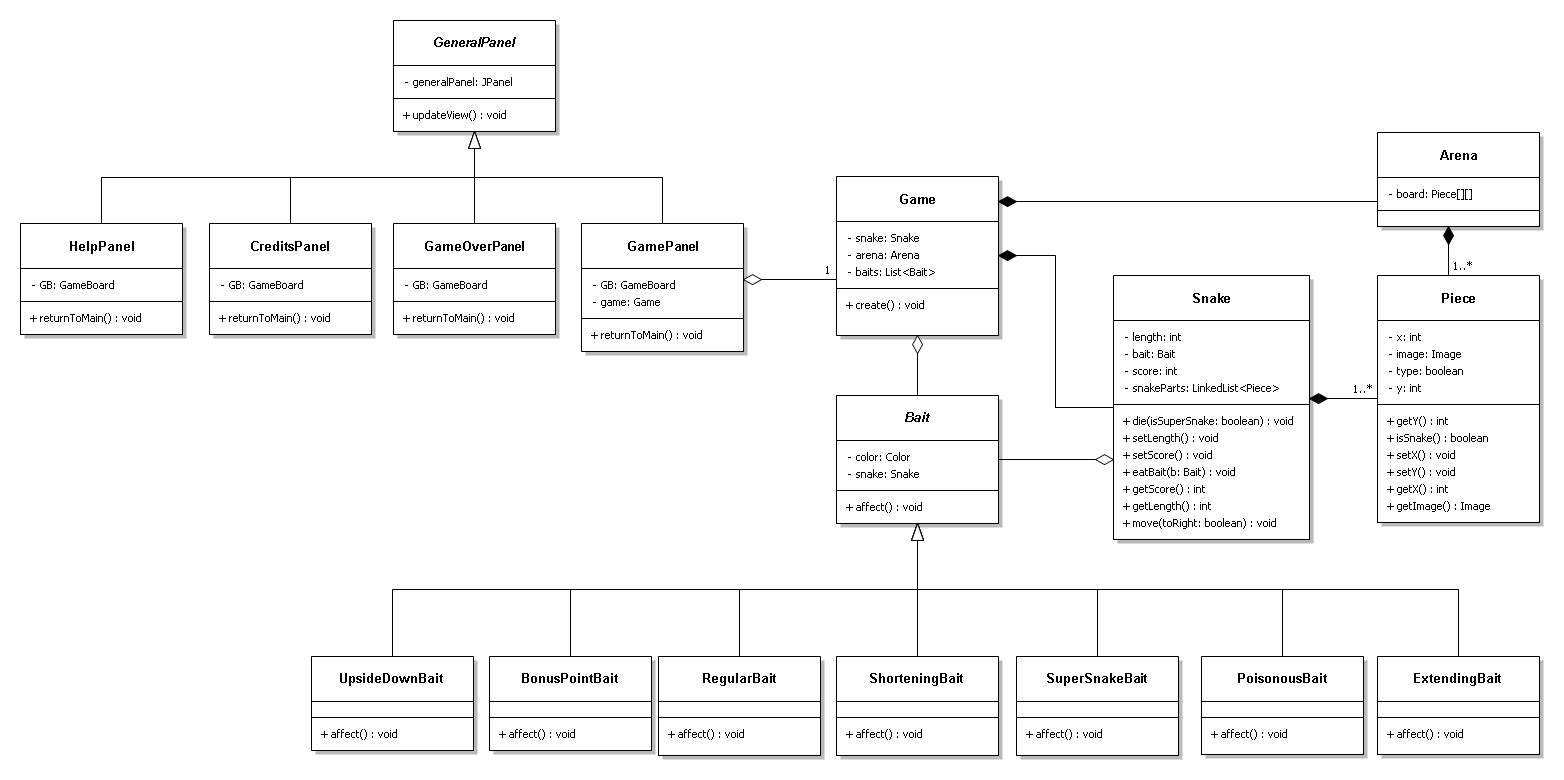
**PoisonousBait Class:** This bait will kill the snake and end the game.

**ExtendingBait Class:** This extends the snake’s length twice as much compared to the normal growth. Normal range of points will be given.

**3.1.2 CLASS DIAGRAMS**

The following diagrams show the class hierarchy in the game.

