University of Crete, Department of Computer Science
HY252 – Object-Oriented Programming (Winter semester 2017-2018)
Teacher: **Yiannis Tzitzikas**
Topic: **Individual course work (project).**
Assistants: M. Moudantonakis (head) and the other assistants

General Instructions and Description of the Work Deliverables

Phases

- **Phase A: 10% of final grade** (clause 5 for passing the course)
  - o November 20 - December 8, 2017
- **Phase B: 15% of final grade** (clause 5 for passing the course)
  - o December 9 - December 23, 2017 (5% bonus) o
  - December 9 - January 8, 2018 (no bonus)

Extensions can only be granted if deemed appropriate and a maximum of 3 days for Phase A and Phase B!

Phase A

In this phase, the design of the application should be done based on the concepts and principles of object-oriented programming that you have been taught. The result of this phase is the definition of the objects, their attributes, and their behavior needed to represent the states and functions of the work topic as described in the utterance. Deliverables of this phase are:

- **Written report** (not in Greek) which will describe the above elements and present the implementation plan of the programming work, so that the passage to the next phase of implementation is ready. **UML class diagrams** should also be included .

- **Source code** that includes the **interfaces** (interfaces) and class outline (class outline) of your program's Java, accompanied by the necessary **javadoc** comments, which will guide the implementation of the next phase.

Briefly, the most important tasks to be done in this phase are:

- Identify the classes and interfaces for each small and large component of the program. Identifying the responsibilities of each class and its possible relationships with others.

- Find the attributes and methods of each class. • Finding the behavior (behaviour) of each class and interface, as well as the communication through messages (method calls) they need to have with each other. • Organization of classes in hierarchies with the goal of maximum reuse of your code.
- For each class that implements an interface, provide the signatures for all methods and the preconditions, postconditions, invariants that govern them in the form of javadoc comments.

IMPORTANT: Note that the more complete and detailed work you do in the design, the more correct, easy and successful the implementation will be. Because in previous years we noticed that those who had not done well in Phase A rarely did well in Phase B, from last year

there is clause 5 in Phase A as well, hopefully this will help you manage your time better and maximize the benefits of the course and the final grade you will achieve.

UML Plugins

For *Eclipse* you can use ObjectAid. You need to register to use it: http://www.objectaid.com/install-license. Installation instructions can be found here: http://www.objectaid.com/installation. For class diagrams see http://www.objectaid.com/class-diagram. Other tools for eclipse can be found at: http://stackoverflow.com/questions/15097766/eclipse-plugin-for-generating-uml-class-from-java-project .

For *Netbeans* (Version >= 8.0) detailed instructions for the easyUML plugin can be found here https://elearn.uoc.gr/pluginfile.php/27783/mod_forum/attachment/10048/uml.7z . Alternatively, for Netbeans one can also use visual paradigm http://www.visual-paradigm.com/product/vpuml/tutorials/modelinginnetbeans.jsp .

Detailed grading for phase A

Deliverables of this phase are:

Deliverable	Units
Written report (not in Greek) which will describe the above elements and present the implementation plan of the programming work, so that the transition to the next phase of implementation is ready	20%
UML class diagrams should also be included and an explanation of what each diagram represents.	10%
Division of work according to the MVC model (Model View Controller) For	15%
each class that implements an interface provide signatures for all methods and pre-conditions, post-conditions and invariants respectively) that govern them in the form of javadoc comments.	15%
Organize classes into hierarchies to maximize code reuse. Use of abstract classes and/or interfaces for the basic classes (Card, Finding).	15%
The necessary classes and methods must exist (25%) Player, Position, Card, Controller There must be the methods for the basic elements of the game Indicatively some basic methods: Method to initialize the game Methods for the moves made in every possible position Methods for when a player discards a card Methods for when the player arrives at a location with an archaeological find Method that determines order (which player plays) Method that checks if we have a winner	25%

References that only copy paste the code from netbeans/eclipse with comments will get a low grade (half a grade and less depending on the case) so as not to offend the students who made a good report. A good report should clearly have your information and be organized into sections (it's good to have numbered pages and a table of contents at the beginning).

Phase B

In this phase, the main implementation of the application must be done, based on the previous design (phase A). It is not mandatory to use the phase A design as it is, as some design choices prove to be in need of revision along the way. However, the final score will also depend on the consistency of the final implementation with respect to the original design. At this stage, deliverables are:

• the **source code** that implements the task
• step-by-step instructions on how your program compiles and runs (README, Ant, Maven
etc) • report, which will analyze:
o the final design of the application, o
what changes were made in relation to the design of phase A (and why), o
the algorithms used
o any variations in the rules in relation to the rules given above
o the design and programming decisions made and how this reflects on the end user (eg ease/difficulty of handling) o the problems faced

o JUnit tests made for correctness checking
o generally anything else you deem necessary to mention (such as for example what maybe you didn't).

Job Rating

Your assignment will be graded on: • whether (and how much) the application design applies the object-oriented programming concepts and techniques you learned in the course
• whether (and how much) the mandatory functions of the application were
implemented • the completeness of the final report, which will record and document the design and implementation of the application
The grading method is described in detail on the next page.

