PROCEEDINGS

of the Union of Scientists - Ruse

Book 5

Mathematics, Informatics and Physics

Volume 13, 2016

RUSE

PROCEEDINGS
OF THE UNION OF SCIENTISTS - RUSE

Union of Scientists - Ruse
16, Konstantin Irechek Street
7000 Ruse
BULGARIA
Phone: (++359 82) 828 135,
(++359 82) 841 634
E-mail: suruse@uni-ruse.bg
web: suruse.uni-ruse.bg

Contacts with Editor
Phone: (++359 82) 888 738
E-mail: zzdravkova@uni-ruse.bg

PROCEEDINGS
of the Union of Scientists – Ruse

ISSN 1314-3077

The Ruse Branch of the Union of Scientists in Bulgaria was founded in 1956. Its first Chairman was Prof. Stoyan Petrov. He was followed by Prof. Trifon Georgiev, Prof. Kolyo Vasilev, Prof. Georgi Popov, Prof. Mityo Kanev, Assoc. Prof. Boris Borisov, Prof. Emil Marinov, Prof. Hristo Beloev. The individual members number nearly 300 recognized scientists from Ruse, organized in 13 scientific sections. There are several collective members too – organizations and companies from Ruse, known for their success in the field of science and higher education, or their applied research activities. The activities of the Union of Scientists – Ruse are numerous: scientific, educational and other humanitarian events directly related to hot issues in the development of Ruse region, including its infrastructure, environment, history and future development; commitment to the development of the scientific organizations in Ruse, the professional development and growth of the scientists and the protection of their individual rights.

The Union of Scientists – Ruse (US – Ruse) organizes publishing of scientific and popular informative literature, and since 1998 – the "Proceedings of the Union of Scientists- Ruse".

BOOK 5

"MATHEMATICS, INFORMATICS AND PHYSICS"

VOLUME 13

CONTENTS

Mathematics

Informatics

BOOK 5

"MATHEMATICS, INFORMATICS AND PHYSICS"

VOLUME 13

web: suruse.uni-ruse.bg

LOGICAL DATA PRE-PROCESSING FOR THE HEARTHSTONE GAME

Valentin Velikov, Iliya Mutafov

Angel Kanchev University of Ruse

Abstract: *This paper presents an approach and internal structure of an application for a preliminary database processing, connected with the game HEARTHSTONE. The main part of time for the game is spent for a difficult logical preparation, usually using a pencil and paper, which can be assisted or replaced with a computer application.*

Keywords: *computer application, HEARTHSTONE game, preliminary database processing.*

INTRODUCTION

Many card games consist of two or more stages and only the final one is the actual playing with cards. At the preliminary steps various preparations are carried out which refine the rules of the current game and using some logic one tries to study the opponent, to guess his cards, his experience, and way of thinking. Examples of such behavior are Contract Bridge, Bridge Beloit and others. Due to their popularity, some of these games were developed as applications.

One of the famous card games, with more than a million and a half registered users in the network, is the HEARTHSTONE. There are many levels, but most of the time is associated with huge preliminary logical preparation, where the player processes the known data mostly by using a pencil and paper. That is how the authors came up with the idea of creating various aids and tools to support the players.

DETAILED DESCRIPTION

1. Existing solutions for Deck Builder and Arena Simulator

1.1. Existing solutions for Deck Builder

In the world may be exist desktop applications with these functionalities, shared online in reddit.com/r/hearthstone, but they are not enough popular or they lost support from the authors.

More successful are the web-apps. Many Hearthstone game sites include **Deck Builder** as one of the own functionality or as a simulation tool. All they have many advantages but for us are more important the disadvantages and unsolved problems. Examples for these tools are:

- deck builder of Hearthpwn [13]. This site take care of all things connected with Hearthstone – news, articles, view points, discussions, stream tracking and so on. Their Deck builder is a separated tool. Disadvantages: the decks are saved on own server. Without network connection is impossible to use the decks; there are bugs in the Arena regime – the cards are shown as inaccessible if they are used more than twice; the color coding is not corresponding with the cards etc.
- deck builder of Hearthhead [14]. Hearthhead offer one of the best Hearthstone databases, so and many tools for simulations and sharing. Disadvantages: uneasy card representation – as infinity list like table. It is necessary to scrolling till the necessary cart,

for which is necessary long time; many and unnecessary tools; up to 30 cards limit; using an own library with a ling manual input of data in the site etc.

