**import** javax.swing.JOptionPane;

**import** java.util.Scanner;

/\*

\* Justin Mendes

\* Created: August 2, 2017

\* Last Edited: August 5, 2017

\* Unit 1 Activity 1 Program/Question 1

\* This program will take the prompt for the school, amount of donations, and population of school to get a rough estimate of the school's donations

\*/

**public** **class** FundRaising

{

**public** **static** **void** main(String[] args)

{

//for restarting the donations

**int** restart = 1;

**while** (restart == 1)

{

//Variable Declarations and Initializations

**double** donations[][] = **new** **double**[9][4], totalDonations = 0;

**int** school, amount, population;

Scanner sc = **new** Scanner(System.***in***);

school = Integer.*parseInt*(JOptionPane.*showInputDialog*(**null**,"Which school is doing the fundraising?\n0 - Catholic Central\n1 - Holy Cross\n2 - Saint John Paul II\n3 - Saint Mother Teresa\n"

+ "4 - Regina Mundi\n5 - St. Joseph\n6 - St. Mary\n7 - St. Thomas Aquinas\n9 - EXIT", "School Selection", JOptionPane.***QUESTION\_MESSAGE***));

amount = Integer.*parseInt*(JOptionPane.*showInputDialog*(**null**,"What is the donation amount?\n0 - 25 cents\n1 - 50 cents\n2 - 1 dollar\n3 - 2 dollars", "Donation Amount Selection", JOptionPane.***QUESTION\_MESSAGE***));

population = Integer.*parseInt*(JOptionPane.*showInputDialog*(**null**,"What is the school population?", "School Population", JOptionPane.***QUESTION\_MESSAGE***));

**switch**(amount)

{

**case** 0: donations[school][amount] = population \* 0.25;

**break**;

**case** 1: donations[school][amount] = population \* 0.50;

**break**;

**case** 2: donations[school][amount] = population;

**break**;

**case** 3: donations[school][amount] = population \* 2;

**break**;

}//end switch

donations[8][amount] += donations[school][amount];

System.***out***.println("Fundraising\n===============================================================================");

System.***out***.println("\tCathCen Holy C JP II MotherT ReginaM St.Joe St.Mary St.Thom TOTAL");

**for**(**int** horizontal = 0; horizontal < 4; horizontal++)

{

**for** (**int** vertical = 0; vertical < 9; vertical++)

{

**if**(vertical == 0)

{

**switch**(horizontal)

{

**case** 0: System.***out***.print("$0. 25 - ");

**break**;

**case** 1: System.***out***.print("$0. 50 - ");

**break**;

**case** 2: System.***out***.print("$1. 00 - ");

**break**;

**case** 3: System.***out***.print("$2. 00 - ");

**break**;

}//end switch

}//end if

System.***out***.print(donations[vertical][horizontal] + "\t");

totalDonations = donations[vertical][horizontal];

}//end loop

System.***out***.println();

}//end loop

System.***out***.print("TOTAL DONATIONS = " + totalDonations + "\n");

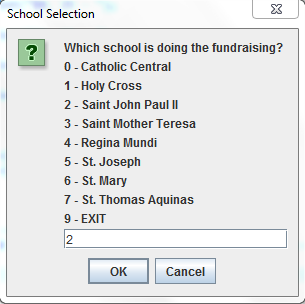
System.***out***.println("Press 1 to add another donation\nPress anything else to exit.");

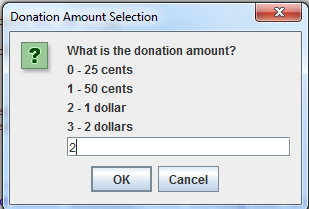
restart = sc.nextInt();