For clarifications regarding the above task, you can send messages with your questions to the relevant **forum** on the moodle website. Questions sent to the course list hy252-list@csd.uoc.gr **they will not be answered.**

Good work

Detailed grading for phase B

The detailed grading for Phase B of the Project will be as follows:

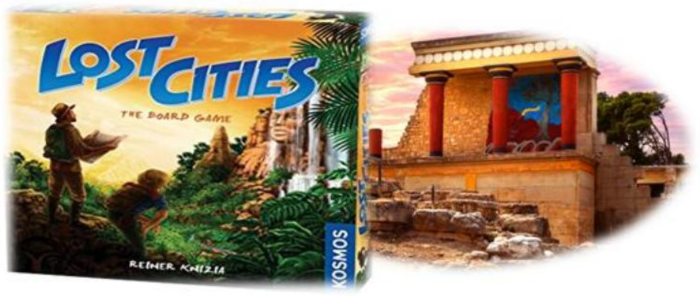
Deliverable	**Units**
Reference	5
Javadoc Comments (Pre – Post conditions)	5
JUnit Tests	5
UML Diagrams based on the final version and their explanation	5
Functionality-Graphics	
Initializing Board and Dealing Cards Keeping Turn	12
Moving Pawns	8
+ Playing Value Cards (1-10) Mythos Ariadne and	20
Minotaur Cards (and Power of Theseus) Archaeological Finds End of	10
Game Score Calculation -	12
Winner Historical	6
Information about Minoan	8
Civilization Save Game (Bonus)	4
	5

<u>IMPORTANT:</u> The minimum implementation needed to get a score of 50/100 is to have a GUI, initialize the board, pawns and cards, and move a pawn on the board. The above assumes that there is communication between the graphical interface and the model through the controller and there is sufficient reference, UML diagram comments and JUnit tests.

University of Crete, Department of Computer Science
HY252 – Object-Oriented Programming (Winter semester 2017-2018)
Teacher: **Yiannis Tzitzikas**
Topic: **Individual course work (project).**
Assistants: M. Moutantonakis (head) and the other assistants

In Search of the Lost Minoan Palaces

Project



Educational Objectives

- System specification and design
- Specification of Abstract Data Types (ABDs) required for successful completion of the system
- Implementation of components (ATD) of the system whose specification is given
- Use of inheritance and polymorphism
- Creating a Graphical Interface
- Dependency of the system core from the GUI
- Reuse interfaces and classes Use JFC (Java Collection Framework)
- Documentation, Control

Brief Job Description

Target. In this assignment you are asked to design and implement a variant of it "Lost States" board game entitled "Searching for Minoan Palaces".

The work is individual and you are not allowed to use code that you have not written yourself. If a duplicate is found the job will be zeroed out.

Description and Purpose of the Game.

Archaeologists believe that the destruction of the Minoan civilization was caused by the violent eruption of the Santorini volcano. Specifically, it is believed that the eruption of the volcano took place in 1450 BC and for this reason the Minoan Palaces were destroyed. His purpose game is to discover the 4 great palaces of the Minoan Civilization that were lost due to the eruption of the volcano.

The game you will implement consists of 2 players. Each player has 4 pawns: 3 archaeologists and a "Theseus" searching for his four great "lost palaces"
Minoan culture. To reach a lost palace one must travel a path of nine steps. On his turn a player may discard a card (with values ranging from 1 to 10) and move either an archaeologist or Theseus one step. The color of the card indicates the path to follow. The player should try to throw a card with a low value at the beginning, since to move the same pawn again he must play a card of equal or higher value. Each player should have each of their 4 pawns on a different path, and it is not necessary to use all of their pawns. The object of the game is for each player to get their pawns as far as possible to gain more points. The winner is

player who will collect the most points.

Detailed Description of the Work

Game Contents

The requested game consists of:

• 2 players
• 4 pawns for each player
  o 3 Archaeologists
  o 1 Theseus
• 1 board with 4 paths of 9 steps dimension 4x9 (4 rows-9 columns)
• 100 Cards
  o 20 cards for each palace with values 1-10 (each value appears 2 times)
  o 12 Ariadne myth cards (3 for each palace)
  o 8 Minotaur cards (2 for each palace) • 20
findings divided into o 4 rare findings
  (Disc of Phaistos, Ring of Minos (Knossos), Hair Jewelry,
    Ryto Zakrou)
  o 10 statuettes of the "Goddess of Snakes"
  o 6 “Murals” (each mural has a different value)

Preparing the game

Game initialization includes

• choosing which player will play first (e.g. random),
• the initialization of the board with the 4 paths of palaces,
• initializing the findings and placing them randomly on the board in positions 2,4,6,8,9 of each palace (pay attention to the limitations for rare findings),
• initializing the stack of 100 cards and shuffling them,
• dealing 8 cards to each player.