- deck builder of Tempostorm.[16] - Tempostorm is site of Hearthstone team with the same name. As the beginning he is created to popularity the register trade-mark “Reynad” by Andrey Yanyuk. Now the site offer articles and opinion of the best players in the world.. За съжаление инструментът им за създаване на тесте е направен за използване в тези ревюта, и това създава известни ограничения при споделяне. Disadvantages: uneasy card representation, bugs in the searching function, limit of the number of duplicated cards in the deck etc.
- deck builder of HSDeck [15] - Disadvantages: the engraving of the separated cards are created separately and hey are not corresponding with the game; the cards are not color coded – there is not difference between neutral cards and the cards from the classes; the text in the cards is overburdened; there are used irrelevantly tooltips in the list; “flying” elements etc.

1.2. Existing solutions for Deck Arena Simulator

For difference by **deck builder** tools, arena simulators are not so popular because of their limited functionality. The arena simulators offer enacted of the possible scenario only which is with low possibility in the real game. Examples for arena simulators:

- Arena Simulator of Hearthhead – Arena practice [11] - the advantages and disadvantages concur with the same of deck builder, excluding the specific (filters, for example);
- Arena Simulator of Hearthpwn [12] – Like hearthhead, the hearthpwn have available with the same two tools, they have the same aspect and the same resources, so advantages and disadvantages of the arena simulator are the same as the builder;
- Arena Simulator of arenavalue [10] – the site arenavalue is destined for arena aspect. So it support special tools only. Disadvantages in connection with the specific tools, the user has to confirm after every choice (button click to continue) instead of automatically continue to next choice. Moreover if the choice is not confirm he can not be changed.

2. Basic game rules

Hearthstone: Heroes of WarCraft in all respects follows the classic principle of Blizzard: the game is easy to learn and difficult to master. The rules are simple and quick to absorb [8], [1], [2]. On the battlefield there are two heroes (classes), aiming to reduce the opponent's health from 30 to 0. The choice of heroes is large enough: druid, hunter, mage, paladin, priest, rogue, shaman, warlock, warrior. Participation in a battle gives experience through which the hero goes a level up. Going several levels up unlocks new cards. The classes differ in their special skill and a set of cards that can be placed only in the deck of the respective hero. Each deck contains 30 cards that the player chooses himself before going into battle. Different heroes have a different style of play, but it is not necessarily to use this. For example: for the skills of a **rogue** the need of **combo** is typical, i.e. to play at least one more card before playing this skill.

In his own deck the player can place up to two copies of the same card. The cards are divided into 5 groups according to their rarity: basic, common, rare, epic and legendary. However, there is no card, against which there are no dozens of counteractions, regardless of its power. Each card belongs to one of three categories: weapons, charms and minions.

3. Designing of the system

The application, called "Deck Builder", is designed for users and fans of the game [7]. His advantage over the built-in functions of the game is that it is quite "light" as application in comparison with the original game. It has the whole database of cards existing in the game and each of them can be used to create a deck. For comparison: in the game the player can used cards, which he already owns (i.e. either bought or unlocked them using the game tools) [4], [3] and there is no available simulator to create a future deck. Another limitation that is decided: the authors have put a ceiling on the maximum number of cards [6] which may be located in the deck - 30. This is a problem because the best players build their strategy by adding all the necessary cards (in their opinion), and then they begin to remove the least necessary (according to the given situation) until they reach the required number of 30.

3.1. Choosing a programming environment:

The application was developed using Java FX – a set of graphics and media packages for creating Rich Internet Applications which is a further development of Java Swing [9], [7].

