Labs:

Connect Four
Pick Up Trash
Wumpus World (8 Days)
Rodent’s Revenge

Connect Four

In this lab you will be creating a graphical version of the game Connect Four. This project will use the connect four and player classes from your last version.

The game will start with red going first. The game will end when one player has four in a row or when there is no winner and all the spots on the board have been taken.

Before your window loads the program ask the user if they want to play against another player or the computer. The computer will make random valid moves.

You will decide how large your graphics window will be and if you want to use a frame or frame/panel.

When the game is over you will need to display the result and tell the user they can right click to start a new game.

Coding

Create the below files to build your program. If you would like to build a program with only a frame then combine the requirements of the frame and panel.

Main:

	Description
Methods	
Main	• Gathers the number of players • Creates a ConnectFourFrame

ConnectFourFrame (extends JFrame):

	Description
Methods	
ConnectFourFrame(int players)	• Creates/adds a ConnectFourPanel • Adjusts it size to be big enough for the panel and the frames decorations

ConnectFourPanel (extends JPanel & implements MouseListener):

	Description
Constructors	
ConnectFourPanel(int players)	• Sets the size to the dimensions you picked • Sets up the buffer • Sets the mode • Call Reset
Public Static Final Attributes	
int ONE_PLAYER	Stores 1 and is used for mode.
int TWO_PLAYER	Stores 2 and is used for mode.
Attributes	
int turn	• Stores which players turn it is RED / BLACK Note: RED / BLACK come from ConnectFourGame
int mode	Stores if it is a one player or two player game.
ConnectFourGame game	Stores the game
BufferedImage buffer	Stores the off screen image
Methods	
void reset()	Replaces game with a new ConnectFourGame Object
void paint(Graphics g)	• Paints the game to the screen • Paints the correct result when the game is over
void mousePressed(MouseEvent e)	• Code to handle playing (left Button) • Code to handle reset (Right Button)
void mouseReleased(MouseEvent e)	Not used
void mouseClicked(MouseEvent e)	Not used
void mouseEntered(MouseEvent e)	Not used
void mouseExited(MouseEvent e)	Not used

Extra Credit:

Make the computer player always win if there is a win available and always block if it cannot win, but red will on the next turn.

Other Grades:

- ConnectFourFrame

2 Minor Grades:

- ConnectFourPanel

Major:

- Entire Project (See Rubric)

Rubric (1 major grade)

Points	Task
20	Display
10	Both players can move
20	Invalid moves are not accepted
20	Correct end game results
20	The computer moves properly
10	When the game is over, a right click will make a new game
15	Extra Credit

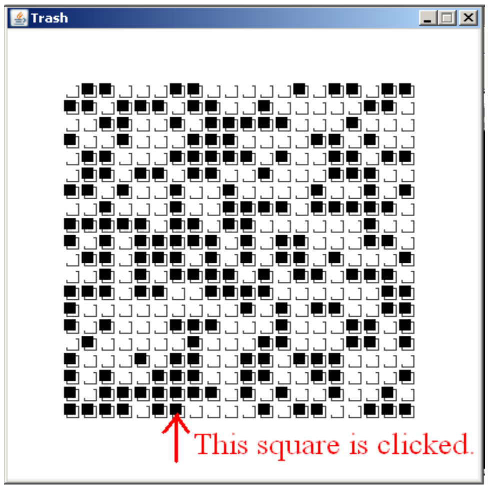
Pick Up Trash

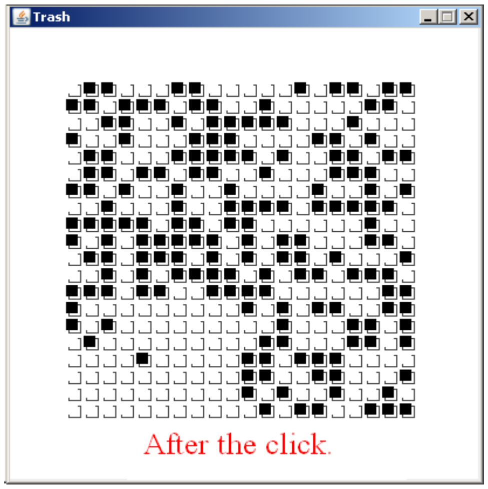
Modify the given code to produce a trash collecting program.