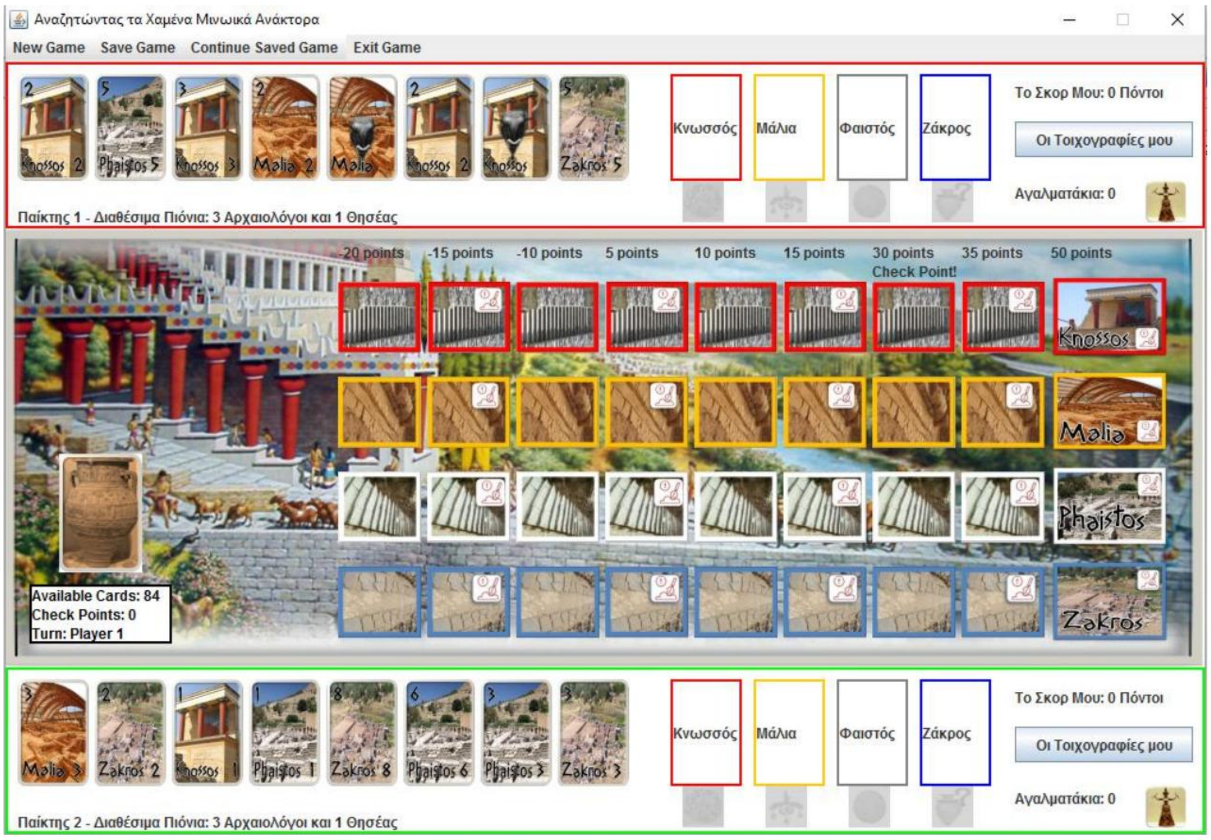
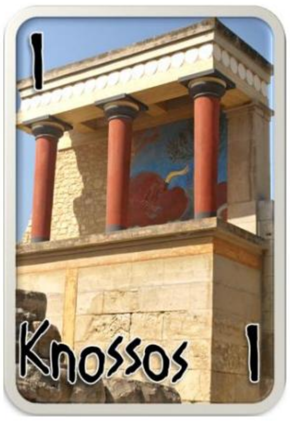
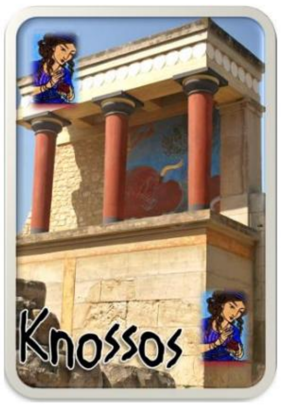
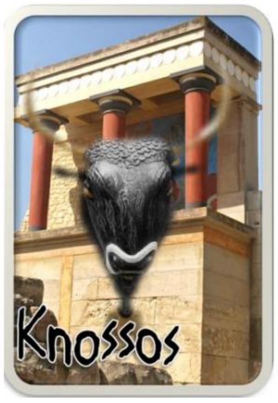


Figure 1. Game board at the beginning

Cards & Trails

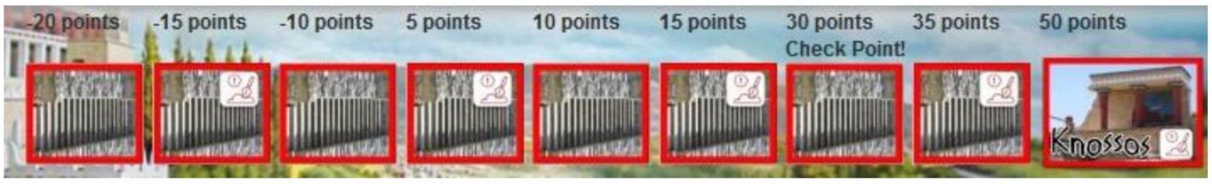
25 Cards correspond to each Palace. Using these cards, each player can move one (and only one) of his pawns only to the corresponding path (in the images we see the cards for the palace of Knossos).