**Figure 5: Class Diagram for the Core of the Game and User Interface**

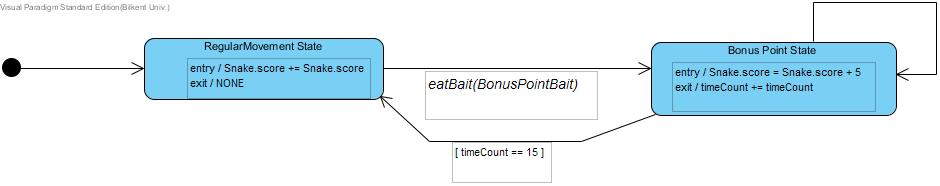


**3.2. DYNAMIC MODELS**

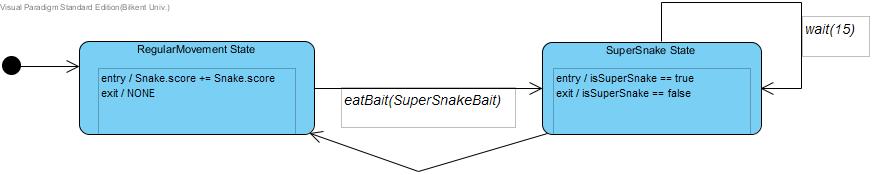
**3.2.1. STATE CHART DIAGRAMS**

The following figures illustrate the state chart diagrams given out of the proposed system.

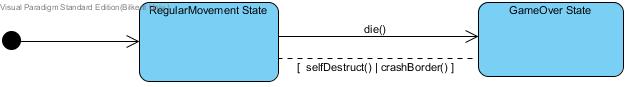
**Figure 7: BonusPointBait State Chart Diagram**



**Figure 8: SuperSnakeBait State Chart Diagram**

****

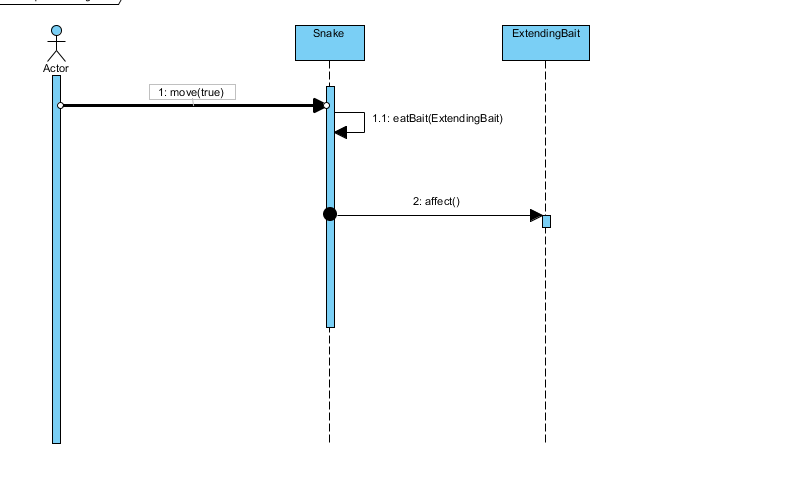
**Figure 9: State Chart Diagram for Death Positions**

