**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** MineSweeper **extends** JFrame {

**static** **final** **int** ***LEFTCLICK*** = 1;

**static** **final** **int** ***RIGHTCLICK*** = 3;

//declares the variables in the class MineSweeper so they can be used throughout the class

**char** board[][]; //to keep track of the position of mines and numbers

JButton buttons[][]; //to hold the buttons

JButton changeSetting = **new** JButton("Configuration"); //button to change difficulty/start a new game

**int** nBombs;

**int** count; //to keep track of the amount of mines left

JLabel countLabel; //to show to the user the value of count

**int** size, percentage;

**public** MineSweeper(**int** s, **int** per) {

size = s;

percentage = per;

board= **new** **char**[size][size];

buttons= **new** JButton[size][size];

nBombs = (**int**)((percentage / 100.0) \* (size \* size)); //determines the amount of bombs that will be put in board

count = nBombs; //count starts with the number of mines

countLabel = **new** JLabel();

JPanel bottom = **new** JPanel();

changeSetting.addActionListener(**new** configuration()); //add action listener to the configuration button

bottom.add(countLabel); //add the count and the configuration button to the bottom panel

bottom.add(changeSetting);

setBombs(nBombs); //initialize the board with numbers and mines

JPanel panel = **new** JPanel(); //panel to hold the buttons

panel.setLayout(**new** GridLayout(size, size)); //change to a grid layout

**for**(**int** i=0; i<size; i++)

**for**(**int** j=0; j<size; j++) { //add buttons to the array of buttons and to the grid

buttons[i][j] = **new** JButton("");

panel.add(buttons[i][j]);

buttons[i][j].addMouseListener(**new** buttonClicked(i,j)); //add mouse listener to each button according to their location

}

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***); //establish default settings of the window

setLocation(500,100);

setSize(900,900);

setLayout(**new** BorderLayout());

setVisible(**true**);

add(panel, BorderLayout.***CENTER***); //add the buttons to the center

add(bottom, BorderLayout.***SOUTH***); //add the count and the configuration button to the bottom

countLabel.setText("Mine left: "+ count);

}

**private** **class** configuration **implements** ActionListener { //if configuration button is pressed

**public** **void** actionPerformed(ActionEvent e) {

setVisible(**false**); //current window disappear

**new** menu(); //starts the menu window

}

}

**void** lose(){ //method if player loses

**for**(**int** i=0; i<size; i++)

**for**(**int** j=0; j<size; j++)

**if**(board[i][j] == 'x') {

buttons[i][j].setBackground(Color.***RED***); //display position of all mines

buttons[i][j].setText("X");

}

JOptionPane.*showMessageDialog*(**null**, "You lost!"); //display loss message

setVisible(**false**); //current window disappears

**new** MineSweeper(size, percentage); //a new window appears

}

**void** check(){ //check after each play if the player won

**int** goal = size\*size; //determine the amount of cases

**int** count = 0;

**for**(**int** i=0; i<size;i++)

**for**(**int** j=0; j<size;j++)

**if**(board[i][j] == 'p' || board[i][j] == 'x') //counts number of exposed/mine cases

count++;

**if**(count == goal){ //if all cases are exposed, player wins

JOptionPane.*showMessageDialog*(**null**, "Good job you won!!"); //display winning message

setVisible(**false**); //current window disappears

**new** MineSweeper(size, percentage); //starts a new game

}

}

**void** reveal (**int** r, **int** c) { //method when a button is pressed

**if**(board[r][c] == 'x') //if location has a mine, launch lose method

lose();

**else** **if**(board[r][c] != ' ') { //if location has a number

buttons[r][c].setBackground(Color.***YELLOW***); //set background color to yellow

buttons[r][c].setText("" + board[r][c]); //set text to the number from board

board[r][c] = 'p'; //set the location on board to p-> in mouse listener, p is ignore or endless recursion

check(); //check if player won

}

**else** **if**(board[r][c] == ' ') { //if location is blank

buttons[r][c].setBackground(Color.***CYAN***); //set color to cyan

buttons[r][c].setText(" "); //set text to " ", so that flag cannot be set on it-> see mouselistener

board[r][c] = 'p'; //set location on board to p

check(); //check if player won

**for**(**int** i=(r-1); i<=(r+1); i++) //reveals surrounding cases that aren't mines

**for**(**int** j=(c-1); j<=(c+1); j++)

**if**(i>=0 && i<size && j>=0 && j<size && board[i][j] != 'x' && board[i][j] != 'p')

reveal(i,j); //reveals with recursion, if case surrounding case isn't exposed (->location on board isn't p)

}

}

**private** **class** buttonClicked **implements** MouseListener {

**int** r, c;

**public** buttonClicked (**int** row, **int** col) {

r = row;

c = col;

}

**public** **void** mousePressed(MouseEvent e) {

**if**(e.getButton() == ***LEFTCLICK***) { //if left click, reveals the case

**if**(board[r][c] != 'p')

reveal(r, c);

}

**else** **if**(e.getButton() == ***RIGHTCLICK***) { //if right click, set/unset a flag

**if**(buttons[r][c].getText() == ""){ //if flag is unset and case not revealed

buttons[r][c].setText("|>"); //change text to flag

count--; //decrease the numnber of unknown mines left

countLabel.setText("Mine left: " + count); //updates the label

}

**else** **if**(buttons[r][c].getText() == "|>"){ //if flag is already set on case

buttons[r][c].setText(""); //unset the flag

count++; //increase the count

countLabel.setText("Mine left: " + count); //update the label

}

}

}

**public** **void** mouseClicked(MouseEvent arg0) {

}

**public** **void** mouseEntered(MouseEvent arg0) {

}

**public** **void** mouseExited(MouseEvent arg0) {

}

**public** **void** mouseReleased(MouseEvent arg0) {

}

}

**void** setBombs(**int** nBombs) { //method to set the bombs in the board and numbers for non-bomb cases

**for**(**int** i=0; i < size; i++)

**for**(**int** j=0; j < size; j++)

board[i][j] = ' '; //reset the board

**int** row, column;

**for**(**int** i=0; i < nBombs; i++) { //assign each bomb to a case

**do** {

row = (**int**) (Math.*random*() \* size); //get random location in board from random width and height

column = (**int**) (Math.*random*() \* size);

} **while**(board[row][column] == 'x'); //makes sure this location isn't already assigned with a bomb

board[row][column] = 'x';

}

**for**(**int** i=0; i<size; i++)

**for**(**int** j=0; j<size; j++)

**if**(board[i][j] != 'x') { //assign number of bombs surrounding each non-bomb case

**int** number = 0;

**for**(**int** r = i-1; r <= i+1; r++) //goes from width location -1 to width location +1

**for**(**int** c = j-1; c <= j+1; c++) //goes from height location -1 to height location +1

**if**( r>=0 && r< size && c>=0 && c<size && board[r][c] == 'x') {

//check if location to check is in the board then if there's a bomb add 1 to the number displayed

number++;

}

**if**(number != 0) // assign number to case unless the number is 0

board[i][j] = (**char**) (number+ 48); //add 48 so to match the character of the number from 0-9

}

}

}