Card	Information
20 Cards with value 1-10 (2 for each value) for every palace 	Each time the player discards such a card, he moves one step to the corresponding palace. However, the cards must be thrown in ascending order (one cannot throw a card with the number 3 if the last card they threw was a card with the number 4) while ties are also allowed (someone can throw a card with the number 3 on top of a card with number 3).

3 Ariadne Myth Cards for each palace	
	Each time the player discards such a card, he moves two steps to the corresponding palace. These cards are thrown separately from cards 1-10 and do not affect the order.
2 Minotaur Cards for each palace	
	Whenever the player discards such a card to the opponent, then the opponent must move 2 steps back to the corresponding palace, unless the opponent's pawn is Theseus, who always counters this attack.

Path

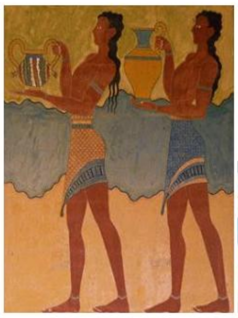
Each Path has 9 steps. The further one reaches, the more points he gets, while if he reaches one of the first three steps, then he has a negative score. At positions 2,4,6,8,9 of each Path there are archaeological finds.



Archeological findings

At positions 2,4,6,8,9 of each path there is an excavation marker where the player can excavate to discover 20 archaeological finds. The 20 finds are placed randomly and are locked in slots. The only restriction is that the 4 rare finds must be randomly located in one of the 5 archeologically existing locations of the corresponding palace foundings.

Archaeological Find	Information
4 Rare Finds **Disc of Phaistos (Worth 35 Points)** 	The Discus of Phaistos is randomly placed in one of the 5 locations on the Path of Phaistos. Its value is 35 points. Once the player discovers the Disc, they add it to their collection.
Ring of Minos (Worth 25 Points) 	The ring of Minos is randomly placed in one of the 5 locations on the path of Knossos. Its value is 25 points. Once the player discovers the ring, they add it to their collection.
Hair Jewelry (Worth 25 Points) 	The Jewel of Malia is randomly placed in one of the 5 locations of the path of Malia. Its value is 25 points. Once the player discovers the jewel, they add it to their collection.
Ryto Zakrou (Worth 25 Points) 	The Ryto of Zakros is randomly placed in one of the 5 positions of the path of Zakros. Its value is 25 points. Once the player discovers the ryto, they add it to the collection of.
Statuettes of the "Snake Goddess" 	There are 10 Statuettes. Each of them is randomly placed on the remaining 16 available slots (the other 4 have been filled by the rare finds). When a player reaches such a location, they add such a statue to their collection. His goal is to collect as many statues as he can.

Wall paintings	
20 Points **20 Points**  15 Points 15 Points 15 Points 20 Points	There are 6 murals that are randomly placed in the remaining available slots. When a player discovers a mural, they can photograph it. But the murals don't move from their position (they don't become a player's property), so if the other player passes by the same position, then they can take a photo of it too. On the left we see the value of photographing each mural.

Actions taken on each player's turn

Each player's turn consists of 2 phases:

~~**1st phase:** Playing Cards~~

The player can:

1) either drop a card of a new castle and place one of its pawns on the corresponding path,

2) either discard a card of an existing palace to move his pawn to the next position on the corresponding path,

3) either throw a Minotaur card at the opponent,

4) either discard a card.

~~**2nd phase:** He must take one card from the card pile~~

Detailed instructions for the alternatives of each phase follow.

1st phase: Playing Cards

~~**The player chooses a card and does one of the following four actions:**~~

1. Start path exploration

The player may begin exploring a path by discarding a card with a value of 1-10 and choosing which pawn to place there (Theseus or an Archaeologist). Each player can place only one of his pawns on each path, without having to use all of his pawns.

Clarifications

ÿ Both players can be on a certain path.

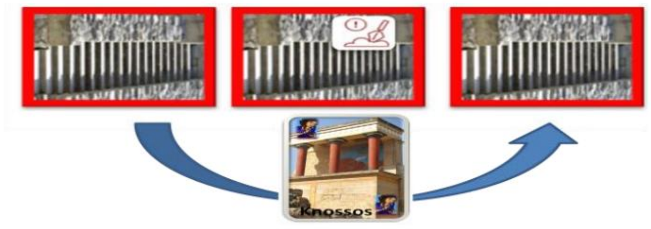
ÿ A path cannot be started with a Mitu Ariadni card.