3.2. General application diagram:

The general application diagram is shown in Figure 1.

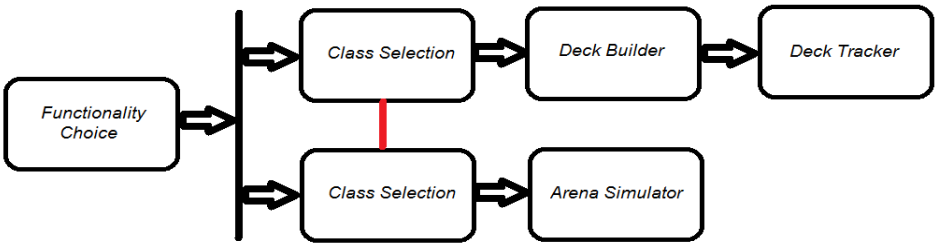

Fig. 1. General application diagram

3.3. Structures for data representation. Data interconnections.

A new class, called **Card,** was created, which represents a card in the game (Figure 2), and it keeps information about the properties of each game card.

Each attribute of the class corresponds to the leading feature of the card. It should be noted that each attribute is of type **String**, except attributes **atk** and **hp**, which are of type of **byte** in order to use them in future for additional features of the application. The attributes are:

- cClass – the play card class. The various classes have unique cards which can not be used by other classes. To distinguish them more easily, cards of different classes have specific class-coloring. The cards in Figure 2 are:
  - {1} – class Warrior: the presented card is a **weapon**, and all cards of this type are black. The class, to which the weapon belongs, can be identified by the thin frame surrounding the text on the card – on the figure this is not visible;
  - {2} – class Paladin: recognizable by the yellow color of the card;
  - {3} – neutral: in gray color – the neutral cards can be used in any deck, regardless of the class.

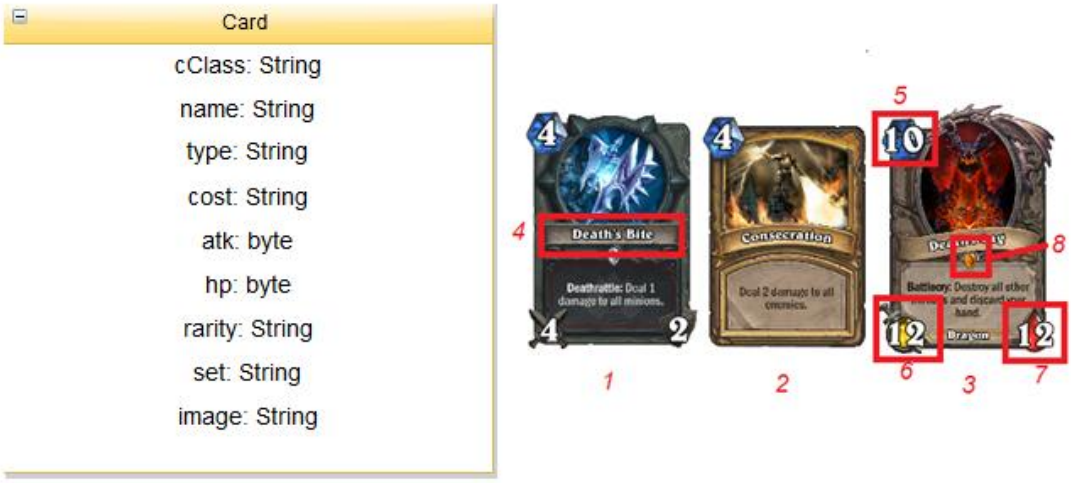

Fig. 2. Frame for a card representation

- name – the name of the card {4};
- type – type of the card. In fig.2, the cards are: (1) – **weapon**, (2) – **spell**, (3) – **minion**;
- cost – how much of the resource **mana** the card requires to be played {5};
- atk – attacking points {6};
- hp – health points {7};
- rarity – the rarity of the card {8} – it is marked with crystal, painted under the name on the card and means how rare it is, i.e. how likely it is to appear when purchasing and opening a new cards pack. If the crystal is absent, then this card is of the so called **Basic Set**, i.e. the set of "free" cards, available to each player;
- set – to which set the card belongs. This is an encyclopaedic property which is not expressed, and it is added for a future use
- image – the overall image of the card.