The program will create a 20x20 grid of trash cans that are randomly filled with trash. When the user clicks on a trash bag the program will remove the trash in that cell and all the trash touching the picked up trash to the north, east, south and west.

When neighboring trash is collected in this way its neighboring trash is collected and so on.

Example:





Drawing the Grid of trash cans:

- Each trashcan is represented by a dark gray 10x10 non-filled rectangle.
- Empty trashcans are created by drawing a white filled rectangle that is 10x10 and two pixels and over two pixels to the left.
- Filled trashcans are created by drawing a black filled rectangle that is 10x10 and two pixels and over two pixels to the left.

Rubric (2 minor grades)

Points	Task
20	Panel Constructor
30	Paint Method
10	Mouse Pressed Method
40	Pick Up Trash Method

Wumpus World

In this lab you will create a graphical version of the game Wumpus World.

How the game works:

The player must venture into a dark cave to find treasure. He takes with him a bow and a single arrow.

In the cave there are pits that will kill the player if he/she were to fall into them. Although the player will not be able to see the pits, due to the cave being dark, he will feel a breeze when he is near a pit.

In addition to the pits, there is a wumpus in the cave. A wumpus is a smelly monster that will eat the player when he/she stumbles into it. The player will know he/she is near a wumpus due to its stench.

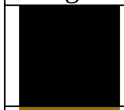
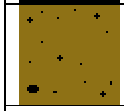
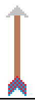
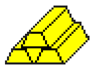
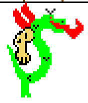
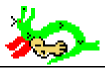
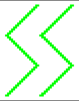
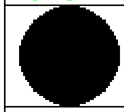
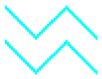
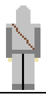
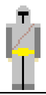
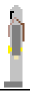
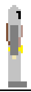
The player may only fire his arrow one time. If the arrow hits the wumpus it will die and the player will hear a scream.

The player will only be able to see squares he/she is on or has been to in the past. To win the game the player must navigate the cave, retrieve the treasure and then climb up the ladder.

Controls:

Key	Action
‘w’	Moves the player up one square
‘s’	Moves the player down one square
‘a’	Moves the player left one square
‘d’	Moves the player right one square
‘i’	Shoots upward (only works if you have an arrow)
‘k’	Shoots downward (only works if you have an arrow)
‘j’	Shoots left (only works if you have an arrow)
‘l’	Shoots right (only works if you have an arrow)
‘c’	Climbs the ladder (only works if you have the gold)
‘p’	Picks up the gold (only when on the square with the gold)
‘n’	Creates a new game (only works after you win or die)
“*”	Toggles on/off cheat mode (When on there will be no fog of war)

Images Used:

Image	Description
	Cell with fog of war
	The cave floor
	Arrow
	Treasure
	Ladder
	Wumpus
	Dead wumpus
	Stench
	Pit
	Breeze
	Player facing up
	Player facing down
	Player facing left
	Player facing right

Your game will consist of a 10x10 map (500x500), with an additional area below the map. The area below the map will be used to show the player's inventory (arrow & gold) and messages. All of this information can be shown with text or graphics.

Needed messages:

- You feel a breeze (when near a pit)
- You smell a stench (when near the Wumpus, near dead Wumpus or on a dead Wumpus)
- You see a glimmer (when on the gold)
-You bump into a ladder (when on the ladder)
-You fell down a pit to your death (when on a pit)

-You are eaten by the Wumpus (When on the Wumpus)
- You hear a scream (On the turn you kill the Wumpus)

Coding:

Main:

	Description
Methods	
main	Creates a Wumpus Frame

WumpusFrame (extends JFrame):

	Description
Constructors	
WumpusFrame()	• Creates/adds a WumpusPanel • Adjusts it size to be big enough for the panel and the frames decorations

WumpusPanel (extends JPanel & implements KeyListener)