2. Continue path exploration

The player can continue a path by discarding a card with a value greater than or equal to the previous one. He can also drop a Mith Ariadne card (which does not affect the ascending order). If the player passes by an unexplored dig box, then it is opened (eg the player is informed via a Java Dialog). If it's either a statuette or a rare find, he adds it to his collection (so the other player can't take it). If it's a mural, he just "photographs" it and the mural stays there.

Clarifications

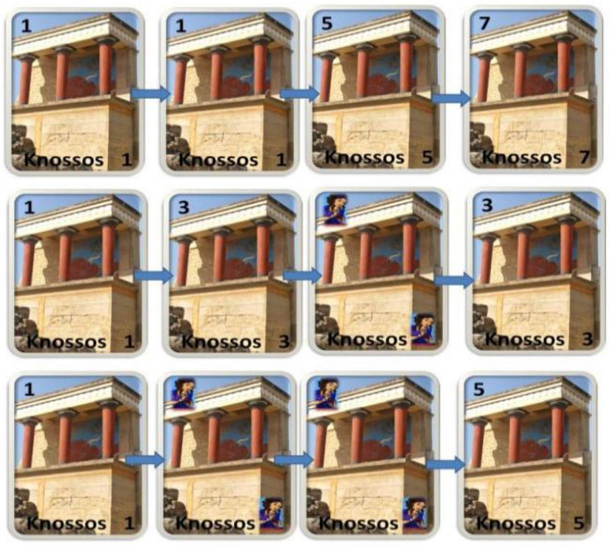
ÿ If the player passes through an excavation box via the Mitu Ariadne card, then they can "excavate" or photograph a mural.



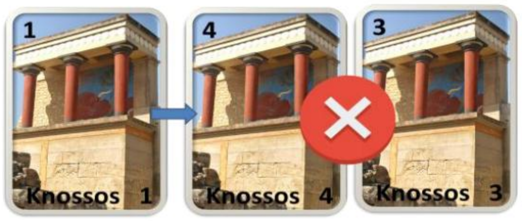
ÿ The Mitu Ariadne card does not affect the ascending order. Example: If a player has discarded the card 3 for Knossos and then discards the Mitus Ariadne (of Knossos) card, to proceed to the next round again he may discard a card with a value (value 3 or higher) or discard a new one card of Mitu Ariadnis.

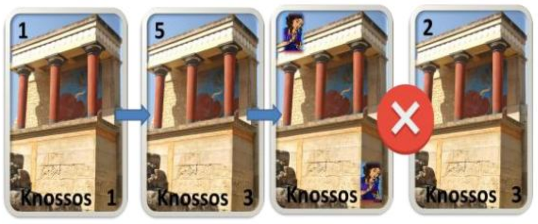
ÿ If the player is in the penultimate position on the path (position 8) and discards a card Mitou Ariadnis, just moves to the last position (position 9).

ÿ Valid Moves



ÿ Invalid Moves





3. Attack an opponent with a Minotaur card

The player can drop one Minotaur card to the opponent for a specific palace. The opponent's player moves back 2 steps unless one of the following cases applies.

1. If the opponent's Theseus is in that particular palace, then he always parries this attack.

2. The card cannot be used if the opponent has reached (or surpassed) it seventh step **(Check Point)** on the given path even with an Archaeologist pawn.

Clarifications

ÿ If the opponent is in the first position of the Path, the attack has no value. If it is in the second position of the Path, it simply returns to the first position.
ÿ Because the player can go back and pass a mural again, he cannot "photograph" it a second time.

4. Rejection of card

The player discards a card (it is placed in the discard pile).

2nd phase: He must take one card from the card pile

The player takes one card from the card pile (each player must always have 8 cards at the end of each round).

Game over

Case (a): when the cards of the Card pile run out.

Case (b): when at least 4 of any pawns (regardless of player) reach the seventh step **(Check Point)** on any path, then the game ends. It is not necessary for the pawns to be on 4 different paths nor do they all belong to the same player. A possible case (certainly not the only one) where the game ends is the following: if both players have reached the seventh step of Knossos, 1 player has reached the seventh step of Phaistos and 1 player has reached the seventh step of Malia, and no one has reached the seventh step of Zakros.

Score- Winner

Each player's score is a sum of the following:

1. The score from the murals he has photographed
2. Of the score of rare finds he has discovered
3. The score from the statuettes he has collected according to the table below:

Number of Statues	0	1	2	3	4	5	6	7	8	9	10
Score	0	-20	-15 5		15	30	45	55	70 90	100	