****

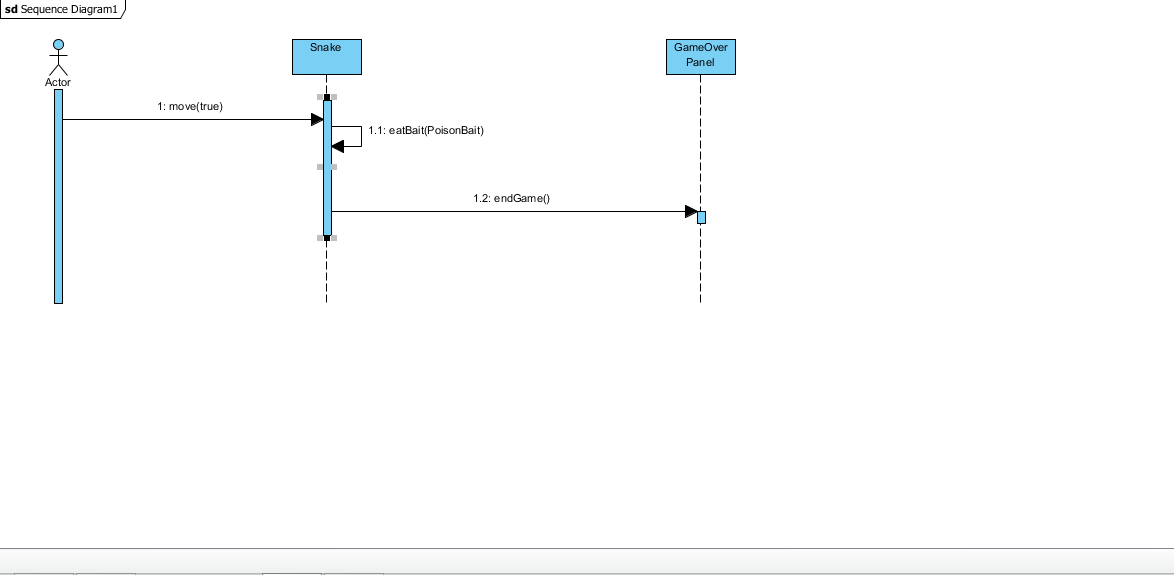
**3.2.2. SEQUENCE DIAGRAMS**

Here are the sequence diagrams which control the flow of the game progressively.

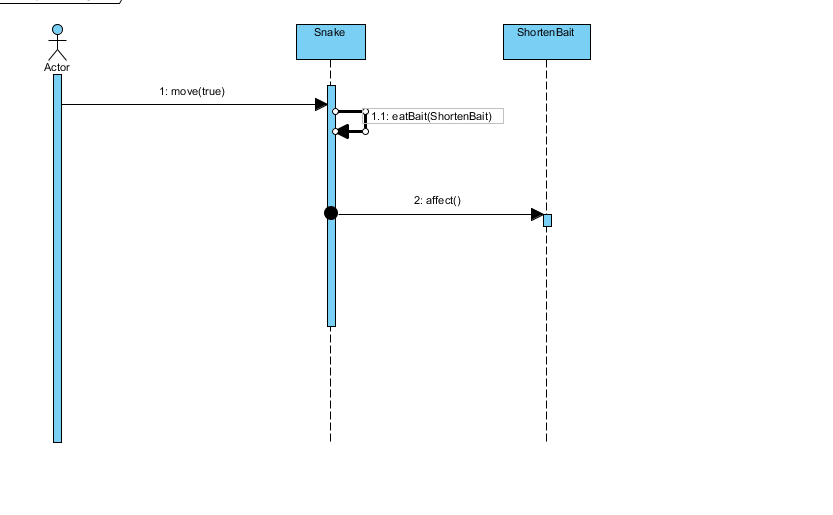
**Figure 10: ExtendingBait Sequence Diagram**



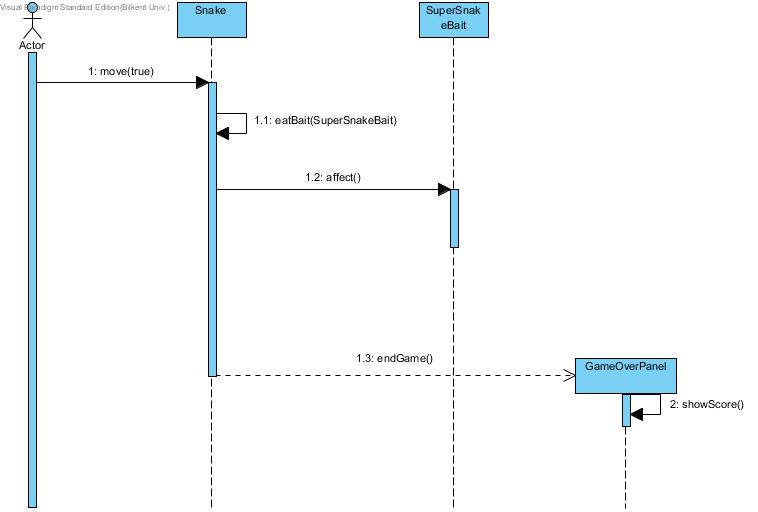
**Figure 11: PoisonousBait Sequence Diagram**

****

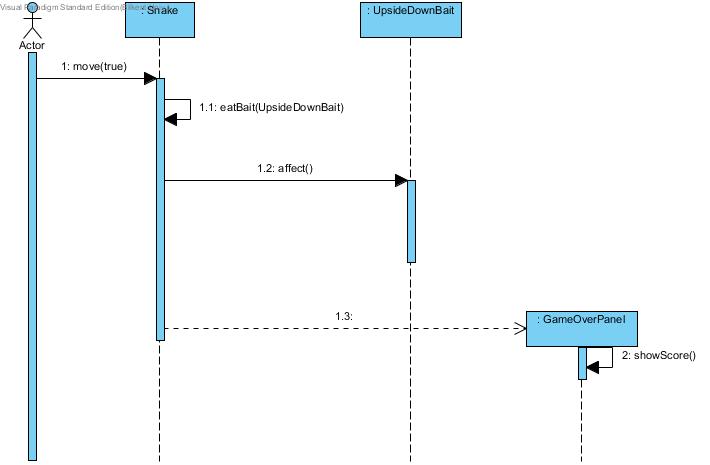
**Figure 12: ShorteningBait Sequence Diagram**

