import java.io.\*;

import java.util.\*;

import java.util.concurrent.TimeUnit;

public class Games {

private static final File *GamesFile* = new File("Games.txt");

private static final File *GamesTempFile* = new File("GamesTemp.txt");

private static int *gamePoints* = 0;

private static int *nimCompScore* = 0;

private static int *nimUserScore* = 0;

private static int *battleshipCompScore* = 0;

private static int *battleshipUserScore* = 0;

private static int *yahtzeeHighScore* = 0;

private static int *bowlingHighScore* = 0;

//utility class

private Games() {

}

public static int GetGamePoints() {

*ReadGamesFile*();

return *gamePoints*;

}

public static void DeleteGamesFile() {

if (!*GamesFile*.delete()) {

System.*out*.println("Could not delete Games file. Must manually delete.");

} else {

System.*out*.println("Deleted Games file.");

}

}

public static void AddGamePoints() throws IOException {

*ReadGamesFile*();

*gamePoints*++;

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(*GamesTempFile*));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (*GamesTempFile*.createNewFile()) {

System.*out*.println("New temporary games file created.");

}

} catch (Exception e) {

System.*out*.println("An error occurred while creating a temporary games file.");

}

gamesFileReader.nextLine();

gamesPw.println(*gamePoints*);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!*GamesFile*.delete()) {

System.*out*.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!*GamesTempFile*.renameTo(*GamesFile*)) {

System.*out*.println("Could not rename temporary file.");

}

}

//subtracts one point from the user's game points

public static void SubtractGamePoints() throws IOException {

//reads the games file

*ReadGamesFile*();

//subtracts one from the game points variable

*gamePoints*--;

//replaces the new points with the old one

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(*GamesTempFile*));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (*GamesTempFile*.createNewFile()) {

System.*out*.println("New temporary games file created.");

}

} catch (Exception e) {

System.*out*.println("An error occurred while creating a temporary games file.");

}

gamesFileReader.nextLine();

gamesPw.println(*gamePoints*);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!*GamesFile*.delete()) {

System.*out*.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!*GamesTempFile*.renameTo(*GamesFile*)) {

System.*out*.println("Could not rename temporary file.");

}

}

//creates the games file if one does not exist

public static void CreateGamesFile() {

try {

if (*GamesFile*.createNewFile()) {

System.*out*.println("New games file created.");

} else {

System.*out*.println("The games file already exists.");

}

} catch (Exception e) {

System.*out*.println("An error occurred while creating the games file. Please try again.");

e.printStackTrace();

}

try (FileWriter fw = new FileWriter("Games.txt", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter pw = new PrintWriter(bw)) {

pw.println(*gamePoints* + "\n" + *nimCompScore* + "\n" + *nimUserScore* + "\n" + *battleshipCompScore* + "\n" +

*battleshipUserScore* + "\n" + *yahtzeeHighScore* + "\n" + *bowlingHighScore*);

} catch (Exception e) {

System.*out*.println("An error occurred while writing to the Games file.");

}

}

//reads the Games file

public static void ReadGamesFile() {

try {

Scanner fileReader = new Scanner(new FileInputStream("Games.txt"));

//no repeats are needed, so variables are written directly to their corresponding variables

//order stays constant, so what the values of all lines means are known

*gamePoints* = fileReader.nextInt();

*nimCompScore* = fileReader.nextInt();

*nimUserScore* = fileReader.nextInt();

*battleshipCompScore* = fileReader.nextInt();

*battleshipUserScore* = fileReader.nextInt();

*yahtzeeHighScore* = fileReader.nextInt();

*bowlingHighScore* = fileReader.nextInt();

fileReader.close();

} catch (IOException ex) {

System.*out*.println("Error while reading games file. Please try again.");

}

}

//nim game, game where the user takes rocks against computer and the one to take the last one loses

public static void Nim() throws IOException, InterruptedException {

*ReadGamesFile*(); //reads games file to get current scores

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

int stones;

boolean turnRestart = true;

//prints title screen

System.*out*.println("<---------------------------------------------------------------->\n" +