4. Each archaeologist earns as many points as are written on his point of path located (can be a negative number)

5. Theseus earns twice as many points as those listed on his point
   of path found (even negative points are doubled)

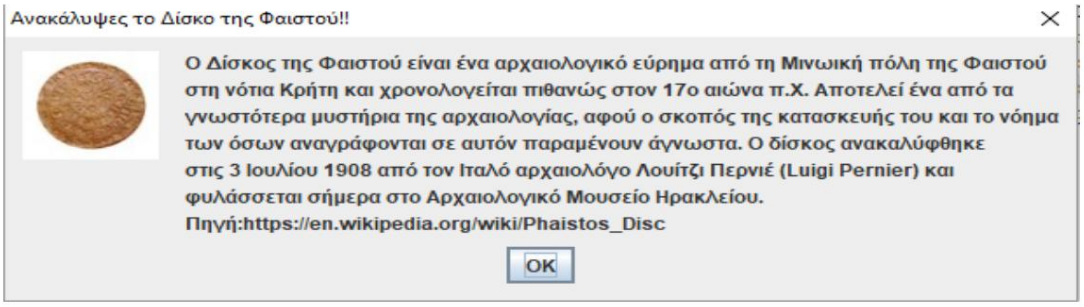
Whichever player has the highest score wins, in case of a tie the following tie criteria apply:

(a). Whoever has earned the most of the 4 rare finds wins
(b). If there is no winner from a) whoever has more murals wins
(c) If there is no winner from (a), (b) whoever has the most figurines wins

(d) If no winner emerges from (a), (b), (c) we have a tie

Historical information

You are given a csv file (available in both *Greek* and *Greek),* which contains information about the 4 Minoan palaces, the 4 rare finds, the 6 wall paintings and the Snake Goddess. Someone should be able to see information about each of them in any way you choose. For example, when he clicks on the palaces he will be able to output the information through a Message Dialog. Regarding the findings, when the player reaches a box with findings, you can again use a message dialog to show information about what each player "discovered", as shown in the image below.


Ανακάλυψες το Δίσκο της Φαιστού!!

Ο Δίσκος της Φαιστού είναι ένα αρχαιολογικό εύρημα από τη Μινωική πόλη της Φαιστού στη νότια Κρήτη και χρονολογείται πιθανώς στον 17ο αιώνα π.Χ. Αποτελεί ένα από τα γνωστότερα μυστήρια της αρχαιολογίας, αφού ο σκοπός της κατασκευής του και το νόημα των όσων αναγράφονται σε αυτόν παραμένουν άγνωστα. Ο δίσκος ανακαλύφθηκε στις 3 Ιουλίου 1908 από τον Ιταλό αρχαιολόγο Λουίτζι Περνιέ (Luigi Pernier) και φυλάσσεται σήμερα στο Αρχαιολογικό Μουσείο Ηρακλείου.
Πηγή:https://en.wikipedia.org/wiki/Phaistos_Disc

Bonus (5%)

Possibility to save the game and later continue it in the next instance of running the application. In particular, the user should be given an option to save and exit the application through the menu. This option will result in persistent storage of the state of the game at that given time. It is up to you, as the application designers, to decide what to store and how. Then, given that such a save action has taken place, your app should be able to again via the run menu offer the option to resume that particular game.

Design/Implementation Hints

During game design (and in general any software or technical project), the aim is to decompose the system into smaller parts, with the aim of assigning clearly defined and defined responsibilities to each part and validating that all parts together achieve the goals of the system. Therefore, design is a process of solving and breaking down the original problem into individual smaller and more easily solvable sub-problems that will satisfy functional requirements and be subject to specific principles of good design.

One such principle is the disconnection of the model (model), which describes the data, their behavior and the set of rules that governs them, from their display (view). A key goal of such a decoupled approach is to minimize the required code interventions that can have future changes to either the model or the display mode/medium. This leads to better code quality, with lower maintenance, extensibility and reuse costs. To make this easier to understand consider the following scenario. Suppose you initially designed an application that for various reasons prints its output to the console. Having followed such a 'loosely' coupling architecture, the transition to a graphical windowed environment is smooth, simply by appropriately modifying (extending) those processes that were responsible for rendering the model to the console.

Following is a suggested design of the application, which you can follow and extend accordingly. This design strategy is not unique, as an object-oriented software system can obviously be structured in many ways. Just think of the number of possible classes, the functions they can contain and the possible relationships between them. Therefore, you are free to propose your own design as long as it follows the principles we have outlined and is properly documented.

MVC (Model-View-Controller) pattern

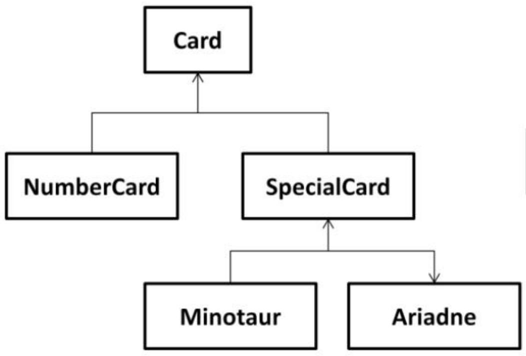
The basic principle in the development of this work should be the MVC (Model-View-Controller) pattern. According to this standard, the development of the graphical interface of the game (View) should be separated from the core of the game that contains all the state information (Model) and from the mechanism for managing and updating the actions of the game with its graphic display (Controller). Watch a video to understand it fully: http://www.newthinktank.com/2013/02/mvc-java-tutorial/.

