|  |  |
| --- | --- |
| https://user-cdn.uef.edu.vn/newsimg/files/logo%20nhan%20dien%20TH/3.jpg | **TRƯỜNG ĐẠI HỌC KINH TẾ TÀI CHÍNH TP. HCM**  **KHOA CÔNG NGHỆ THÔNG TIN** |

**FINAL PROJECT**

# Course:

# Java Technology

**Project:**

**Cognitive and Memory game**

TP. Hồ Chí Minh – 2022

**TRƯỜNG ĐẠI HỌC KINH TẾ TÀI CHÍNH TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**FINAL PROJECT**

**Course:**

# Java Technology

**Project:**

**Cognitive and Memory game**

TP. Hồ Chí Minh – 2022

**Mục lục**

[Course:  Java Technology 1](#_Toc138259501)

[Java Technology 2](#_Toc138259502)

[CHAPTER 0. INTRODUCTION 1](#_Toc138259503)

[CHAPTER 1. PROJECT SUMMARY 3](#_Toc138259504)

[**1.1.** **Overview** 3](#_Toc138259505)

[**1.2** **Scope of research, scope of application** 4](#_Toc138259506)

[**1.3** **Function, work done** 4](#_Toc138259507)

[CHAPTER 2. THEORETICAL BASIS 5](#_Toc138259508)

[**2.1. Java language** 5](#_Toc138259509)

[**2.1.1. History** 5](#_Toc138259510)

[**2.1.2. Java language features** 5](#_Toc138259511)

[CHAPTER 3. library introduction 7](#_Toc138259512)

[**2.** **ActionListener** 7](#_Toc138259513)

[CHAPTER 4. Programming a Flip Game With Java 15](#_Toc138259514)

[**4.1. The game rules are very simple** 15](#_Toc138259515)

[**4.2. Initialize the data and set the interface.** 15](#_Toc138259516)

[**4.2.1. Image** 15](#_Toc138259517)

[**4.2.2. Matrix - image representation** 16](#_Toc138259518)

[**4.2.3. Add images to the interface.** 19](#_Toc138259519)

[**4.3. Handling operations** 20](#_Toc138259520)

[**4.3.1. Click an image** 20](#_Toc138259521)

[**4.3.2. Time in the game screen** 22](#_Toc138259522)

[**4.3.3. New game and Next level** 23](#_Toc138259523)

[CHAPTER 5: SUMMARY 25](#_Toc138259524)

[MIÊU TẢ CÔNG VIỆC 26](#_Toc138259525)

[TÀI LIỆU THAM KHẢO 27](#_Toc138259526)

NHẬN XÉT – ĐÁNH GIÁ

1. Hình thức trình bày

1. Nội dung

Điểm: Hình thức (20%) Nội dung (80%) Tổng kết:

Ngày Tháng Năm 2022

Giảng viên

Nguyễn Văn Tân, ThS.

# CHAPTER 0. INTRODUCTION



Java is an object-oriented programming language (titled C++) released by Sun Microsystems in the mid-1990s. Programs written in the Java programming language can run on any system where a Java virtual machine (Java Virtual) is installed. Machine). Java is an object-oriented programming language. The strength of the object-oriented programming method is reflected in the ability to model the system based on real objects, the ability to encapsulate and protect data safety, the ability to reuse the source code to save money. save costs and resources, especially the ability to share source code in the professional developer community. These strengths promise to promote the development of an advanced programming environment along with the software assembly industry with available component libraries.

A Java topic must highlight the nature of the subject, be practical and bring practicality and can be implemented in practice. It is for these reasons that we have decided to choose the topic **Flipping game** as the final topic of this course.

The program has basic requirements such as:

Functions of the game: create a new game screen, play again, pause, continue playing, score, announce results, exit the game, play instructions, change the interface.

Graphics requirements: the game must be designed so that it is easy to use, reasonably designed to be easy to manipulate.

Handle in-game player events: tile flips, tile clears, button click events.

During the teaching process, thanks to the dedicated guidance of teacher **Nguyen Van Tan** and the help of the teachers in the department, we have completed the exercise with the topic **Flipping game** programming written in the language of the teacher. Java language.

