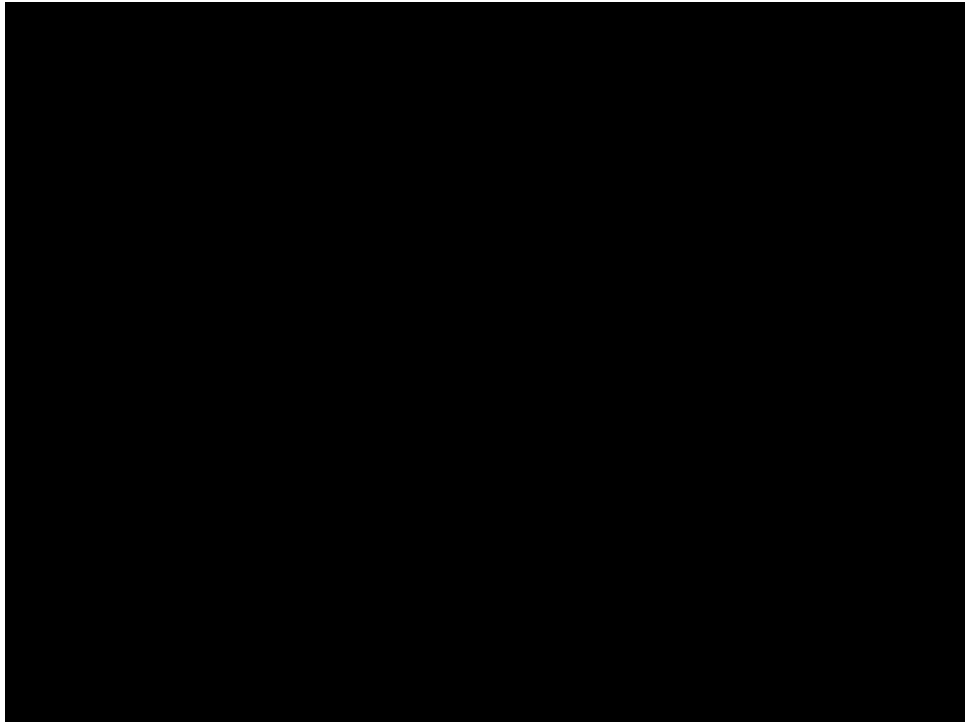
2023-03-08

Unit 9 Project

Rogue

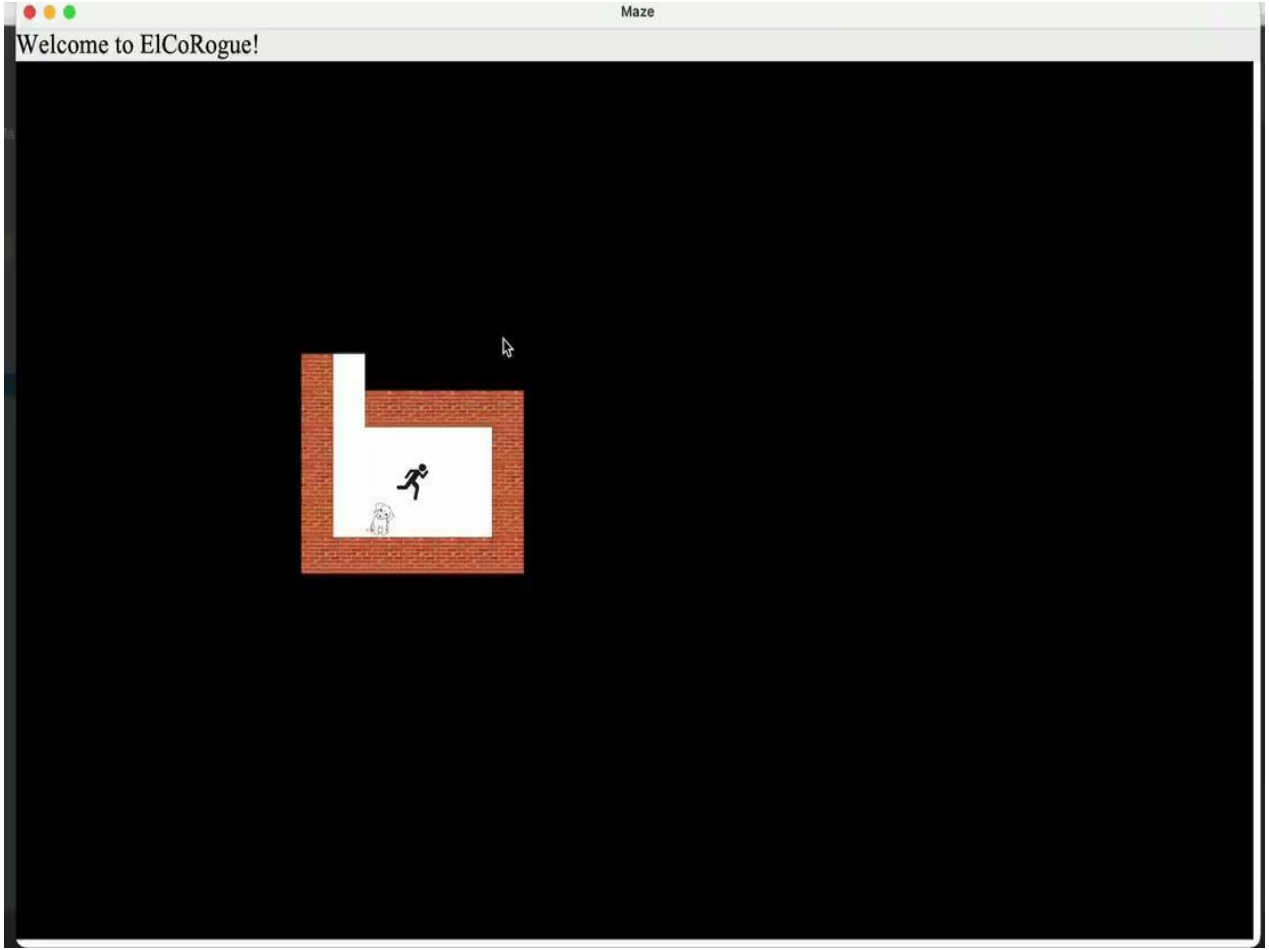
Rogue is a classic text-mode video game from the 80s. It spawned many similar games which are referred to as "Rogue-like." It ran on DOS systems (IBM PC), but also on Unix systems.



Maze Game

The Unit 9 Project is a Rogue-like maze game.