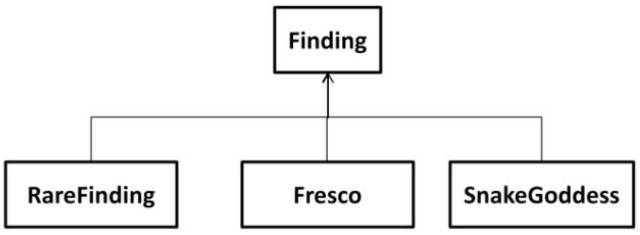
More specifically the Model can be considered to consist of anything related to the game data. In this sense the "Pawns", the "board", the "Players" are part of the game model as they describe the data that the Game Controller is called upon to manage. The Controller is in charge of management of the interaction of the graphical interface with the model. The MVC pattern gives you the possibility to separate the implementation of the above and consequently make your program easier to develop and debug (each component can be developed and tested separately).

Recommended Classes – Model

The model should include all the contents of the game, i.e. there will be classes for the **cards,** positions and **board** of the game, for the **players and** for the **archaeological finds.**

More specifically, a base class is the **Card class.** This class models a game card and defines its attributes (instance variables, e.g. which palace it belongs to) and its useful methods, e.g. matchCard (Card c) where it will check if a card can be played over from another. Cards would be appropriate to be divided into subclasses, such as **NumberCard** and **Special Card ,** as shown below. Specifically NumberCard
will include the cards that have some value (1-10) and the Special Card could have subclasses for Ariadne's Minotaur and Myth cards.

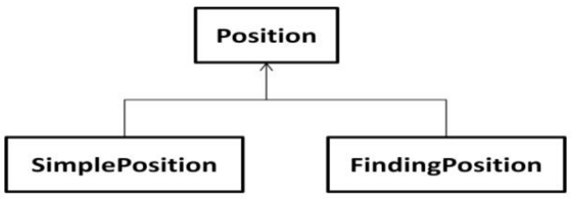




For archaeological finds you can make a **Finding class,** where each of them has a description and an image, while they could be divided into subclasses according to their type, eg **RareFinding, Fresco** and **SnakesGoddess.**

A necessary class of the game is that of the player **(class Player).** As its name suggests, this class simulates the game player. Each player keeps 4 pawns, archaeological finds, cards, has a score, murals, statuettes, etc.

Regarding the **positions** of the game, each position **(class Position)** belongs to a path (palace) and has a specific score. Also, some positions have archaeological findings and some do not, so it would be correct to have a separation with the appropriate subclasses, since these classes should have some additional fields. For example, in the class for archaeological finds there should be a method that returns the find that is in that location and a method that will return if that find is still available (or if a player has already taken it).



Also one could make a class for the path of a palace where each path would belong to a palace and the 9 positions of each path would be contained. Finally, you can also create a class for the board **(class Board)** of the game where all the paths, the stack of cards, etc. will be contained.

Controller

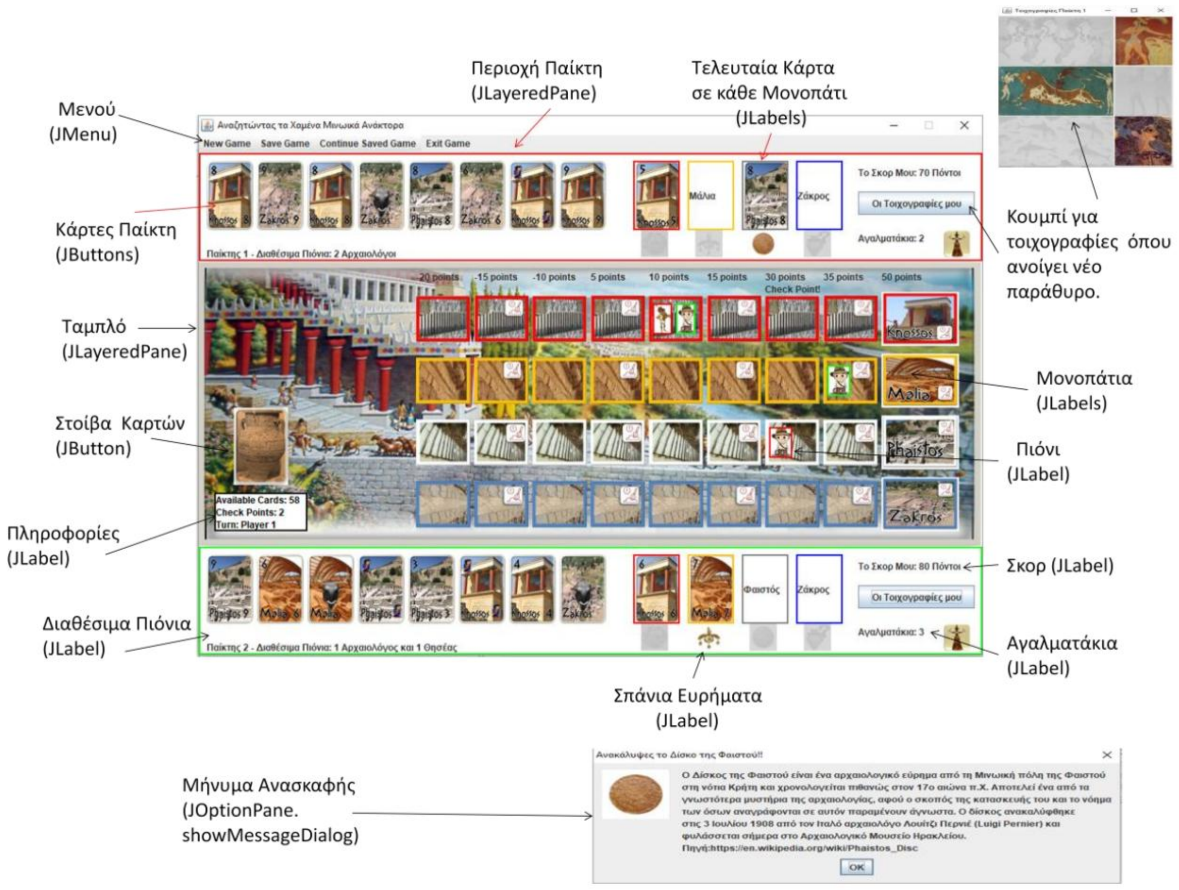
Controller has access to game data and interface. From the interface it receives information about the movements of the players and with this it should update the game model. Accordingly the refresh of the game model should be used by the Controller to refresh the interface (as a result of player movement). This class is the "communication and coordination brain" of the game, as it orchestrates all its possible functions and provides the necessary methods to the outside for the core of the game (the set of rules) to communicate with its graphical representation. Specifically, it checks for its initialization, for its smooth conduct, the order that the players will play, the correctness of the movements, the termination of the game and the selection of the winner.