From there, we saw how applications are created, and helped us master algorithmic methods, object-oriented programming on Java.In the process of making the topic, although I tried to find out, it is still inevitable that errors and functions have not been completed.

That is why we look forward to the input of teachers.

We sincerely thank you!

# CHAPTER 1. PROJECT SUMMARY

This section will briefly describe the game "Flipping game" which is quite popular today, although the name sounds new.

* 1. **Overview**

In our free time or after every stressful working hour, each of us wants to find a game or an activity to relieve stress and regain our spirits. Maybe it's sports activities, or playing entertainment games depending on each person's time budget, and for those who have little time but want a highly entertaining game, the game **Flipping game** is one of the games that meet that need.

**Flipping game** is a game written according to the title of the flipping game - a game that is quite popular today, the name " **Flipping game** " is created by us, based on the title of a famous flipping game on the internet. network. Depending on the images that the game uses, the name for this game is also very different such as: Finding Pets, Purble Place(Window 7), Critter Match ...

A screenshot of a game

Description automatically generated

Figure 1.1: Critter Match game interface.

The main goal of the topic is to help players strengthen their memory, senses and feel comfortable after conquering all the boxes containing the pictures. The theme has normal, difficult and easy levels of play suitable for each player and is equally intellectual when the player has to calculate so that the total number of times opening the same images.

The project details the steps to build the game "**Flipping game**" using Java language and approaches to solving problems in the game as well as the results obtained based on a number of examples. test example while programming.

* 1. **Scope of research, scope of application**

Research scope: self-study.

Scope of application: personal.

## **Function, work done**

|  |  |  |
| --- | --- | --- |
| **STT** | **Function, work** | **Made medicine** |
| 1 | Create a friendly interface, easy to use. | ✓ |
| 2 | Can play 1 person or 2 people. | ✓ |
| 3 | Random images to buttons. | ✓ |
| 4 | Increase the difficulty when passing the screen | ✓ |
| 5 | Calculate the score and save the scores. | ✓ |
| 6 | Play again. | ✓ |

**Table 1.1 : The job function was done.**

# CHAPTER 2. THEORETICAL BASIS

This chapter will present the theoretical bases used, the game design model play.

## **2.1. Java language**

### **2.1.1. History**

In 1990, James Gosling was commissioned by Sun Microsystems to build a programming software for consumer electronics with the goal of being compact and compatible with many different hardware devices. At first this group planned to use C++ but then they found C++ unsuitable because it was too cumbersome and required many hardware devices to support... So they decided to build a new language and name it. Oak (later changed to Java because Oak had the same name).

Oak was first used in the Green Project, in which the development team applied the control system to household electrical appliances such as TVs, VCRs, lights, phones as well as all portable computer devices.

In 1993, the Internet developed, so that Internet information was transferred from text to graphic environment. The Java development team believes that Java, if developed, will be the most perfect language for programming on the Web. So they built small applications embedded in the Web (called Web Applets) and deployed a complete application, a Web browser (called hot java) easily demonstrated the powerful capabilities of Java.

Since 1995, Java is rapidly gaining acceptance as a powerful tool for developing applications for the Internet. Web browsers introduced support for Java one after another, starting with Netscape Communications with Netscape Navigator Web (version 2.0) and then Microsoft Explorer 3.0.

### **2.1.2. Java language features**

* Compact: Java version 1.0 is quite simple because the size of the base compiler and the supporting class is very small – 40Kbytes for each interpreter and for the support class. The size of the standard library and the routing support layer is only about 175Kbytes.
* Object Oriented: Every application written in Java must be built on and through objects.
* Distributed: Java's library has a network layer ( java.net ). It has the ability to access from a remote server, or access objects through the network easily.
* Interpretation: When the source file of java is completely compiled into a .class file, it is
* easily run on many different environments and platforms containing the interpreter.
* Powerful: A Java program must require certainty, it is not easy to be arbitrary in variable declaration, Java does not provide pointer variables like other programming languages.
* High security: Java is not only a pure programming language, but it also provides many levels to control the safety of program execution.
* Virtual machines: When the source code is compiled as bytecode, Java creates virtual machines on each system so that applications can run easily.
* Portability: The compiled Java source code can be run on any platform based on the virtual machine alone. That's why Java has a slogan "write once run anywhere".
* Efficiency: Java is quite slow compared to some other languages ​​like C/C++ because it takes time to compile to file.class and then interpret to run, however, in other environments, Java is relatively efficient. its compact and portable features.
* Multithreading: Java supports multi-threaded programming, parallel programming avoids boring serialization, improving job execution time.