" 00 0 00000 00 00 000 \n" +

" 0 0 0 0 0 0 0 0 0 0 \n" +

" 0 0 0 0 0 0 0 000 000 000 \n" +

" 0 00 0 0 0 0 0 0 0\n" +

" 0 0 00000 0 0 000 000 \n" +

"<---------------------------------------------------------------->\n");

TimeUnit.*SECONDS*.sleep(2);

System.*out*.println("\nNim (Complex Stone Simulator)" + "\n" + //print out the game rules

"RULES:" + "\n" +

"A random number of stones between 15 to 30 is chosen," + "\n" +

"You and the computer take turns taking 1 to 3 stones," + "\n" +

"The side that takes the last stone loses," + "\n" +

"The counter represents your win loss ratio." + "\n");

//Generates a random number from 15 to 30

stones = (int) (30 - (Math.*random*() \* 14.99999999999999999999999999999999999999999999999999999999999));

while (turnRestart) {

String stonePrint;

String plurality;

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

if (stones == 1) { //checks to ensure proper grammar

stonePrint = " stone.";

plurality = "There is ";

} else {

stonePrint = " stones.";

plurality = "There are ";

}

System.*out*.println(plurality + stones + stonePrint); //prints out statement on state of the game

stones = *NimComputerTurn*(stones); //calls the computerTurn method

if (stones <= 0) { //if it is computer's turn and there are no stones left

*nimUserScore*++; //add one to the players counter

System.*out*.println("There are " + stones + " stones."); //Statements declaring you have won

System.*out*.println("You've won against the computer! It has given up hope, and is now crying in the corner. " +

"The game will now reset, but you get the first move.");

System.*out*.println("The score is now Computer: " + *nimCompScore* + " to Player: " + *nimUserScore* + ".");

System.*out*.println("Writing new score to text file.");

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(*GamesTempFile*));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (*GamesTempFile*.createNewFile()) {

System.*out*.println("New temporary games file created.");

}

} catch (Exception e) {

System.*out*.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(*nimUserScore*);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!*GamesFile*.delete()) {

System.*out*.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!*GamesTempFile*.renameTo(*GamesFile*)) {

System.*out*.println("Could not rename temporary file.");

}

System.*out*.println("New score successfully written");

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

System.*out*.println("Would you like to play again? (Type yes or no)");

String playAgain = input.nextLine();

if (playAgain.equalsIgnoreCase("yes")) {

System.*out*.println("Great! Resetting stones...");

//Generates a random number from 15 to 30

stones = (int) (30 - (Math.*random*() \* 14.99999999999999999999999999999999999999999999999999999999999));

} else {

turnRestart = false;

}

}

if (stones == 1) {

stonePrint = " stone.";

plurality = "There is ";

} else {

stonePrint = " stones.";

plurality = "There are ";

}

System.*out*.println(plurality + stones + stonePrint); //prints out statement on state of the game

stones = *NimPlayerTurn*(stones); //calls the playerTurn method

if (stones == -1) {

System.*out*.println("Returning to main menu.");

return;

}

if (stones <= 0) { //if it is not the computers turn and there are no stones left

*nimCompScore*++; //adds one to the computer's counter

System.*out*.println("There are " + stones + " stones."); //statements declaring you've lost

System.*out*.println("Oh no, you've lost... I think you should take a break, and go back to working.");

System.*out*.println("The score is now Computer: " + *nimCompScore* + " to Player: " + *nimUserScore* + ".");

System.*out*.println("Writing new score to text file.");

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(*GamesTempFile*));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (*GamesTempFile*.createNewFile()) {

System.*out*.println("New temporary games file created.");

}

} catch (Exception e) {

System.*out*.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(*nimCompScore*);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!*GamesFile*.delete()) {

System.*out*.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!*GamesTempFile*.renameTo(*GamesFile*)) {

System.*out*.println("Could not rename temporary file.");

}

System.*out*.println("New score successfully written");

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

System.*out*.println("Would you like to play again? (Type yes or no)");

String playAgain = input.nextLine();

if (playAgain.equalsIgnoreCase("yes")) {

System.*out*.println("Great! Resetting stones...");

//generates new random number

stones = (int) (30 - (Math.*random*() \* 14.99999999999999999999999999999999999999999999999999999999999));