Since for the work of the program it is necessary to store data for over 500 different cards in the same data structure which must change its size dynamically, it is suitable to use the an **ArrayList** (figure 3). Several **ArrayLists** are used, some of them are just buffers or intermediate – they are used in shuffling or sorting data.

When launching a functionality of the application, all cards are stored as objects of class **Card** into **ArrayList** of type **Card**: **ArrayList<Card>** ***cards*** of class **DataStrings**. This data is copied into an **ArrayList** with the same name in the controller class of the called functionality and the copied **ArrayList** is used as main.

In the **Deck Builder** functionality, the data from main **ArrayList** is divided, depending on the chosen game class, in two separate **ArrayLists** – **classCards** and **neutralCards**, which are also of type **Card** (Fig.3).

The string attribute **image** is taken from the two **ArrayLists** and it is used to find the path to the image which is inserted in the appropriate album.

Besides, the names of all cards are divided into 4 **ArrayLists** (fig.4) of class **DataStrings** with type **String** - **basic**, **rare**, **epic**, **legendary**, which are used as a basis for distinguishing the cards by their property **rarity**.

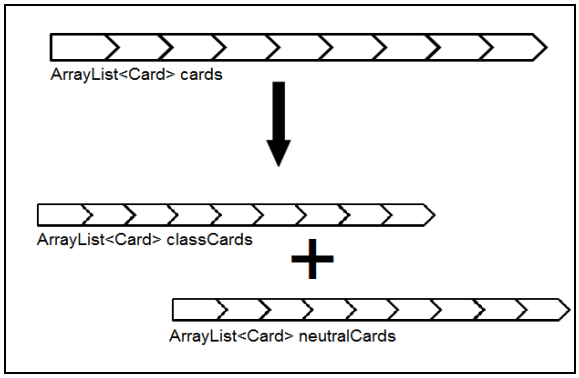

Fig. 3. Separating the cards into 2 lists depending on the album

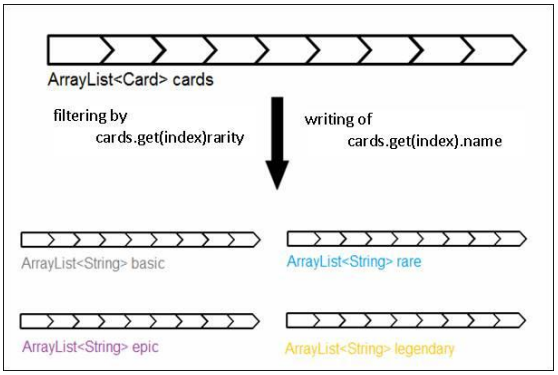

Fig. 4. Separating the cards into 4 lists depending on the property rarity

The colors of the scheme are consistent with distinctive colors for the rarity of the cards.

To add the cards to the list, the data for the selected card are transferred from the main **ArrayList** to **ObservableList**, as well as to a new **ArrayList**.

ObservableList<String> ***dataGet*** collects only the names of the added by the user cards (i.e. **Card.name**), and numbers are added into **ArrayList<String>** ***multiples***, indicating how many times the corresponding card was chosen. The data of **dataGet** go through next **ObservableList<String>** ***tempList***, where they are sorted, and delivered to the last **ObservableList<String>** ***data,*** to be used for visualizing the data in the list being creating (fig.5).