	Description
Constructors	
WumpusPanel()	• Sets its size • Creates the buffer • Loads the images • Calls reset
Public Static Final Attributes	
int PLAYING	Stores 0 and is used for status.
int DEAD	Stores 1 and is used for status.
int WON	Stores 2 and is used for status.
Attributes	
int status	Stores the status of the game
WumpusPlayer player	Stores the player
WumpusMap map	Stores the map and its data
The following are BufferedImages: arrow fog gold ladder pit breeze wumpus deadWumpus stench playerUp playerDown playerLeft playerRight	Stores images for the game
BufferedImage buffer	Stores the offscreen image
Methods	
void reset()	• Sets status to PLAYING • Create the map • Places the player at the position of the ladder
void paint(Graphics g)	Paints the game world to the screen, with the appropriate messages.

void keyPressed(KeyEvent e)	Not used
void keyReleased(KeyEvent e)	Not used
void keyTyped(KeyEvent e)	Handles all the players inputs
void addNotify()	• Calls super.addNotify() • Calls requestFocus()

WumpusMap

	Description
Constructors	
WumpusMap	Calls createMap
Public Static Final Attributes	
int NUM_ROWS	Stores the number of rows and it will hold 10.
int NUM_COLUMNS	Stores the number of columns and it will hold 10.
int NUM_PITS	Stores the number of pits and it will hold 10.
Attributes	
WumpusSquare[][] grid	Stores a 10x10 grid of WumpusSquares
int ladderC	Stores the column of the ladder
int ladderR	Stores the row of the ladder
Methods	
void createMap()	Creates the 10x10 grid Randomly places 10 pits and their breezes (cannot be on the gold, wumps or ladder) Randomly places the gold (cannot be on a pit or ladder) Randomly places the wumpus and its stenches (cannot be on a pit or ladder) Randomly places the ladder (cannot be on the gold, wumps or a pit) Sets ladderX and ladderY
int getLadderCol()	Returns the ladder's column
int getLadderRow()	Returns the ladder's row
WumpusSquare getSquare(int row, int col)	Returns the square at the given row and column or null if the location is invalid
String toString()	Returns a string containing a text representation of the map. Example: P****P**** **P***L*** ****P***** *******P** ****P***** *P****P*** *G*P****** ***W****** *****P****

WumpusSquare

	Description
Constructors	
WumpusSquare()	Sets all the attributes to false
Attributes	
The Following are booleans: gold ladder pit breeze wumpus deadWumpus stench	Stores if the square has the given attribute
boolean: visited	Stores if a square has been visited by the player or not
Methods	
Accessors for all the attributes	Returns the value of an attribute
Mutators for all the attributes	Changes the value of an attribute
String toString()	Returns: '*' = for empty 'W' = wumpus 'D' = dead wumpus 'L' = ladder 'P' = pit 'G' = gold

WumpusPlayer

	Description
Constructors	
WumpusPlayer()	Sets the players direction to NORTH, gold to false and arrow to true.
Public Static Final Attributes	
int NORTH	Stores 0 and is used for direction.
int EAST	Stores 1 and is used for direction.
int SOUTH	Stores 2 and is used for direction.
int WEST	Stores 3 and is used for direction.
Attributes	
int direction	Stores the direction of the player
boolean arrow	True when the player has an arrow
boolean gold	True when the player has the gold
int colPosition	The players col position in the grid
int rowPosition	The players row position in the grid
Methods	
Accessors for all the attributes	Returns the value of an attribute
Mutators for all the attributes	Changes the value of an attribute

Suggestions for the file build order:

- WumpusPlayer
- WumpusSquare
- WumpusMap
- WumpusPanel
- WumpusFrame
- Main

NOTE: You may add other variables and methods as needed!!!

Extra Credit

Make your program always generate a map with a solution.

Other Grades:

- WumpusPlayer
- WumpusSquare

Minor Grades:

- WumpusMap
- WumpusPanel
- WumpusFrame

Major Grades:

- Project

Rubric (1 major grade)

Points	Task
20	Creates / Displays the map
10	Player can move properly
10	Fog of war works properly
10	Player dies on a pit or wumpus and only cheat /new game works
10	Arrows work properly
10	Can Pick up the gold
10	Player can win only by climbing the ladder while he/she has the gold. Only cheat/new game works
10	Inventory / message display works properly
10	Cheat mode can be toggled on/off
20	Extra credit

Rodent's Revenge

In this lab you will create the game Rodent's Revenge.