****

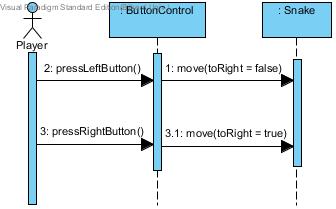
**Figure 13: SuperSnakeBait Sequence Diagram**

****

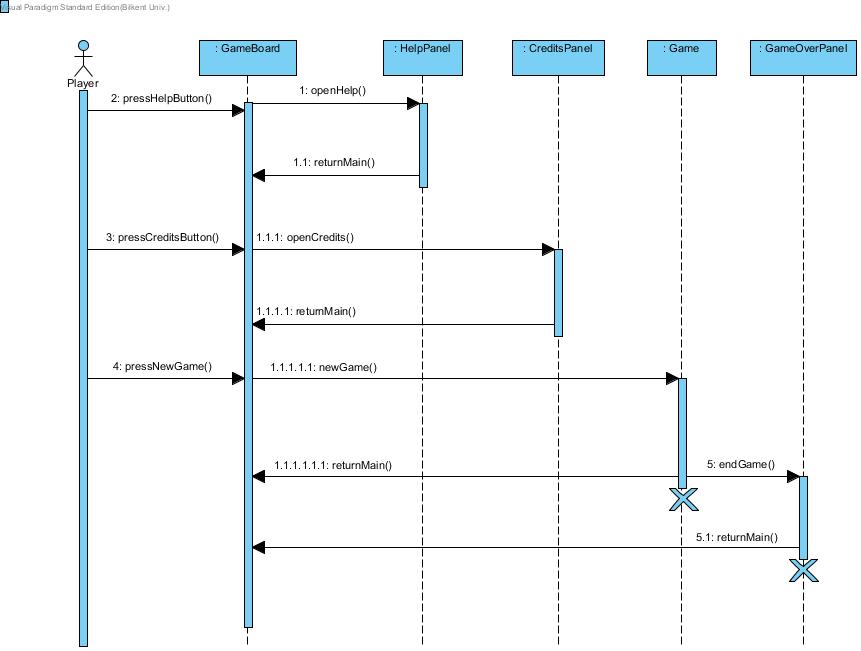
**Figure 14: UpsideDownBait Sequence Diagram**

****

**Figure 15: Sequence Diagram for Regular Movement**

****

**Figure 16: Sequence Diagram for Panel Transitions of the Game**

****

**4. Conclusions and Lessons Learned**

In the analysis part of the report, we tried to explain what our project is, what it does, which functionalities it has, what we aim to do and how we will do it. We understood what requirements, scenarios do. Firstly, we specify game features,functional and non-functional requirements. Functional and non-functional requirements, scenarios help us before considering use case diagrams. Additionally, we saw more detailed image of the project. We used case tools such as Visual Paradigm and using Unified Modeling Language(UML). UML diagrams are really help us to show how our program works. We created Use Case diagrams, Sequence diagrams, State diagrams by using Visual Paradigm.In the system design part of the report, we have explained design goals, architectural patterns, key concerns.

**Lessons Learned**

In this course, we learn how to design a project within a detailed documentation. In each phase of the project, we add up something. We changed some parts while thinking critically after each step. We also use the topics, which we have learnt in the class. For instance, we learn about functional and nonfunctional requirements, scenarios and we also learn how to make object modeling and dynamic modeling. While we are making object modeling we learn how to draw class diagram and in dynamic and object modeling through use case diagram, sequence diagram, class diagram, state diagram.In addition, we specified design goals, key concerns, architectural patterns and design patterns which are also discussed in lessons. We have also learnt how to apply patterns for our project.

**5 . SOFTWARES USED**

Softwares:

* Class & State Chart & Sequence Diagrams: Visual Paradigm 12.2

Standard Edition

NClass Standard Edition

* User Interface Models: ForeUI 4.00

Easynth Solution Inc. Ltd.