# CHAPTER 3. library introduction

## **ActionListener**

**ActionListener** is an interface in Java, belonging to the javax.swing.event package, used to listen for and handle events related to actions in a user interface. The ActionListener interface specifies a single method named actionPerformed(ActionEvent e) to handle events when an action occurs.

A screen shot of a computer program

Description automatically generated with low confidence

The given code is a constructor for the `GameLatHinh` class. Let's explain each part of this constructor:

1. **this.setTitle**("CodeLearn - Game Lật Hình"): This line of code sets the title of the program's window to "CodeLearn - Game Lật Hình".

2. **level = k:** The variable **`level`** is assigned the value of the **`k`** parameter passed to the constructor. It can be assumed that **`level`** represents the game level.

3. **`cn = init(k, score);`:** This code calls the **`init`** method with the **`k`** and **`score`** parameters to initialize and assign a value to the **`cn`** variable. Specifically, it initializes and assigns some object (possibly a UI component) to **`cn`.**

4. **`timer = new Timer(240, new ActionListener() { ... });`:** This code creates a **`Timer`** object with a time interval of 240ms and an inline-defined **`ActionListener`** object. This **`ActionListener`** is used to handle the event when the time of the **`Timer`** elapses. In the **`actionPerformed`** method, it calls the **`open()`** method and stops the **`Timer`** by calling **`timer.stop()`.**

5. **`timer2 = new Timer(100, new ActionListener() { ... });`:** Similar to the previous code, this code creates a **`Timer`** object with a time interval of 100ms and an inline-defined **`ActionListener`** object. This **`ActionListener`** is used to handle the event when the countdown time of the game increases. In the **`actionPerformed`** method, it increments the value of **`time`,** updates the value of the progress bar **(`progressTime`),** and checks if the time has reached the limit **(`maxTime`).** If it has reached the limit, it stops the **`Timer**` and displays a dialog box with information related to the game result.

A picture containing text, screenshot, menu

Description automatically generated

The provided code is a method called **`open()`** in a game program. Here is a detailed explanation of each part of the **`open()`** method:

3.2 Assigning values to variables `**m`**, `**n`**, `**maxTime**`, and **`time`:**

- **`m**` represents the number of rows on the game board, determined by the value of **`k`.**

- **`n`** represents the number of columns on the game board, determined by the value of **`k**`.

- **`maxTime**` represents the maximum time (in units of 0.1 seconds), determined by the value of **`k`.**

- **`time**` is set to 0 to keep track of the playing time.

3.3. Initializing the interface and components:

- Retrieve the container of the **`JFrame**` using **`getContentPane()`.**

- Create a new **`JPanel**` named `**pn`** and set its layout to `**GridLayout**` with `**m**` rows and `**n**` columns.

- Use nested loops to create the game tiles **(`bt[i][j]**`) and add them to `**pn**`.

- Set the `**ActionCommand`** for each tile to determine its position in the matrix.

- Register an `**ActionListener`** for each tile using `**addActionListener(this)`**.

- Set the background color of each tile to black (`**Color.black**`).

- Randomize the values for each tile in the matrix `**a[i][j]`** and set the tile's icon to hidden (`**getIcon(0**)`).

- Set the initial values of the **`tick[i][j]`** array to **`true`** to mark that all tiles are unexplored.

3.4. Creating and configuring the interface for the score and progress bar:

- Create a new **`JPanel**` named **`pn2**` and set its layout to **`FlowLayout**`.

- Create a **`JButton**` with the score value and set its font and background color.

- Create a **`JLabel**` with the text "Score:" and set its font.