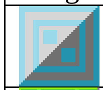
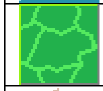
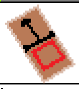
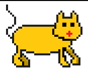
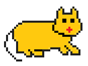
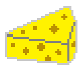
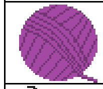
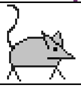
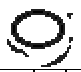
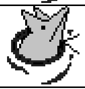
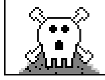
How the game works:

- The player plays as a mouse that is pursued by cats.
- The player dies each time he is within 1 square of a cat, moves onto a trap or is directly in front of a yarn balls path.
- The player gets trapped for 5 turns when he moves onto a hole.
- The play gains 50 points when he moves onto cheese and 1 point when the level changes.
- Cats are trapped when they cannot move. Trapped cats lay down.
- Once all the cats on screen are trapped they change to cheese. Next new cats spawn if there are still more cats for the level or the level changes.
- You have 3 lives in the game
- There will be 5 levels

Controls:

Key	Action
'w'	Moves the player up one square
's'	Moves the player down one square
'a'	Moves the player left one square
'd'	Moves the player right one square

Images Used:

Image	Description
	Wall
	Movable Wall
	Trap
	Moving Cat
	Trapped Cat
	Cheese
	Yarn
	Mouse
	Hole
	Hole with Mouse
	Dead Mouse

Your game will consist of a 23x23 map (920x920), with an additional area to the left of the map. The area to the left of the map will be used to show the level, lives, score and moves until new cats appear.

Coding:

Main:

	Description
Methods	
main	Creates a Rodent Frame

RodentFrame (extends JFrame):

	Description
Constructors	
RodentFrame()	• Creates/adds a RodentPanel • Adjusts it size to be big enough for the panel and the frames decorations

RodentPanel (extends JPanel & implements KeyListener)

	Description
Constructors	
RodentPanel()	• Sets its size • Creates the buffer • Loads the images • Calls reset
Attributes	
RodentGame game	Stores the game
Methods	
void reset()	Creates a new game
void paint(Graphics g)	Paints the game world to the screen.
void keyPressed(KeyEvent e)	Not used
void keyReleased(KeyEvent e)	Not used
void keyTyped(KeyEvent e)	• Handles all the players inputs • Calls Games Update when ‘w’, ‘s’, ‘d’ or ‘a’ is pressed
void addNotify()	• Calls super.addNotify() • Calls requestFocus()

RodentGame

	Description
Constructors	
RodentGame	• Sets status to PLAYING • Sets level to 1 • Sets score to 0 • Loadings values into levelNames • Calls loadLevel
Public Static Final Attributes	
int PLAYING	Stores 0 and is used for status.
int DEAD	Stores 1 and is used for status.
int WIN	Stores 2 and is used for status.
Attributes	
Mouse mouse	Stores the mouse
GamePiece[][] grid	Stores the map data
ArrayList<Integer> catTimes;	Stores what times the cats spawn for the current level
String[] levelNames	Stores the names of the level files
int score	Stores the players score
int time	Stores the number of moves since the level started
int status	Stores the status of the game PLAYING, DEAD or WIN
int level	Store the level number
Methods	
GamePiece get(int x, int y)	Returns the piece at loation (x,y) or null if the location is invalid
int getLevel()	Returns the current level number
String nextSpawn()	Returns a string containing the number of moves until new cats spawn or "NA" if there are no more cats left in the level.
GamePiece set(GamePeice gp, int x, int y)	• Returns null if (x,y) is out of bounds • When gp is null it will put null in spot (x,y) and return the old value of the location. • When gp is not null o The gp is placed in spot (x,y) o gp's x and y are corrected o the old value of the location is return
Mouse getMouse()	Returns the mouse
boolean loadLevel(int level)	Loads the given level number using the levelNames array to determine what file should be read in. 1 – Wall 0 – Open Ground 2 – Movable Wall m – Mouse y – Yarn c – Cat d – Dead Mouse h – Cheese o – Hole t – Trap Returns true if successful and false if unsuccessful.

void update()	When the status is playing will call update on all items in the grid and take care of any other needed game changes.
int getStatus()	Returns the game's status
ArrayList<GamePiece> getPieces()	Returns an ArrayList cantaining all the items in the game's grid.
void setScore(int score)	Changes the score to the received value
int getScore()	Returns the score
void setStatus(int status)	Changes the status to the received value
ArrayList<Cat> getCats()	Returns an ArrayList of the Cats in the game's grid
public distanceTo(Point p, GamePiece gp)	Returns the distance from gp to the specified point.