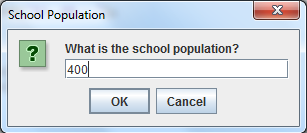
}//end loop

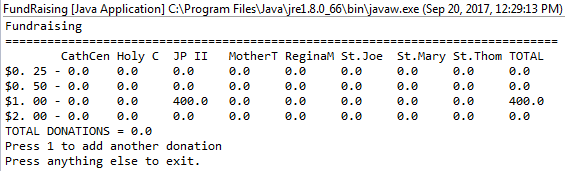
}//end main

}//end class









**import** javax.swing.JOptionPane;

/\*

\* Justin Mendes

\* Created: August 2, 2017

\* Last Edited: August 15, 2017

\* Unit 2 Activity 2 Program/Question 2

\* This program will prompt the user to type letters or guess the word in a game of Hangman

\*/

**public** **class** Hangman

{

**static** String *wrongGuess* = "";

**static** StringBuilder *display*;

**static** **int** *categorySpot*, *chances* = 7;

**public** **static** **void** main(String[] args)

{

//Variable Declarations and Initializations

String phrases[][] =

{

{"Zoo Animals", "Polar Bear", "Monkey", "Seal", "Monkey", "Tiger", "Hippo", "Alligator", "Ostrich", "Falcon", "Orangutang"},

{"PC Games", "League of Legends", "Overwatch", "Minecraft", "World of Warcraft", "Team Fortress 2", "Counter Strike", "Tetris", "Portal 2", "Skyrim", "The Witcher 3"},

{"Books", "Harry Potter", "Hunger Games", "Percy Jackson", "Twilight", "The Diary of a Wimpy Kid", "To Kill a Mockingbird", "A Series Of Unfortunate Events", "Green Eggs and Ham", "The City of Bones", "The Chronicles of Narnia"},

{"Sports Teams", "Toronto Raptors", "New York Yankees", "Manchester United", "Dallas Cowboys", "Los Angeles Lakers", "Boston Red Sox", "Chicago Bears", "New York Mets", "New Orleans Pelicans"}

};

String guess = " ";

**char** stickman[][] = **new** **char**[3][5];

*categorySpot* = *randomWholeNumber*(0, 3);

String category = phrases[*categorySpot*][0], chosenPhrase = phrases[*categorySpot*][*randomWholeNumber*(1, 10)];

*display* = **new** StringBuilder(chosenPhrase.replaceAll("\\S", "-"));

**while**(*chances* > 0)

{

*letterGuess*(chosenPhrase, guess);

System.***out***.println("\nCategory: " + category + "\nNumber of Words: " + *numberOfWords*(chosenPhrase) + "\n" + *display*);

guess = JOptionPane.*showInputDialog*(**null**,"Number of guesses remaining: " + *chances* + "\nWrong guesses so far:" + *wrongGuess* +"\n\nGuess a letter or number:\nOR\ntype \"guess\" to guess the word or the phrase:", "Hangman", JOptionPane.***QUESTION\_MESSAGE***);

**if**(guess.equalsIgnoreCase("guess"))

{

*chances* = *wordGuess*(*chances*, chosenPhrase);

}//end if

**else** **if**(!chosenPhrase.toUpperCase().contains(guess.substring(0, 1).toUpperCase()))//to make sure only the first value in the guess is looked upon & else if as the user has to type in "guess" (thus the g will be considered as the guess)

{

*chances*--;

**switch**(*chances*)

{

**case** 6: stickman[1][0] = 'O';

**break**;

**case** 5: stickman[0][1] = '/';

**break**;

**case** 4: stickman[1][1] = '|';

**break**;

**case** 3: stickman[2][1] = '\\';

**break**;

**case** 2: stickman[1][2] = '|';

**break**;

**case** 1: stickman[0][3] = '/';

**break**;

**case** 0: stickman[2][3] = '\\';

*endingScreen*(chosenPhrase, 0);

**break**;

}//end switch

**for**(**int** horizontal = 0; horizontal < 5; horizontal++)

{

**for** (**int** vertical = 0; vertical < 3; vertical++)

{

System.***out***.print(stickman[vertical][horizontal]);

}//end loop

System.***out***.println();

}//end loop

}//end if

**if**(*display*.toString().equalsIgnoreCase(chosenPhrase))

{

*endingScreen*(chosenPhrase, *chances*);