- Add **`score\_lb`** and **`score\_bt**` to **`pn2**`.

- Create a **`JProgressBar**` with a range from 0 to `**maxTime`** and set its initial value to **`maxTime**`. Set the foreground color of the progress bar to orange.

3.5. Creating and configuring the game layout:

- Call the **`createMatrix()`** method to set up the game matrix.

- Call the **`showMatrix()`** method to display the initial game board.

- Add `**pn`** to the container.

- Add the progress bar **(`progressTime**`) to the north position of the container.

- Add `**pn2**` to the south position of the container.

- Set the visibility of the JFrame to true.

- Set the size of the JFrame based on the number of columns **(`n`)** and rows **(`m`)** of the game board, as well as additional spacing.

- Set the default close operation of the JFrame.

- Set the JFrame to be non-resizable.

3.6. Return the container.

In summary, the `**init()`** method sets up the initial state of the Flip Game by creating the game board, initializing the score and progress bar, and configuring the game layout on the JFrame.

A screen shot of a computer code

Description automatically generated with low confidence

The provided code is a method called `**showMatrix()**`. Here is an explanation of what this method does:

1. Iterate through the rows of the game matrix:

- Use a `**fo**r` loop with a loop variable **`i**` to iterate from 0 to `**m-1`** (the number of rows).

2. Iterate through the columns of the game matrix within each row:

- Use a nested `**for**` loop with a loop variable **`j**` to iterate from 0 to `**n-1**` (the number of columns).

3. Print the value of each cell in the game matrix:

- Use `**System.out.printf("%3d", a[i][j])`** to print the value of the cell at position **`(i, j)`** in a formatted manner.

- The **`%3d**` format specifier ensures that each value is printed within a field of width 3 characters, right-aligned.

4. Print a new line after printing each row:

- Use `**System.out.println()`** to print an empty line after printing all the values in a row.

5. Print a horizontal line to separate each iteration of `showMatrix()`:

- Use `**System.out.println("-----------------");`** to print a line of dashes as a visual separator.

6. Print an empty line:

- Use **`System.out.println();`** to print an empty line for readability.

In summary, the **`showMatrix()`** method is used to display the current state of the game matrix **`a[][]`** on the console. It prints the values of each cell in a tabular format, row by row, and adds visual separators between iterations of **`showMatrix()`.**

A picture containing text, font, multimedia software, screenshot

Description automatically generated

The provided code is a private method called `**getIcon(int index)`.** Here is an explanation of what this method does:

1. Declare and initialize variables:

- **`width`** represents the desired width of the icon image.

- **`height**` represents the desired height of the icon image.

2. Load the icon image:

- Use `**getClass().getResource("/Game/icon/icon" + index + ".jpg")`** to retrieve the resource URL of the icon image based on the given `**index**`.

- Create a new **`ImageIcon**` object using the retrieved resource URL.

- Get the `**Image`** object from the **`ImageIcon**` using `**getImage()`.**

3. Scale the image to the desired dimensions:

- Use `**image.getScaledInstance(width, height, image.SCALE\_SMOOTH**)` to obtain a scaled instance of the image with the specified `**width`** and `**height**`. The `**SCALE\_SMOOTH**` argument ensures that the scaling operation is performed smoothly.

4. Create an **`Icon`** object from the scaled image:

- Create a new `**Icon`** object using the scaled image obtained in the previous step.

5. Return the created **`Icon**` object.

In summary, the `**getIcon(int index)`** method is used to retrieve and scale an icon image based on the given `**index`**. It loads the icon image from a specified resource path and returns an `Icon` object that represents the scaled image.

A picture containing text, screenshot, font

Description automatically generated

The provided code includes two methods: `**newGame()`** and `**nextGame()**`. Here is an explanation of what these methods do:

1. `**newGame()`** method:

- Close the current game window by calling `**this.dispose()`.**

- Create a new instance of the **`GameLatHinh**` class with initial level 0 and a score of 100.

- The `**GameLatHinh**` constructor is called with the specified level and score, creating a new game window.