Out of the box, this is a working maze game. But you will write new subclasses to enhance it, and add your own personal stamp to it.



WIMP (computing)

10 languages

Article Talk Read Edit View history

From Wikipedia, the free encyclopedia

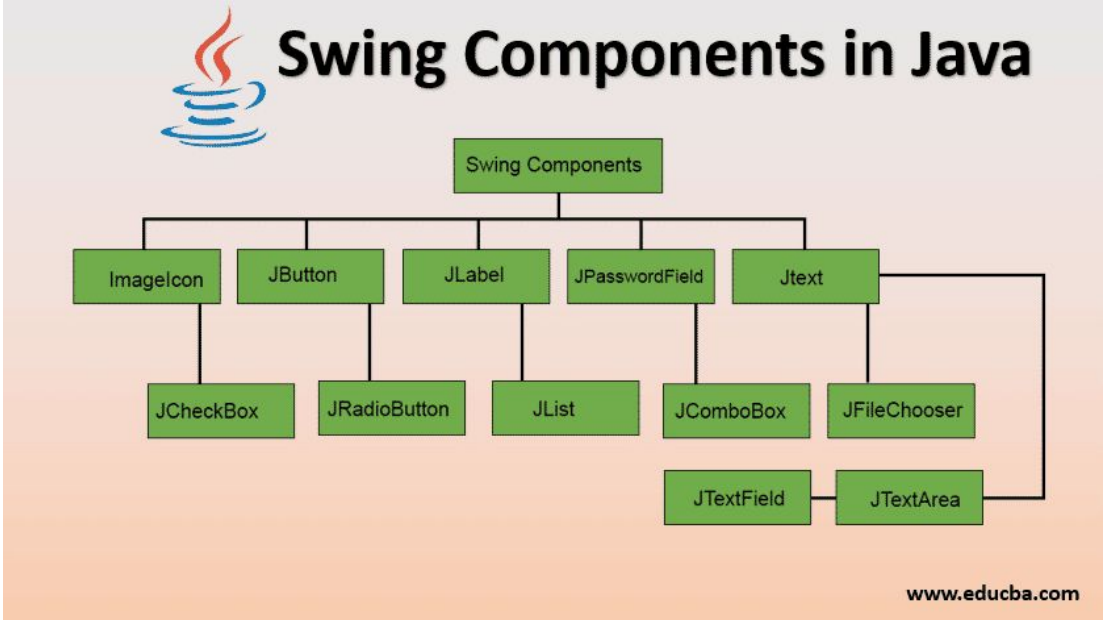
Not to be confused with WIMP (software bundle).