}//end if

}//end while

}//end main

**public** **static** **int** wordGuess(**int** chances, String phrase)

{

String guessCheck;

guessCheck = JOptionPane.*showInputDialog*(**null**,"Enter the word or the WHOLE phrase", "Hangman", JOptionPane.***QUESTION\_MESSAGE***);

**if**(phrase.equalsIgnoreCase(guessCheck))

{

*endingScreen*(phrase, chances);

**return** chances;

}//end if

**else**

{

**return** chances - 1;

}//end else

}//end method wordGuess

**public** **static** **int** randomWholeNumber(**int** lowVal, **int** highVal)

{

**int** randomNumber = -1;

**while** (randomNumber > highVal || randomNumber < lowVal)

{

randomNumber = (**int**) ((Math.*random*() \* highVal) + lowVal);

}//end loop

**return** randomNumber;

}//end method randomWholeNumber

**public** **static** **void** letterGuess(String phrase, String guess)

{

**for**(**int** i = 0; i < phrase.length(); i++)

{

**if**(phrase.toUpperCase().charAt(i) == guess.toUpperCase().charAt(0))

{

*display*.setCharAt(i, phrase.toUpperCase().charAt(i));

}//end if

}//end loop

**if**(!phrase.toUpperCase().contains(guess.toUpperCase()))

{

*wrongGuess* += guess.toUpperCase() + " ";

}//end if

}//end method displayLetters

**public** **static** **int** numberOfWords(String phrase)

{

//to split up the phrase string into an array

//by every space and take that length for each word

String[] words = phrase.split(" ");

**return** words.length;

}//end method numberOfWords

**public** **static** **void** endingScreen(String phrase, **int** chances)

{

**if**(chances != 0)

{

System.***out***.println("===================================\nYou have revealed the right word/phrase/title (" + phrase + ") with " + chances + " chances! Congratulations!");

System.*exit*(0);

}//end if

**else**

{

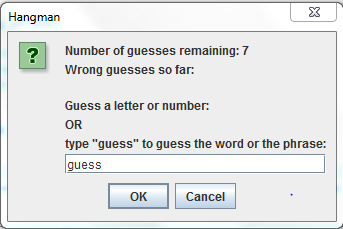
System.***out***.println("===================================\nUh oh! GAME OVER! Nice try guessing the word/phrase/title (" + phrase + ")");

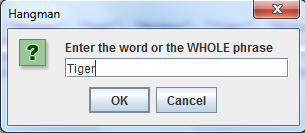
System.*exit*(0);

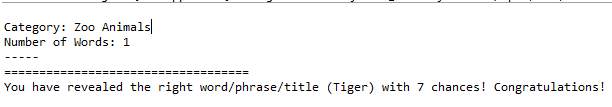
}//end else

}//end method endingScreen

}//end class







**import** javax.swing.JOptionPane;

/\*

\* Justin Mendes

\* Created: August 10, 2017

\* Last Edited: August 27, 2017

\* Unit 2 Activity 2 Program/Question 3

\* This program will simulate a simple life form.

\*/

**public** **class** GameOfLife

{

**static** **int** *generation* = 0;

**public** **static** **void** main(String[] args)

{

//Variable Declarations and Initializations

**boolean**[][] cells = **new** **boolean**[20][20];

**int** numberOfCells;

**int** random1, random2;//these are to store the random whole numbers so the method will not change the random number each time

String advance = "1";

numberOfCells = Integer.*parseInt*(JOptionPane.*showInputDialog*(**null**,"Welcome to the game of life!\nPlease enter how many cells you would like to enter the game with.", "Game of Life Initialization", JOptionPane.***QUESTION\_MESSAGE***));

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

{

random1 = *randomWholeNumber*();

random2 = *randomWholeNumber*();

**if**(cells[random1][random2] == **false**)//to initialize all of the new alive cells

{

cells[random1][random2] = **true**;

}//end if

**else**//to make sure that there will be the full number of randomly placed living cells (when the random spot is already a living cell)

{

i--;

}//end else

}//end loop

**while**(advance.equals("1"))

{

*printTable*(cells);

cells = *lifeConditions*(cells);

advance = JOptionPane.*showInputDialog*(**null**,"1 - Advance one generation.\nexit - exit game", "Game of Life Advance", JOptionPane.***QUESTION\_MESSAGE***);