} else {

turnRestart = false;

}

}

}

}

private static int NimPlayerTurn(int stones) { //playerTurn method

boolean restart = true;

int stonesTake;

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

while (restart) { //repeats this code if user enters improper value

restart = false;

System.*out*.println("It's your turn. How many stones do you want to take? (Enter -1 to exit)"); //asks user how many stones they want

stonesTake = input.nextInt();

if (stonesTake == -1) {

stones = -1;

} else if (stonesTake > stones) { //watches to make sure the stones wanted are not greater than the stones that exist

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

System.*out*.println("You can't take more stones than the stones that exist.");

restart = true;

} else if (stonesTake == 1 || stonesTake == 2 || stonesTake == 3) { //if input is proper

stones = stones - stonesTake;

} else {

System.*out*.println(" "); //else prompt user to try again

System.*out*.println("Enter an integer value between 1 and 3.");

restart = true;

}

}

return stones; //returns remaining stones

}

private static int NimComputerTurn(int stones) { //computerTurn method

int stonesSubtracted = 0;

if (stones > 29) { //subtracts one if the number is thirty

stones = stones - 1;

stonesSubtracted = 1;

}

if (stones > 25 && stones < 29) { //ONLY DESCRIBED ONCE SINCE THIS REPEATS FOR DIFFERENT PARAMETERS

stones = stones - 1; //subtract one stone

stonesSubtracted = 1; //store how much is being subtracted

if (stones != 25) { //if stone is not desired value

stones = stones - 1; //subtract another stone

stonesSubtracted = 2; //store how much is being subtracted

if (stones != 25) { //if stone is not desired value

stones = stones - 1; //subtract another stone

stonesSubtracted = 3; //store how much is being subtracted

}

}

}

if (stones > 21 && stones < 25) {

stones = stones - 1;

stonesSubtracted = 1;

if (stones != 21) {

stones = stones - 1;

stonesSubtracted = 2;

if (stones != 21) {

stones = stones - 1;

stonesSubtracted = 3;

}

}

}

if (stones > 17 && stones < 21) {

stones = stones - 1;

stonesSubtracted = 1;

if (stones != 17) {

stones = stones - 1;

stonesSubtracted = 2;

if (stones != 17) {

stones = stones - 1;

stonesSubtracted = 3;

}

}

}

if (stones > 13 && stones < 17) {

stones = stones - 1;

stonesSubtracted = 1;

if (stones != 13) {

stones = stones - 1;

stonesSubtracted = 2;

if (stones != 13) {

stones = stones - 1;

stonesSubtracted = 3;

}

}

}

if (stones > 9 && stones < 13) {

stones = stones - 1;

stonesSubtracted = 1;

if (stones != 9) {

stones = stones - 1;

stonesSubtracted = 2;

if (stones != 9) {

stones = stones - 1;

stonesSubtracted = 3;

}

}

}

if (stones > 5 && stones < 9) {

stones = stones - 1;

stonesSubtracted = 1;

if (stones != 5) {

stones = stones - 1;

stonesSubtracted = 2;

if (stones != 5) {

stones = stones - 1;

stonesSubtracted = 3;

}

}

}

if (stones > 1 && stones < 5) {

stones = stones - 1;

stonesSubtracted = 1;

if (stones != 1) {

stones = stones - 1;

stonesSubtracted = 2;

if (stones != 1) {

stones = stones - 1;

stonesSubtracted = 3;

}

}

}

if (stonesSubtracted == 0) { //if nothing has been subtracted

if (stones == 1) { //if there is one stone left, only take one

stones = 0;

stonesSubtracted = 1;

} else {

stonesSubtracted = (int) ((Math.*random*() \* 3.999999999999999999999999999999999999999999)); //else choose random from 1 to 3

if (stonesSubtracted == 0) { //if it chooses random number 0

stonesSubtracted = 1; //just subtract 1 stone

}

}

stones = stones - stonesSubtracted; //subtract stones from total

}

if (stonesSubtracted == 1) { //ensure grammar is correct

System.*out*.println("The computer took " + stonesSubtracted + " stone!"); //singular for 1