Think through the flow of the game and include in your design all the necessary features of the class. In particular, it is important to think and design how the different parts of the model and the controller will communicate with each other, for example what will be the flow when the player throws a card, which methods (which will be in different classes) and with which order should be used.

Graphic Illustration

The graphical interface of the game must present the board and enable players to simultaneously play the game on a computer. A design
an example of the implementation of this interface is presented in Figure 2. In this image we have a dashboard, where it has a background and above it we have the paths. Above and below the paths is each player's area, where they have their cards, and information about the cards they have dropped on each path and their archaeological finds. This graphic illustration is indicative. Any variation that does not reduce the functionality of the game is acceptable. For example one could add anything they like that makes the game more enjoyable like their names

players, sounds (e.g. for the cards) etc.



Ο Δίσκος της Φαιστού είναι ένα αρχαιολογικό εύρημα από τη Μινωική πόλη της Φαιστού στη νότια Κρήτη και χρονολογείται πιθανώς στον 17ο αιώνα π.Χ. Αποτελεί ένα από τα γνωστότερα μυστήρια της αρχαιολογίας, αφού ο σκοπός της κατασκευής του και το νόημα των όσων αναγράφονται σε αυτόν παραμένουν άγνωστα. Ο δίσκος ανακαλύφθηκε στις 3 Ιουλίου 1908 από τον Ιταλό αρχαιολόγο Λουίτζι Περνιέ (Luigi Pernier) και φυλάσσεται σήμερα στο Αρχαιολογικό Μουσείο Ηρακλείου.
Πηγή:https://en.wikipedia.org/wiki/Phaistos_Disc

Figure 2

Recommended Classes – View

Finally the View is the graphical interface with which the end users of the game interact. In terms of interface the use of inheritance can facilitate development by creating new types of UI components that extend the functionality of existing ones (for example creating a CardButton type can help you handle positions as JButtons but with the attributes you you want the game cards to have).

You can use the JLayeredPane class, where one can place one component (eg, button, label, text) on top of another and each component has specific coordinates. The game board can be made with either buttons or JLabels. In any case, it is important to always show where the pawns are in the graphical environment. It's important that the cards are a button, where you can right-click to play a card, and left-click to discard a card. The messages when excavating, but also to see the murals that each player has acquired, can be made via Java Dialog, where you can use any type of Java Dialogs that suits you (see

http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html).

Alternatively, one can also use a BorderLayout for the design, to separate the player area and the board, while for the paths one could use a Grid Layout (dimensions 4*9). **There is no restriction on the graphics implementation, basically try to use the implementation that you find most preferable.**

Auxiliary

Below you are given some indicative links which can help you in the implementation:

Official Instructions in English. http://riograndegames.com/getFile.php?id=382

Be careful, several instructions are different from the pronunciation. You always rely on the pronunciation instructions!!!

A tutorial on the MVC (Model View Controller) model in Java

http://www.newthinktank.com/2013/02/mvc-java-tutorial/ (be sure to check it out)

The photos for all the contents of the game (cards, paths, finds, pawns etc) and the csv file (in Greek and Greek) have been uploaded to elearn!

You will **also be given a project from the previous year so that you can get an early sense of what your final implementation will be like.**

Good work!!!