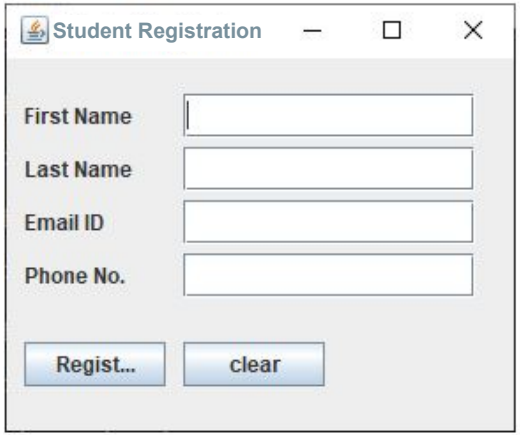
In human–computer interaction, **WIMP** stands for "windows, icons, menus, pointer",[1][2][3] denoting a style of interaction using these elements of the user interface. Other expansions are sometimes used, such as substituting "mouse" and "mice" for menus, or "pull-down menu" and "pointing" for pointer.[4][5][6]

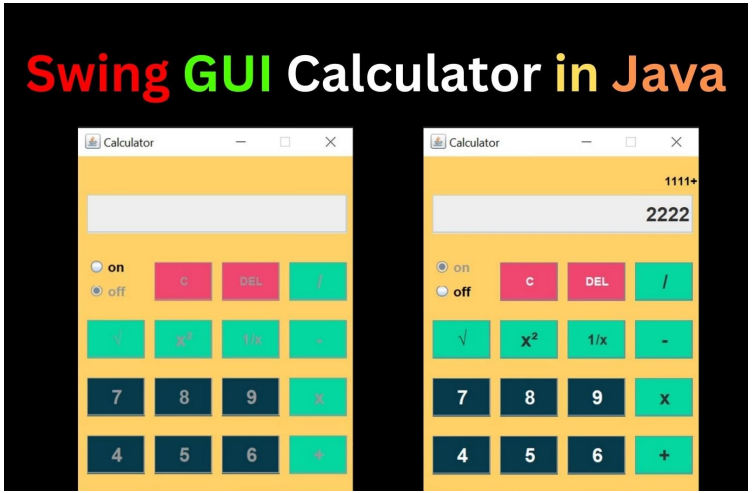
A word processing program that uses a WIMP paradigm, providing mouse-operated toolbars and menus to access its functions

Though the acronym has fallen into disuse, it has often been likened to the term *graphical user interface (GUI)*. Any interface that uses graphics can be called a GUI, and WIMP systems derive from such systems. However, while all WIMP systems use graphics as a key element (the icon and pointer elements), and therefore are GUIs, the reverse is not true. Some GUIs are not based in windows, icons, menus, and pointers. For example, most mobile phones represent actions as icons and menus, but often do not rely on a conventional pointer or containerized windows to host program interactions.[*citation needed*]

WIMP interaction was developed at Xerox PARC (see Xerox Alto, developed in 1973) and popularized with Apple's introduction of the Macintosh in 1984, which added the concepts of the "menu bar" and extended window management.[7]