GamePiece (Abstact Class)

	Description
Constructors	
GamePiece(RodentGame game, int state, int x, int y)	• Sets game, state, x ,y, and loads images into the images list
Attributes	
int x	Stores the x position of the object
int y	Stores the y position of the object
ArrayList<BufferedImages>	Stores the images of the object
int state	Stores data about how the is object should behave.
RodentGame	Stores the game.
Methods	
abstract void act()	Method to be overwritten to create object behavior
abstract boolean canMove(int x, int y)	Method to be overwritten to determine if the object can move to the given location.
abstract BufferedImage getImage()	Returns the correct image based on the state of the object
ArrayList<Buffered> getImages()	Returns images
int getState()	Returns the state of the object.
void addImage(String name)	Loads the image with the given name from the images folder into the images list.
int getX()	Returns x.
int getY()	Returns y.
void setX(int x)	Changes x to the received value.
void setY(int y)	Changes y to the received value.
void setState(int state)	Changes state to the received value.
void move(int x, int y)	Moves the object to the given location if can move is true for the location. Note: Uses game.set

Cheese (Extends GamePiece)

	Description
Constructors	
Cheese(RodentGame game,int x, int y)	• Call super to set data • Adds the cheese image to images
Methods	
BufferedImage getImage()	Overrides getImages: returns the cheese image
boolean canMove(int x, int y)	Overrides canMove: Returns false
act()	Overrides act: has no code

Wall (Extends GamePiece)

	Description
Constructors	
Wall(RodentGame game,int x, int y)	• Call super to set data • Adds the movable wall image to images
Methods	
BufferedImage getImage()	Overrides getImages: returns the wall image
boolean canMove(int x, int y)	Overrides canMove: Returns false
act()	Overrides act: has no code

Trap (Extends GamePiece)

	Description
Constructors	
Trap(RodentGame game,int x, int y)	• Call super to set data • Adds the trap image to images
Methods	
BufferedImage getImage()	Overrides getImages: returns the trap image
boolean canMove(int x, int y)	Overrides canMove: Returns false
void act()	Overrides act: has no code

DeadMouse (Extends GamePiece)

	Description
Constructors	
DeadMouse(RodentGame game,int x, int y)	• Call super to set data and sets time to 10 • Adds the DeadMouse image to images
Attributes	
int time	Number of turns until the it is removed
Methods	
BufferedImage getImage()	Overrides getImages: returns the dead mouse image
boolean canMove(int x, int y)	Overrides canMove: Returns false
void act()	Overrides act: subtracts 1 from time and removes the itself from the game when time is 0

MovableWall (Extends GamePiece)

	Description
Constructors	
MovableWall(RodentGame game,int x, int y)	• Call super to set data • Adds the wall Image to images
Methods	
BufferedImage getImage()	Overrides getImages: returns the movable wall image
boolean canMove(int x, int y)	Overrides canMove: Returns true when the new location is 1 square away up /down/ left/ right and that location stores a null, cheese or another movable wall that can also move in the same direction. Otherwise the method returns false.
void move(int x, int y)	Overrides move(int x, int y): if can move is true for the given location the moveable wall will move the new location. When the new location has a movable wall that movable wall will be moved before the current movable wall is moved.
void act()	Overrides act: has no code

Hole (Extends GamePiece)

	Description
Constructors	
Hole(RodentGame game,int x, int y)	• Call super to set data • Sets mouse to null • Sets time to 5 • Adds the hole and hole with mouse images to images
Attributes	
Mouse mouse	Stores the mouse when it is in the hole
int time	Stores the number of turns until the mouse is released
Methods	
Mouse getMouse()	Returns mouse
void setMouse(Mouse mouse)	Changes mouse to the received value
BufferedImage getImage()	Overrides getImages: returns the hole or hole with mouse image depending on if the hole has a mouse
boolean canMove(int x, int y)	Overrides canMove: Returns false
void move(int x, int y)	Overrides move(int x, int y): has no code
void act()	Overrides act: • When mouse is null it does nothing • When mouse is not null ○ Subtracts 1 from time ○ Sets the mouse's state to MOVING ○ When time is 0 the hole is removed and the mouse is placed where the hole was