2. **`nextGame()`** method:

- Close the current game window by calling `**this.dispose()`.**

- Create a new instance of the `**GameLatHinh`** class with an incremented level and an updated score.

- The **`level`** is increased by 1 to move to the next level.

- The score is calculated by adding the remaining time (computed as the difference between `**maxTime**` and `**time**`) divided by 50 to the current score obtained from `**score\_bt.getText()`.**

- The `**GameLatHinh**` constructor is called with the updated level and score, creating a new game window.

In summary, both `**newGame()`** and `**nextGame()**` methods are responsible for creating a new instance of the `**GameLatHinh`** class with different initial parameters. They dispose of the current game window and create a fresh game window to start a new game or proceed to the next level, respectively.

A screen shot of a computer

Description automatically generated with medium confidence

The provided code includes a method `**showDialogNewGame(String message, String title**)` and an **`@Override**` annotation. Here's an explanation of what the code does:

1. `**showDialogNewGame(String message, String title)`** method:

- Takes two parameters: **`message`** (the content of the dialog) and **`title**` (the title of the dialog).

- Uses the **`JOptionPane.showOptionDialog()`** method to display a dialog box with options.

- The dialog box includes the specified **`message**` as its content and **`title**` as its title.

- The options presented in the dialog box are "Yes" and "No".

- The `**JOptionPane.YES\_NO\_OPTION**` argument specifies that the dialog box should have Yes/No options.

- The `**JOptionPane.QUESTION\_MESSAGE**` argument specifies that the dialog box should display a question mark icon.

- The `**null**` arguments represent the custom options, default option, and icon (which are not used in this case).

- The return value of `**JOptionPane.showOptionDialog**()` is assigned to the variable **`select**`.

2. Check the user's selection:

- If the user selects "Yes" (option 0), the `**newGame()`** method is called to start a new game.

- If the user selects "No" (option 1) or closes the dialog box, the program exits by calling `**System.exit(0)`.**

3. **`@Override**` annotation:

- This annotation indicates that the method is intended to override a method in the superclass or implement an interface method. The overridden method might be inherited from a parent class or implemented from an interface.

In summary, the `**showDialogNewGame()**` method displays a dialog box with a specified message and title, providing the user with the option to start a new game or exit the program based on their selection.

# CHAPTER 4. Programming a Flip Game With Java

Flipping game is a game familiar to most people. To make the game more interesting, try with pictures of your crush, maybe you can practice your coding skills and crush your crush with this simple game.

## **4.1. The game rules are very simple**

Each screen will have 2 \* k (an even number) of hidden pictures, the pictures do not change position in each level. You can open two pictures consecutively (you will see the two pictures) if the two images are the same they will be deleted, otherwise the two pictures will continue to be hidden.

When you open all the same pairs of pictures, the game screen will end, to make the game more interesting, there will be time pressure in each level, requiring you to be very calm to remember the position of the player. pictures.

## **4.2. Initialize the data and set the interface.**

### **4.2.1. Image**

To make this game you need a number of different images.

Your names should be in the order shown below, for example icon1.jpg, icon2.jpg, icon2.jpg, ...

A screenshot of a computer screen

Description automatically generated with low confidence**Figure 4.2.1: Images use for User Interface Flipping game**

### **4.2.2. Matrix - image representation**

We can see that each number represents an image, two images are similar if the values in their respective matrices are equal.

If in that level the pictures are arranged in m rows and n columns. We will use matrix a to represent those images.



We see that, in a matrix, the numbers always appear in the matrix exactly twice, so how to set up such a matrix, especially since it must be randomly arranged.

For example, if you have the most image images, you need to create a matrix of size m \* n for the game, the first thing you need to do is select (m \* n) / 2 images in your image images. , then just double its count.

A picture containing text, screenshot, software, display

Description automatically generated

After executing the above code, then the first m \* n / 2 elements of sequence b is how the discriminant index of the images (no two numbers are the same)

We will double the m \* n / 2 of the first number of b.



Finally, randomly sort the sequence b from 0 to m \* n - 1 and put it in matrix a. Full program for creating matrix a:

A picture containing text, screenshot, design

Description automatically generated  
.