Swing Components in Java
Swing Components
ImageIcon
JButton
JLabel
JPasswordField
Jtext
JCheckBox
JRadioButton
JList
JComboBox
JFileChooser
JTextField
JTextArea
www.educba.com


Student Registration
First Name
Last Name
Email ID
Phone No.
Regist...
clear


Swing GUI Calculator in Java
Calculator
on
off
C
DEL
/
√
x²
1/x
-
7
8
9
x
4
5
6
+
Calculator
1111+
2222
on
off
C
DEL
/
√
x²
1/x
-
7
8
9
x
4
5
6
+

In the beginning, there was the command line...

Console / Command Line (CLI) Applications

Read from Standard Input (usually keyboard)

Write to Standard Output (usually a console window)

May have their own command loop

May be invoked from the OS command line

Pretty much what interacting with a computer was like until, say, the arrival of the original Mac in 1984

Still used every day by developers, but billions of people have probably never seen one

WIMP / GUI Applications

Often launched using an icon from the Windows/Mac/Linux OS desktop, which itself is a GUI application

Initialization creates a hierarchy of UI components (e.g., JFrame, JLabel, JPanel)

Main loop is an **event loop**, usually part of the GUI framework

Interaction with the user is handled through **event objects**: mouse clicks, key presses, text field changed, repeating timer fired

In Java Swing, classes register themselves as **listeners** for events.

The way most desktop and mobile apps are built


hello.

Rubric

#1 Make your own maze!

Edit the file maze.txt. You can make your own yourself, or you can use a maze generator such as https://www.dcode.fr/maze-generator

(If you use that maze generator, select Single Character display, not Square, for best results, and make sure you specify # as the wall character.)

Any size maze is supported, but don't make it too big because I have to play it!

15x15 or 20x20 is probably a decent size, although you can go smaller too.

Once you've generated or drawn a maze.txt, make sure you place the player, by putting a capital P somewhere. To place the little dog, put a lowercase d.

When editing maze.txt, you may find it helpful to press the INSERT key on the keyboard to enter Overwrite mode. (10 points)

Maze Generator (Perfect) - Onli

dcode.fr/maze-generator

Search for a tool

Search a tool on dCode by keywords:

e.g. type 'sudoku'

Browse the full dCode tools' list

Results

Maze Generator - dCode

Tag(s) : Fun/Miscellaneous

Maze Generator

Fun/Miscellaneous › Maze Generator

Maze Generation

Width (number of corridors) 15

Height (number of corridors) 15

Wall Design

- Black block '' (by default)
- Use this character for walls: #

Path Design

- Empty space ' ' (by default)
- Use this character for paths .

Display

- Square (adapted path width)
- Single chararacter (more rectangular)

GENERATE

See also: Game of Life — Random Selection

Maze Generator (Perfect) - Onli

dcode.fr/maze-generator

Maze Generator

Fun/Miscellaneous › Maze Generator

Maze Generation

Width (number of corridors) 15

Height (number of corridors) 15

Summary

Maze Generation

What is a perfect maze? (Definition)

Feedback

Search for a tool

Search a tool on dCode by keywords:

e.g. type 'sudoku'

Browse the full dCode to

Results

Maze Generator - dCode

Tag(s) : Fun/Miscellaneous

Share

dCode and more

dCode is free and its tools are a valu
games, maths, geocaching, puzzles a
solve every day!
A suggestion ? a feedback ? a bug ? ar
dCode!