} else {

System.*out*.println("The computer took " + stonesSubtracted + " stones!"); //plural for greater than 1

}

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

if (stones < 0) { //safeguard

stones = 0;

}

return stones;

}

//battleship, game where you must guess the positions of your opponenets ships before they guess yours

public static void Battleship() throws IOException, InterruptedException {

//all 2d arrays needed and arrays for information

System.*out*.println("<----------------------------------------------------------------------------------------------->\n" +

" 0000 0 00000 00000 0 00000 0000 0 0 00000 0000 o 0 \n" +

" 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0\n" +

" 0000 0 0 0 0 0 00000 000 00000 0 0000 0 0 0 000\n" +

" 0 0 00000 0 0 0 0 0 0 0 0 0 0 0 0 0 \n" +

" 0000 0 0 0 0 00000 00000 0000 0 0 00000 0 000 00000 \n" +

"<----------------------------------------------------------------------------------------------->\n");

TimeUnit.*SECONDS*.sleep(2);

System.*out*.println("Welcome to battleship! Subtracting 1 game point. Enjoy your game!");

System.*out*.println("Instructions:\n" +

"In this game of battleship, it is you against the computer! All of your ships can be placed however\n" +

"you want, and the difficulty of the computer is chosen by you as well! The difficulty number chosen\n" +

"for the computer is how many shots it gets each round, while your shots are limited to 1 per round.\n");

*SubtractGamePoints*();

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

Scanner input2 = new Scanner(System.*in*);

String[][] board = new String[11][11];

String[][] playerCompBoard = new String[11][11];

String[][] playerBoard = new String[12][13];

String[] alphabetLabel = {" A ", " B ", " C ", " D ", " E ", " F ", " G ", " H ", " I ", " J "};

String[] alphabetLabelAlpha = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "NULL"};

String[] numberLabel = {" 1 ", " 2 ", " 3 ", " 4 ", " 5 ", " 6 ", " 7 ", " 8 ", " 9 ", "10 "};

String[] ship = {" carrier (occupies 5 spaces) ", " battleship (occupies 4 spaces) ", " cruiser (occupies 3 " +

"spaces) ", " submarine (occupies 3 spaces) ", " destroyer (occupies 2 spaces) "};

//fills all 2d arrays with brackets and labels the axes

for (int i = 0; i <= 10; i++) {

for (int j = 0; j <= 10; j++) {

board[i][j] = "[ ]";

}

}

System.arraycopy(alphabetLabel, 0, board[0], 1, 10);

for (int i = 1; i <= 10; i++) {

board[i][0] = numberLabel[i - 1];

}

for (int i = 0; i <= 10; i++) {

for (int j = 0; j <= 10; j++) {

playerBoard[i][j] = "[ ]";

}

}

System.arraycopy(alphabetLabel, 0, playerBoard[0], 1, 10);

for (int i = 1; i <= 10; i++) {

playerBoard[i][0] = numberLabel[i - 1];

}

for (int i = 0; i <= 10; i++) {

for (int j = 0; j <= 10; j++) {

playerCompBoard[i][j] = "[ ]";

}

}

System.arraycopy(alphabetLabel, 0, playerCompBoard[0], 1, 10);

for (int i = 1; i <= 10; i++) {

playerCompBoard[i][0] = numberLabel[i - 1];

}

//prints out the players board

System.out.println("Your board:");

for (int i = 0; i <= 10; i++) {

System.out.print("| ");

for (int j = 0; j <= 10; j++) {

System.out.print(playerBoard[i][j] + " | ");

}

System.out.println();

}

//variables needed for ship placement for the player

String direction = null;

String xValue;

int counter = 0;

int yValue = 0;

int xValueConvert = 12;

boolean qRestart = true;

boolean playerPlacement = true;

boolean xRestart = true;

boolean yRestart = true;

while (playerPlacement) {

System.out.println("Please enter whether you would like your" + ship[counter] + "to be placed horizontally or " +

"vertically. (type h or v, -1 to exit)");

System.out.println("WARNING: Ships can overlap. It is your choice how you wan to make use of this information.\n" +

"However, please don't completely overlap a ship with another ship.");

while (qRestart) {

qRestart = false;