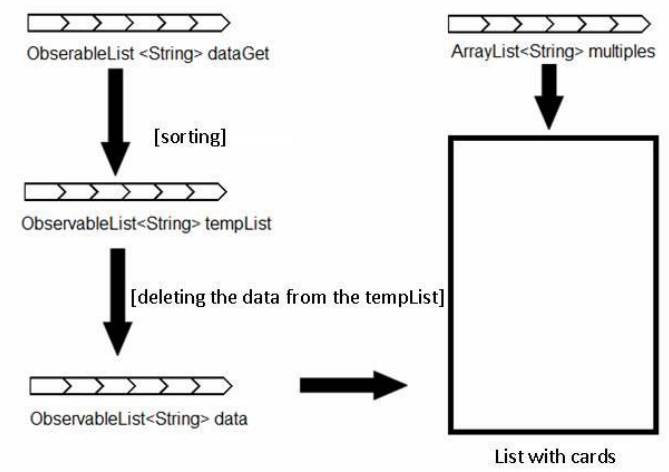

Fig. 5. Functional links between major lists

When the **Deck Tracker** functionality is started the data, used for constructing a deck, is copied (fig.6). This copy is needed to keep the old data, and to distinguish the data from both functionalities, so that there is no confusion when closing or restarting. This data is copied again, and it serves as primary data which replace the original data when restarting the list when a new duel begins.

New buffer structures are also added, which carry out the same functions as those used in the **Deck Builder**.

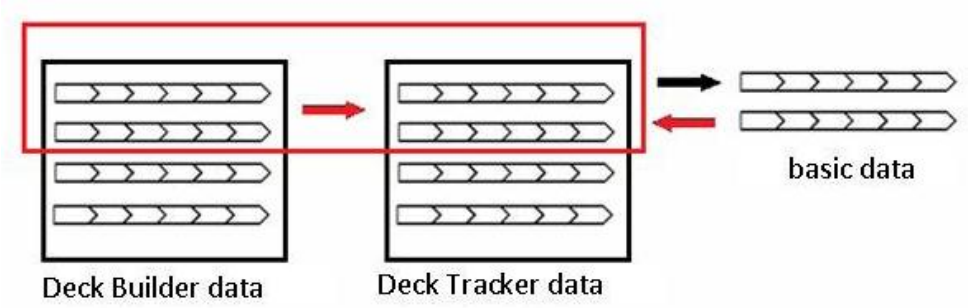

Fig. 6. Copying the data from **Deck Builder** by starting of **Deck Tracker**

In **Arena Simulator** functionality after the creation of the main **ArrayList** the cards are separated into 4 **ArrayLists** of type **Card** – **whiteCards**, **blueCards**, **purpleCards** and **orangeCards**. The separation is performed according to the **rarity** property, and certain cards, which according to the **Blizzard's** rules are not used in the **Arena,** are excluded (fig.7).

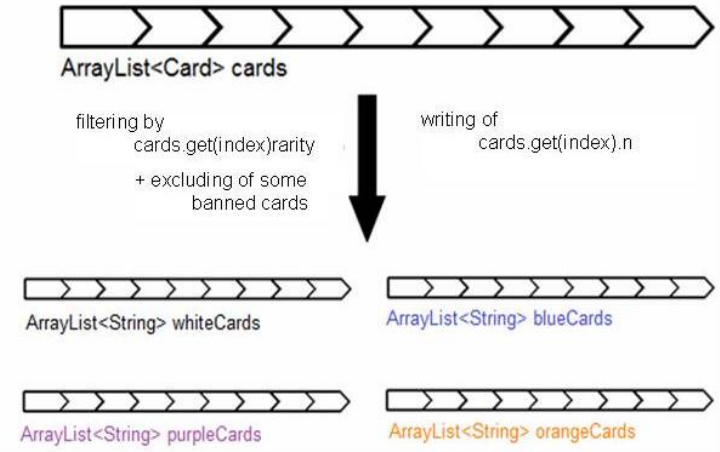

Fig. 7. Separating the cards by property **rarity** at the start of Arena Simulator

Like in the **Deck Builder** functionality the same **ObservableLists** of type **String** are used – **dataGet**, **tempList** and **data**, as well as **ArrayList<String> multiples**, but a new one **ObservableList<String> costs** is also used, which records the attributes **cost** of the added cards to be used to create a bar chart (fig.8).