15x15

```
  ###########################
    #   #       #   #   # #   #
# ### ### ### ##### # ### # # #
#       # # #     #     #   # #
##### # ### # # ####### ### ###
#   # #       #   #   #   # # #
### ####### ### # # ### ### # #
#         # #   #     #   #   #
# # # ### ########## # ### ###
# # #   #   # # # #     # #   #
# ##### ##### # # ##### # ### #
#   #       # #     # #       #
# ####### ### # ##### ##### # #
#   #       # #   #   # #   # #
# ### # # ### # ##### # ### # #
# # # # # # #     #   # # # # #
### # ##### ##### # ### # ### #
# # #   # #       #           #
# # # # # # # # # # ##### # #
#   # #     # #   # # #     # #
### ##### ### ####### ### #####
#   #     #   #   #     # #   #
# # # # ######## # ######## #
# #   #   #   # #   #     #   #
##### ### # ### # # # ####### #
#     # # # # # # # #     #   #
# # ### ### # # # # ### ### ###
# #   #   # # #   #           #
### ##### # # ##### ######### #
#     #             #
###########################
```

#2 "Re-skin" the maze graphics!

Replace the original images wall.jpg, player.png, etc. with images of your own choice. You can find images on the Internet, such as using the Image search option on Google.

If you like drawing, you could use an online drawing tool such as https://jspaint.app, or even draw with pen and paper and take a picture with a phone.

You will probably need to save the image file to the local computer and then use the Upload File button in replit.

Many image formats will work, but .jpg/.jpeg and .png are definitely supported.

IMPORTANT: Your images should be roughly square dimensions, and not be super high resolution. If the images are too high-resolution, the game will run slow. You can use a website like https://simpleimageresizer.com to reduce the size of your images. (10 points)

#3 Support alternative movement keys

Gamers often prefer using W (up), A (left), S (down), D (right) keys to the arrow keys for movement.

Look at the keyPressed method in Game.java and enhance it to support WASD as well as the arrow keys. The key codes for WASD start with VK_ for Virtual Key, e.g. VK_W, VK_A, VK_S, VK_D.

Another popular set of movement keys is H (left), J (down), K (up), and L (right)... you could choose to support those as well.

(10 points)

#4 Add a monster

Add a new MazeObject subclass which is a monster/creature of some kind that will chase the player.

Register it with the MazeObjectFactory. You will need to supply an image file, just like you did in Step 2.

Upload your image file to replit and override the getImagePath method in your subclass to return the filename of the image.

Add some instances of your creature subclass to maze.txt in different parts of your maze.

You can make multiple classes of creature if you want, but only one is required. (10 points)

MazeObject.java - methods to override in subclasses

```java
// Subclasses must override getImagePath to return the path of the image file to use.
public abstract String getImagePath();

// Subclasses must override getName to return a descriptive name.
public abstract String getName();

// Subclasses may override this to indicate whether light cannot pass through.
// (Wall, for instance, overrides this to return true.)
public boolean isOpaque() { return false; }

// Subclasses may override this to provide per-tick behavior, such as movement.
public void tick() {}

// Subclasses may override this to provide interactive behavior. The status to be displayed
// should be returned, or null if none.
public String interact() { return null; }
```

#5 Animate your monsters

Make the creatures move when the game invokes the "tick" method. look at the logic in Dog.java for inspiration. You will need to override the tick method in your monster subclass.

(10 points)

#6 Add an Item

The Item abstract class can be subclassed to put items in the game.

The way items behave in this game is that you pick them up by running over them.

The only item that comes out of the box is PieceOfTrash, which is trash spread around the maze which you can pick up as an (uncredited) good deed.

Add your own Item subclass which has more in-game significance, like a key to unlock the door out of the maze. You'll need to find another image file to represent the item, and upload it into the replit. Place your item(s) around the maze.

(10 points)

#7 Make a losing scenario

The game needs a way for the player to lose!

If the player bumps into one of the creatures or is bumped by it, the player should lose and the game should end.

The game can be ended by invoking the gameOver method of the Game object. (20 points)

(You can add a more elaborate combat scenario if you want, like a concept of player health and monster health and the ability to hit and be hit multiple times.)

#8 Make a winning scenario

The game needs a better way to win! Currently, the player wins the game by finding the exit of the maze and walking out.

Make it a little harder to win the game... but you can decide how you want the game to be won.

One idea is spread some items around the maze, and the user must collect all of them to unlock the way out.

Note how the Player class overrides the moveTo method and ends the game if the player moves off the map.

Change that to block them unless your winning criteria have been met.

(20 points)