//takes input and repeats if input is wrong

direction = (String.valueOf(input.nextLine())).toLowerCase();

if (direction.equals("-1")) {

return;

} else if (!direction.equals("h") && !direction.equals("v")) {

System.out.println("Please enter h or v.");

qRestart = true;

}

}

//asks player to enter ship x and y values and places the ship into the players board vertically

//re asks the user if value entered is not valid

if (direction.equals("v")) {

{

while (xRestart) {

xRestart = false;

System.out.println("Please enter the x value you would like the ship to start at (letter value)");

xValue = (String.valueOf(input.nextLine())).toUpperCase();

for (int i = 0; i <= 9; i++) {

if (xValue.equals(alphabetLabelAlpha[i])) {

xValueConvert = i;

xRestart = false;

} else if (xValueConvert == 12) {

xRestart = true;

}

}

xValueConvert++;

}

while (yRestart) {

yValue = 0;

boolean indicateIncompatibleData = false;

yRestart = false;

System.out.println("Please enter the y value you would like the ship to start at (number value)");

String strYValue = input.nextLine();

try {

Integer.parseInt(strYValue);

} catch (Exception e) {

indicateIncompatibleData = true;

}

if (indicateIncompatibleData) {

System.out.println("Please enter a valid number that will allow the ship to fit in the grid.");

yRestart = true;

} else yValue = Integer.parseInt(strYValue);

//makes sure ship can fit in row entered, if not asks again

boolean indicateWrongEntry = false;

if (counter == 0) if ((yValue + 5) > 11) indicateWrongEntry = true;

if (counter == 1) if ((yValue + 4) > 11) indicateWrongEntry = true;

if (counter == 2 || counter == 3) if ((yValue + 3) > 11) indicateWrongEntry = true;

if (counter == 4) if ((yValue + 2) > 11) indicateWrongEntry = true;

if (indicateWrongEntry) {

System.out.println("Keep in mind the length of the ship!");

yRestart = true;

}

}

}

if (counter == 0) {

for (int i = 0; i < 5; i++) {

playerBoard[yValue++][xValueConvert] = "[C]";

}

} else if (counter == 1) {

for (int i = 0; i < 4; i++) {

playerBoard[yValue++][xValueConvert] = "[B]";

}

} else if (counter == 2) {

for (int i = 0; i < 3; i++) {

playerBoard[yValue++][xValueConvert] = "[R]";

}

} else if (counter == 3) {

for (int i = 0; i < 3; i++) {

playerBoard[yValue++][xValueConvert] = "[S]";

}

} else {

for (int i = 0; i < 2; i++) {

playerBoard[yValue++][xValueConvert] = "[D]";

}

}

}

//asks player to enter ship x and y values and places the ship into the players board horizontally

//re asks the user if value entered is not valid

if (direction.equals("h")) {

while (xRestart) {

xRestart = false;

System.out.println("Please enter the x value you would like the ship to start at (letter value)");

xValue = (String.valueOf(input.nextLine())).toUpperCase();

for (int i = 0; i <= 9; i++) {

if (xValue.equals(alphabetLabelAlpha[i])) {

xValueConvert = i;

xRestart = false;

} else if (xValueConvert > 11) {

xRestart = true;

}

}

//makes sure ship can fit in column entered, if not asks again

boolean indicateWrongEntry = false;

if (counter == 0) if ((xValueConvert + 5) > 11) indicateWrongEntry = true;

if (counter == 1) if ((xValueConvert + 4) > 11) indicateWrongEntry = true;

if (counter == 2 || counter == 3) if ((xValueConvert + 3) > 11) indicateWrongEntry = true;

if (counter == 4) if ((xValueConvert + 2) > 11) indicateWrongEntry = true;

if (indicateWrongEntry) {

System.out.println("Keep in mind the length of the ship!");

xRestart = true;

}

xValueConvert++;

}

while (yRestart) {

yValue = 0;

boolean indicateIncompatibleData = false;

yRestart = false;

System.out.println("Please enter the y value you would like the ship to start at (number value)");

String strYValue = input.nextLine();

try {

Integer.parseInt(strYValue);