Mouse (Extends GamePiece)

	Description
Constructors	
Mouse(RodentGame game,int x, int y)	• Call super to set data • Sets lives to 3 • Adds the moving mouse image to images
Public Static Final Attributes	
int MOVING	Stores 0 and is used for state
int TRAPPED	Stores 1 and is used for state
Attributes	
int lives	Stores the number of lives the mouse has left
Methods	
int getLives()	Returns lives
BufferedImage getImage()	Overrides getImages: returns the moving mouse image
void respawn()	Preforms the following operations: • Subtracts 1 from lives • When lives reaches 0 the game is ended • When lives is 1 or more ○ Respawns the mouse in a random safe location
boolean canMove(int x, int y)	Overrides canMove: Returns true when the new location is 1 square away up /down/ left/ right and that location stores a null, cheese, trap, hole or a movable wall that can also move in the same direction. Otherwise the method returns false.
void move(int x, int y)	Overrides move(int x, int y): if can move is true for the given location it will move to the new location, following these rules: • Cheese – Replaces the Cheese and increases the score by 50 • Trap – Replaces the trap and calls respawn • Hole – Adds itself to the hole • null – move to the new location • Movable wall – Moves the wall and then takes its old location After moving the move respawns if it is directly in the path of a yarn ball.
void act()	Overrides act: has no code

Cat (Extends GamePiece)

	Description
Constructors	
Cat(RodentGame game,int x, int y)	• Call super to set data • Sets state to MOVING • Adds the moving cat and trapped cat images to images
Cat(RodentGame game,int x, int y)	• Call super to set data • Sets state to MOVING • Adds the cat at a location that is far from the mouse
Public Static Final Attributes	
int MOVING	Stores 0 and is used for state
int TRAPPED	Stores 1 and is used for state
Methods	
BufferedImage getImage()	Overrides getImages: returns the moving cat or trapped cat image depending on the state
boolean canMove(int x, int y)	Overrides canMove: Returns true if the location is 1 square away in any direction and stores null.
void eat()	Call respawn on any mouse that is one square away in any direction.
void act()	Overrides act: Preforms the following actions in order: • Calls eat • If it has a valid move ○ Moves to the neighboring empty square that is the closest the moues ○ Changes state to MOVING • If it does not have a valid move ○ Changes state to Trapped • Calls eat

Yarn (Extends GamePiece)

	Description
Constructors	
Yarn(RodentGame game,int x, int y)	• Call super to set data • Sets state randomly • Adds the yarn image to images
Public Static Final Attributes	
int NORTH	Stores 0 and is used for state
int SOUTH	Stores 1 and is used for state
int EAST	Stores 2 and is used for state
int WEST	Stores 3 and is used for state
int NORTH_EAST	Stores 4 and is used for state
int NORTH_WEST	Stores 5 and is used for state
int SOUTH_EAST	Stores 6 and is used for state
int SOUTH_WEST	Stores 7 and is used for state
Attributes	
int direction	Stores the direction the ball is rolling.
Methods	
BufferedImage getImage()	Overrides getImages: returns the yarn image
boolean canMove(int x, int y)	Overrides canMove: Returns true if the location is 1 square away in any direction and stores Cheese, or null.

void flatten()	Call respawn on any mouse that is one square away in direction the yarn is rolling
void act()	Overrides act: Preforms the following actions in order: • Calls flatten • If it the next square in it's direction is empty it moves there • If it does not have a valid move ○ Changes state to Trapped • Calls eat

NOTE: You may add other variables and methods as needed!!!

Other Grades:

- RodentMain
- RodentFrame
- Trap / Cheese / Wall / DeadMouse

Minor Grades:

- GamePiece
- RodentPanel
- RodentGame
- Mouse / Cat
- Hole / Yarn / MovableWall

Major Grades:

- Project

Rubric (1 major grade)

Points	Task
10	Working Trap / Cheese / Wall / DeadMouse
20	Working Hole / Yarn / MovableWall
20	Working Mouse / Cat
20	Controls Work
10	Levels
10	Win / Losing / Restarting
10	Side Panel Data