### **4.2.3. Add images to the interface.**

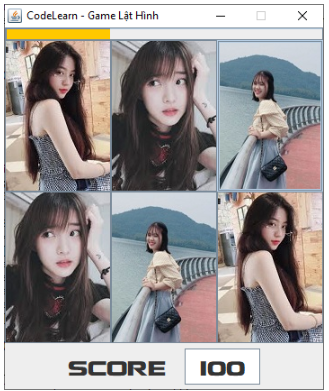
A screenshot of a calendar

Description automatically generated with medium confidenceThe main interface will include Jbuttons in the JFrame class as follows:

To put an image in, we just need to use the setIcon() function. However, each image will receive a different image, so we need to add the getIcon() function as follows:

A screen shot of a computer code

Description automatically generated with low confidence

To put the image corresponding to the button (i, j) we will use the command bt[i][j].setIcon(getIcon(a[i][j])), Then the interface will be:

## **4.3. Handling operations**

### **4.3.1. Click an image**

When starting the game, we will hide the images, for example:

A screenshot of a game with Ice hockey rink in the background

Description automatically generated with medium confidence

We noticed that, when we click on the image for the first time, it immediately opens that image, when we open the image for the second time, we need to check if they are the same or not, if they are the same, delete those two images, and If not, then put those two images in the hidden state.

A screen shot of a computer program

Description automatically generated with low confidence

Use the Timer class to check the image, and create a delay effect when checking the image.

A screen shot of a computer

Description automatically generated with low confidence

### A picture containing text, screenshot, line, design Description automatically generated**4.3.2. Time in the game screen**

To create the time bar as shown above we use JProgressBar in JFrame

A picture containing text, screenshot, font

Description automatically generated

Incorporating Timer (will run when the button is clicked):

A picture containing text, screenshot, software, font

Description automatically generated



The function displays the message:

A screen shot of a computer

Description automatically generated with medium confidence

### **4.3.3. New game and Next level**

Hàm **newGame():**

A picture containing text, screenshot, font

Description automatically generated

The nextGame() function is similar, but note that you must pass additional parameters to save the number of points obtained in the previous screen.

A picture containing text, font, screenshot

Description automatically generated

When the level of the game is observed, the level will be increased, the difficulty will be higher.

A screenshot of a computer

Description automatically generated

When declaring the variable to store score:



# CHAPTER 5: SUMMARY

Above is a guide to creating a flipping game in Java, I have omitted unnecessary places while making the game, if you do not understand any part, you can comment to let me know.

The report on the flip puzzle game presents the design and development process of a flip puzzle game. The game aims to provide players with an engaging and challenging puzzle-solving experience by finding pairs of flipped images on a board.

The report begins with an introduction to the flip puzzle game, highlighting its objectives and significance in developing players' logical thinking and observation skills. It then proceeds to discuss the user interface design, including elements such as the puzzle board, images, and control buttons.

Next, the report delves into the programming process, providing information on the data structure and algorithms used to create the game. An important aspect is the event handling when players select two images for comparison. The report also mentions the technologies employed during the development, such as the programming language and framework used.

Furthermore, the report evaluates the performance of the game, including testing and assessing its features, reliability, and player interaction capabilities. It also highlights the strengths and weaknesses of the game, along with potential areas for improvement in future versions.

Lastly, the report concludes by summarizing the development process and addressing challenges encountered during the design and programming of the flip puzzle game. It emphasizes the significance of the game in enhancing players' logical thinking and observation skills and proposes potential directions for future game development.

In summary, the flip puzzle game report presents the design, development, and evaluation of a flip puzzle game, aiming to improve players' logical thinking and observation skills. It provides insights into the user interface, data structure, algorithms, and technologies used. The report evaluates the game's performance and suggests potential areas for future development.

# JOB DESCRIPTION

|  |  |  |
| --- | --- | --- |
| **STT** |  | **Word** |
| 1 |  | * Search for references * Testing system * Write a general report. * Code support |
| 2 |  | * Knowledge synthesis * Complete the code * PP Presentations |
| 3 |  | * Image search * Code support * Interface design * debug |

# TÀI LIỆU THAM KHẢO