} catch (Exception e) {

indicateIncompatibleData = true;

}

if (indicateIncompatibleData) {

System.out.println("Please enter a valid number that will allow the ship to fit in the grid.");

yRestart = true;

} else yValue = Integer.parseInt(strYValue);

if (yValue >= 11) {

System.out.println("Please pick a place within the boundaries");

yRestart = true;

}

}

if (counter == 0) {

for (int i = 0; i < 5; i++) {

playerBoard[yValue][xValueConvert++] = "[C]";

}

} else if (counter == 1) {

for (int i = 0; i < 4; i++) {

playerBoard[yValue][xValueConvert++] = "[B]";

}

} else if (counter == 2) {

for (int i = 0; i < 3; i++) {

playerBoard[yValue][xValueConvert++] = "[R]";

}

} else if (counter == 3) {

for (int i = 0; i < 3; i++) {

playerBoard[yValue][xValueConvert++] = "[S]";

}

} else {

for (int i = 0; i < 2; i++) {

playerBoard[yValue][xValueConvert++] = "[D]";

}

}

}

//prints out player board

System.out.println("Your board:");

for (int i = 0; i <= 10; i++) {

System.out.print("| ");

for (int j = 0; j <= 10; j++) {

System.out.print(playerBoard[i][j] + " | ");

}

System.out.println();

}

//restarts 5 times for each ship

counter++;

xValueConvert = 12;

xRestart = true;

yRestart = true;

qRestart = true;

direction = " ";

playerPlacement = counter != 5;

}

//requests player enters difficulty, this variable is used later in a fpr loop to determine how many times the computer shoots

int difficulty = 0;

boolean difficultyBoolean = true;

System.out.println("Enter the difficulty you would like to play at. (Difficulty entered is number of times computer shoots, from 1 to 10)");

while (difficultyBoolean) {

difficulty = input2.nextInt();

if (difficulty > 0 && difficulty < 10) {

difficultyBoolean = false;

} else {

System.out.println("Enter a value above 0, but less than 10.");

difficultyBoolean = true;

}

}

//calls ship placement method, which generates the computer's board randomly

ShipPlacement(board);

//checks how many of each ship is in the computer's board

int carNumberComp = 0;

int batNumberComp = 0;

int cruNumberComp = 0;

int subNumberComp = 0;

int desNumberComp = 0;

for (int i = 0; i <= 10; i++) {

for (int j = 0; j <= 10; j++) {

switch (board[i][j]) {

case "[C]" -> carNumberComp++;

case "[B]" -> batNumberComp++;

case "[R]" -> cruNumberComp++;

case "[S]" -> subNumberComp++;

case "[D]" -> desNumberComp++;

}

}

}

//checks how many of each ship is in the player's board

int carNumberPlayer = 0;

int batNumberPlayer = 0;

int cruNumberPlayer = 0;

int subNumberPlayer = 0;

int desNumberPlayer = 0;

for (int i = 0; i <= 10; i++) {

for (int j = 0; j <= 10; j++) {

switch (playerBoard[i][j]) {

case "[C]" -> carNumberPlayer++;

case "[B]" -> batNumberPlayer++;

case "[R]" -> cruNumberPlayer++;

case "[S]" -> subNumberPlayer++;

case "[D]" -> desNumberPlayer++;

}

}

}

BattleshipPlayerCompTurn(playerBoard, board, playerCompBoard, carNumberComp, batNumberComp, cruNumberComp, subNumberComp,

desNumberComp, difficulty, alphabetLabelAlpha, carNumberPlayer, batNumberPlayer, cruNumberPlayer,

subNumberPlayer, desNumberPlayer);

//reveals what the computer's board looked like and restarts the game

System.out.println("This was the computer's board:");

for (int i = 0; i <= 10; i++) {

System.out.print("| ");

for (int j = 0; j <= 10; j++) {

System.out.print(board[i][j] + " | ");

}

System.out.println();

}

System.out.println("Returning to menu...");