}//end loop

}//end main

**public** **static** **int** randomWholeNumber()

{

**int** randomNumber = -1;

**while** (randomNumber > 20 || randomNumber < 0)

{

randomNumber = (**int**) ((Math.*random*() \* 20));

}//end loop

**return** randomNumber;

}//end method randomWholeNumber

**public** **static** **void** printTable(**boolean**[][] cells)

{

System.***out***.println("\n----Generation " + *generation* + "-----------------------");

**for**(**int** column = 0; column < 20; column++)

{

**for**(**int** row = 0; row < 20; row++)

{

**if**(cells[column][row] == **true**)

{

System.***out***.print("O ");

}//end if

**else**

{

System.***out***.print(". ");

}//end else

}//end loop

System.***out***.println();

}//end loop

*generation*++;

}//end method printTable

**public** **static** **boolean**[][] lifeConditions(**boolean**[][] cells)

{

**boolean** conditionChecked;//to make sure that only one if/else condition is chosen

**int** neighbours = 0;//number of neighbours surrounding the chosen cell

**for**(**int** column = 0; column < 20; column++)

{

**for**(**int** row = 0; row < 20; row++)

{

neighbours = *neighbourCheck*(neighbours, cells, column, row);

conditionChecked = **false**;

**if**(cells[column][row] == **true** && neighbours == 2 || neighbours == 3)//cell stays alive with 2-3 alive cell neighbours

{

//it will stay true(alive) so nothing has to happen

conditionChecked = **true**;

}//end if

**else** **if** (cells[column][row] == **true**)//to avoid the default to the code in the scope below (using only else)

{

cells[column][row] = **false**;//cell death

conditionChecked = **true**;

}//end else

**if**(cells[column][row] == **false** && neighbours == 3 && conditionChecked == **false**)//if a dead cells has exactly 3 surrounding alive cell neighbours

{

cells[column][row] = **true**;//it becomes alive

}//end if

}//end loop

}//end loop

**return** cells;

}//end method lifeConditions

**public** **static** **int** neighbourCheck(**int** neighbours, **boolean**[][] cells, **int** column, **int** row)//to handle the Array Out of Bounds Exception (without extending the array further out)

{

neighbours = 0;//this is reset to 0 each time for each new cell position neighbour check

**if**(column - 1 >= 0)//for all of the cells to the LEFT(including LEFT-DIAGONALS)

{

**if**(row - 1 >= 0)

{

**if**(cells[column - 1][row - 1] == **true**)//TOP-LEFT

{

neighbours++;

}//end if

}//end if

**if**(cells[column - 1][row] == **true**)//MIDDLE-LEFT

{

neighbours++;

}//end if

**if**(row + 1 < 20)

{

**if**(cells[column - 1][row + 1] == **true**)//BOTTOM-LEFT

{

neighbours++;

}//end if

}//end if

}//end if

**if**(row - 1 >= 0)//for the TOP-MIDDLE

{

**if**(cells[column][row - 1] == **true**)

{

neighbours++;

}//end if

}//end if

**if**(row + 1 < 20)//for the BOTTOM-MIDDLE

{

**if**(cells[column][row + 1] == **true**)//nested if because it will cause the out of bounds exception if the first if condition is not met

{

neighbours++;

}//end if

}//end if

**if**(column + 1 < 20)//for all of the cells to the RIGHT(including RIGHT-DIAGONALS)

{

**if**(row - 1 >= 0)

{

**if**(cells[column + 1][row - 1] == **true**)//TOP-RIGHT

{

neighbours++;

}//end if

}//end if

**if**(cells[column + 1][row] == **true**)//MIDDLE-RIGHT

{

neighbours++;

}//end if

**if**(row + 1 < 20)

{

**if**(cells[column + 1][row + 1] == **true**)//BOTTOM-RIGHT

{

neighbours++;

}//end if

}//end if

}//end if

**return** neighbours;

}//end method neighbourCheck

}//end class