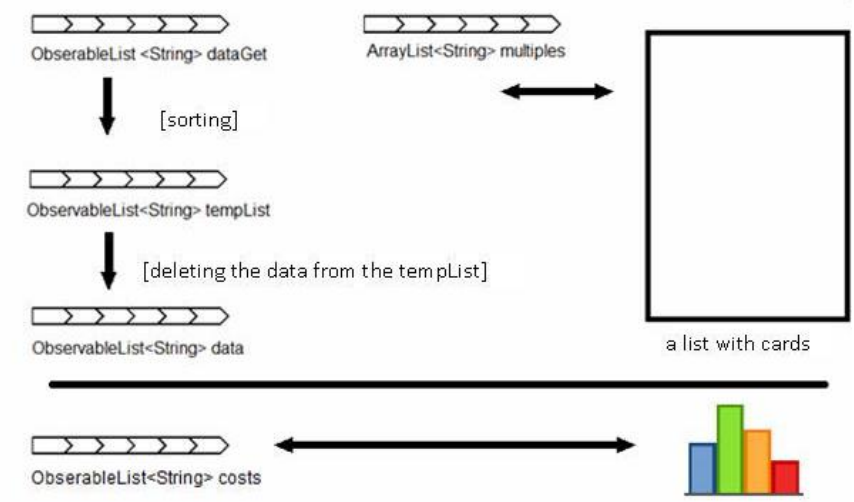

Fig. 8. In **Arena Simulator** is used an additional **ArrayList costs** to creating of bar chart

Random numbers are used to generate cards, from which the player can choose. First an integer from 1 to 100 is generated - **int rarityNum**, which determines the **ArrayList** in which the data, used for card generation, will be found. Once this is determined, the selected **ArrayList** is copied to a new one named **temp**. Then new three numbers are generated in the range from 1 to the number of cards in the selected **ArrayList**. These numbers should not be identical to each other, and determine which cards will be displayed.

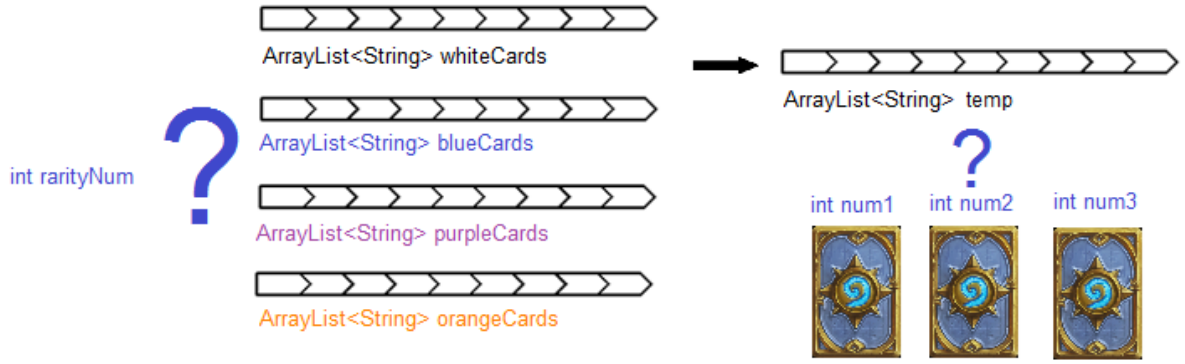

Fig. 9. Generation of cards for choice from the player

4. Application user interface

The successful user interface, fully resembling the original game (Fig. 10/11), is described in detail in [7]. Additional utilities are presented, which support the logical process of constructing the initial cards deck of with statistics, colours and other visual effects (Fig. 12/13).