}

private static void BattleshipPlayerCompTurn(String[][] playerBoard, String[][] board, String[][] playerCompBoard,

int carNumberComp, int batNumberComp, int cruNumberComp, int subNumberComp,

int desNumberComp, int difficulty, String[] alphabetLabelAlpha, int carNumberPlayer,

int batNumberPlayer, int cruNumberPlayer, int subNumberPlayer, int desNumberPlayer) throws IOException {

Scanner input = new Scanner(System.in);

//variables needed for the players turn

String xValue;

int yValue = 0;

int xValueConvert = 12;

boolean move = true;

boolean playerGuess;

boolean compGuess;

boolean xRestartHit;

boolean yRestartHit;

boolean compX;

boolean compY;

int xCompGuess = 0;

int yCompGuess = 0;

int car = 0;

int bat = 0;

int cru = 0;

int sub = 0;

int des = 0;

int car2 = 0;

int bat2 = 0;

int cru2 = 0;

int sub2 = 0;

int des2 = 0;

int playerLose = 0;

int computerLose = 0;

//generates random number to determine who goes first

int coin = (int) (Math.random() \* 1.9999999999999999);

System.out.println("\*You flip a coin\*");

if (coin == 1) System.out.println("You go first.");

else System.out.println("The computer goes first.");

while (move) {

move = false;

if (coin == 1) {

playerGuess = true;

while (playerGuess) {

//shows what is known in computer's board, currently empty

System.out.println("Computer's board:");

for (int i = 0; i <= 10; i++) {

System.out.print("| ");

for (int j = 0; j <= 10; j++) {

System.out.print(playerCompBoard[i][j] + " | ");

}

System.out.println();

}

//tells player if they sunk a ship, records how many ships have been sunk

if (car == carNumberComp) {

System.out.println("You sunk the enemies carrier!");

computerLose++;

car = 0;

}

if (bat == batNumberComp) {

System.out.println("You sunk the enemies battleship!");

computerLose++;

bat = 0;

}

if (cru == cruNumberComp) {

System.out.println("You sunk the enemies cruiser!");

computerLose++;

cru = 0;

}

if (sub == subNumberComp) {

System.out.println("You sunk the enemies submarine!");

computerLose++;

sub = 0;

}

if (des == desNumberComp) {

System.out.println("You sunk the enemies destroyer!");

computerLose++;

des = 0;

}

xRestartHit = true;

yRestartHit = true;

//checks if someone has won or lost

if (computerLose == 5) {

xRestartHit = false;

yRestartHit = false;

System.out.println("You beat the computer on difficulty " + difficulty + "! Now try a harder difficulty.");

System.out.println("Writing new score to text file.");

ReadGamesFile();

battleshipUserScore++;

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(GamesTempFile));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (GamesTempFile.createNewFile()) {

System.out.println("New temporary games file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(battleshipUserScore);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!GamesFile.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!GamesTempFile.renameTo(GamesFile)) {

System.out.println("Could not rename temporary file.");

}

}

//takes x value and converts it into a usable number, same is done for every other time a letter is inputted

//takes x value, repeats if bad input

while (xRestartHit) {

System.out.println("It is your turn. Enter the x value of where you would like to hit. (Letter value)");

xRestartHit = false;

xValue = (String.valueOf(input.nextLine())).toUpperCase();

for (int i = 0; i <= 9; i++) {

if (xValue.equals(alphabetLabelAlpha[i])) {

xValueConvert = i + 1;

xRestartHit = false;

} else if (xValueConvert == 12) {

xRestartHit = true;

}

}

}

//takes y value, repeats if bad input

while (yRestartHit) {

yValue = 0;

boolean indicateIncompatibleData = false;

yRestartHit = false;

System.out.println("Now enter the y value of where you would like to hit. (Number value)");

String strYValue = input.nextLine();

try {

Integer.parseInt(strYValue);

} catch (Exception e) {

indicateIncompatibleData = true;

}

if (indicateIncompatibleData) {

System.out.println("Please enter a valid number.");

yRestartHit = true;

} else yValue = Integer.parseInt(strYValue);

}

//repeats if coordinate has been hit before, and records what happens if there is a hit

//tells user if they hit or miss, and records which ship has been hit for use later

switch (board[yValue][xValueConvert]) {

case "[X]", "[-]" -> {

playerGuess = true;