Fig. 10. A window for selection of game class

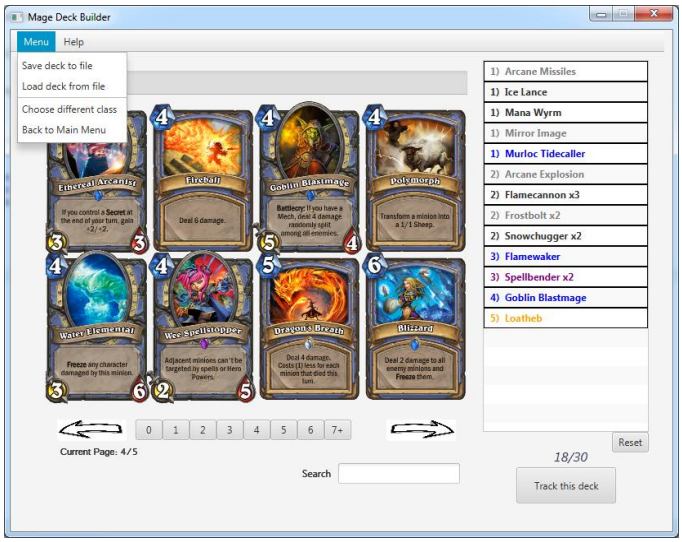

Fig. 11. Window of **Deck Builder** functionality

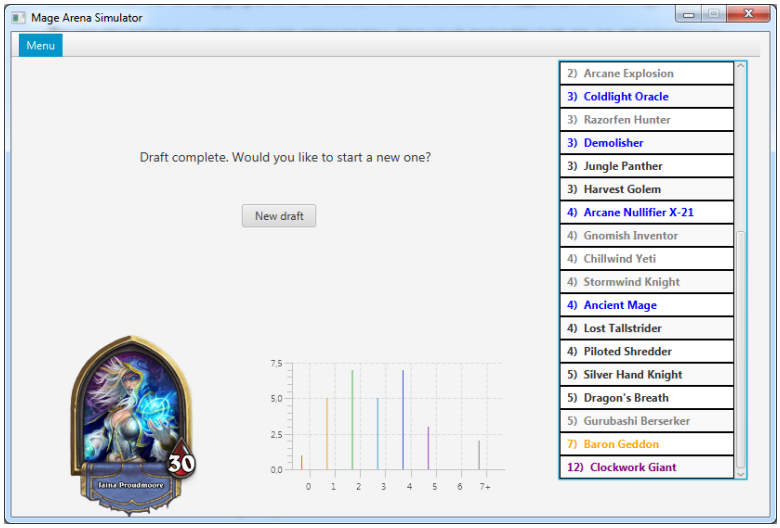

Fig. 12. Possible final part of Arena Simulator

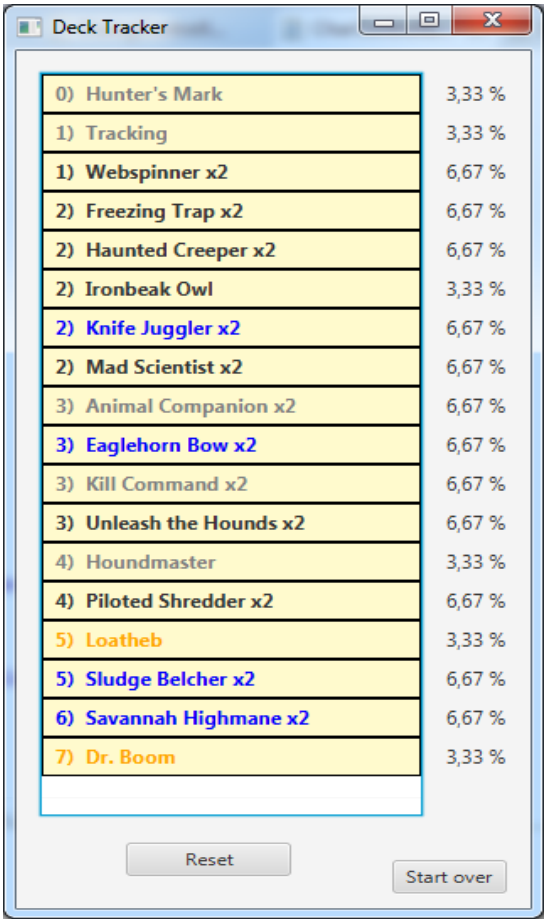

Fig. 13. Window functionality **Deck Tracker**

CONCLUSION

An application with gaming elements is created as a desktop system, which eases the logical part during the preliminary preparation of the cards for the HEARTHSTONE game. The pleasant and intuitive interface, as well as overcoming the shortcomings detected in other implementations [3], [4], [5], make this application attractive and competitive. It can be helpful to over a million and half registered users of the game [3], [6].

REFERENCES

[1] Alexander U. Adler, Hearthstone Card Rarity in the Arena (Original and Naxxramas), Alexadler.net, 25.03.2015, http://blog.alexadler.net/?p=5, 24.05.2015.

[2] Arena Value Arena Practice Tool, Arena Value, http://www.arenavalue.com/, 24.05.2015.

[3] Hearthhead Deck Builder, Hearthhead, ZAM, http://www.hearthhead.com/ ,24.05.2015.

[4] Hearthpwn Deck Builder, Hearthpwn, Curse Inc, http://www.hearthpwn.com/, 24.05.2015.

[5] HSDeck Deck Builder, HSDeck, HSDeck.com, http://www.hsdeck.com/, 24.05.2015.

[6] Tempostorm Deck Builder, Tempostorm, https://tempostorm.com/, 24.05.2015.

[7] Великов В., И. Мутафов. Приложение за логическа обработка на данните към играта HEARTHSTONE. В: Научни трудове на РУ, 2015, т.54, с.6.1, Русе, Русенски университет, 2015.
[8] Митев Г., Hearthstone: Heroes of WarCraft - PCM2.0, 23.05.2014, http://pcmania.bg/, 24.05.2015.
[9] Национална академия по разработка на софтуер, JavaFX – технология за създаване на RIA, НАРС, http://nars.bg/newsletter/javafx/, 24.05.2015.
[10] Arena Simulator на arenavalue - http://www.arenavalue.com/practice, 2016.
[11] Arena Simulator на Hearthhead - http://www.hearthhead.com/arena, 2016.
[12] Arena Simulator на Hearthpwn - http://www.hearthpwn.com/arena, 2016.
[13] deck builder на Hearthpwn - http://www.hearthpwn.com/deckbuilder , 2016.
[14] deck builder на Hearthhead - http://www.hearthhead.com/deckbuilder, 2016.
[15] deck builder на HSDeck. - http://www.hsdeck.com/deck/, 2016.
[16] deck builder на Tempostorm. - https://tempostorm.com/deck-builder, 2016.

CONTACT ADDRESSES

Pr. Assist. Valentin Velikov,PhD
Department of Informatics and Information Technologies
Faculty of Natural Sciences and Education
Angel Kanchev University of Ruse
8 Studentska Str., 7017 Ruse,Bulgaria
Phone: (++359 82) 888 326,
Cell Phone: (++359) 886 011 544,
E-mail: val @ ami. uni-ruse. bg

Iliya Mutafov, BSc student
Department of Informatics and Information Technologies
Faculty of Natural Sciences and Education
Angel Kanchev University of Ruse
8 Studentska Str., 7017 Ruse,Bulgaria
Cell Phone: (++359) 884050191
E-mail: xchokleet@gmail.com

ВЪТРЕШНА СТРУКТУРА НА ПРИЛОЖЕНИЕТО ЗА ЛОГИЧЕСКА ОБРАБОТКА НА ДАННИТЕ КЪМ ИГРАТА HEARTHSTONE

Валентин Великов, Илия Мутафов

Русенски университет „Ангел Кънчев”

Резюме: *Статията представя вътрешната структура на приложението за предварителна подготовка на данните, свързано с играта HEARTHSTONE. Основна част от времето в играта е свързано с огромна предварителна логическа подготовка, най-често „с лист и молив” на известните данни, което може да бъде подпомогнато или заменено от компютърно приложение.*

Ключови думи*: компютърно приложение, играта HEARTHSTONE, предварителна логическа подготовка на данните.*

ISSN 1314-3077