System.out.println("You've already entered that coordinate before.");

}

case "[C]" -> {

playerGuess = false;

playerCompBoard[yValue][xValueConvert] = "[X]";

board[yValue][xValueConvert] = "[X]";

System.out.println("You hit a ship!");

car++;

}

case "[B]" -> {

playerGuess = false;

playerCompBoard[yValue][xValueConvert] = "[X]";

board[yValue][xValueConvert] = "[X]";

System.out.println("You hit a ship!");

bat++;

}

case "[R]" -> {

playerGuess = false;

playerCompBoard[yValue][xValueConvert] = "[X]";

board[yValue][xValueConvert] = "[X]";

System.out.println("You hit a ship!");

cru++;

}

case "[S]" -> {

playerGuess = false;

playerCompBoard[yValue][xValueConvert] = "[X]";

board[yValue][xValueConvert] = "[X]";

System.out.println("You hit a ship!");

sub++;

}

case "[D]" -> {

playerGuess = false;

playerCompBoard[yValue][xValueConvert] = "[X]";

board[yValue][xValueConvert] = "[X]";

System.out.println("You hit a ship!");

des++;

}

default -> {

playerGuess = false;

playerCompBoard[yValue][xValueConvert] = "[-]";

board[yValue][xValueConvert] = "[-]";

System.out.println("You missed...");

}

}

//shows where you hit

System.out.println("Computer's board:");

for (int i = 0; i <= 10; i++) {

System.out.print("| ");

for (int j = 0; j <= 10; j++) {

System.out.print(playerCompBoard[i][j] + " | ");

}

System.out.println();

}

coin = 2;

move = true;

}

} else {

//computer generates a random x and y value, repeats if that number has been hit already

System.out.println("It is the computer's turn.");

//computer hits repeats depending on what what entered for difficulty

for (int i = 0; i < difficulty; i++) {

compGuess = true;

while (compGuess) {

compX = true;

while (compX) {

compX = false;

xCompGuess = (int) (Math.random() \* 10.999999999999999999999);

if (xCompGuess == 0) {

compX = true;

}

}

compY = true;

while (compY) {

compY = false;

yCompGuess = (int) (Math.random() \* 10.999999999999999999999);

if (yCompGuess == 0) {

compY = true;

}

}

//tells user what happened

switch (playerBoard[yCompGuess][xCompGuess]) {

case "[X]", "[-]" -> compGuess = true;

case "[C]" -> {

playerBoard[yCompGuess][xCompGuess] = "[X]";

System.out.println("The computer hit!");

compGuess = false;

car2++;

}

case "[B]" -> {

playerBoard[yCompGuess][xCompGuess] = "[X]";

System.out.println("The computer hit!");

compGuess = false;

bat2++;

}

case "[R]" -> {

playerBoard[yCompGuess][xCompGuess] = "[X]";

System.out.println("The computer hit!");

compGuess = false;

cru2++;

}

case "[S]" -> {

playerBoard[yCompGuess][xCompGuess] = "[X]";

System.out.println("The computer hit!");

compGuess = false;

sub2++;

}

case "[D]" -> {

playerBoard[yCompGuess][xCompGuess] = "[X]";

System.out.println("The computer hit!");

compGuess = false;

des2++;

}

default -> {

playerBoard[yCompGuess][xCompGuess] = "[-]";

System.out.println("The computer missed.");

compGuess = false;

}

}

}

}

//shows what computer has hit on your board

System.out.println("Your board:");

for (int i = 0; i <= 10; i++) {

System.out.print("| ");

for (int j = 0; j <= 10; j++) {

System.out.print(playerBoard[i][j] + " | ");

}

System.out.println();

}

coin = 1;

move = true;

}

//tells user if computer sunk their ship, records how many ships have been sunk

if (car2 == carNumberPlayer) {

System.out.println("Your carrier has been sunk!");

playerLose++;

car2 = 0;

}

if (bat2 == batNumberPlayer) {

System.out.println("Your battleship has been sunk!");

playerLose++;

bat2 = 0;

}

if (cru2 == cruNumberPlayer) {

System.out.println("Your cruiser has been sunk!");

playerLose++;

cru2 = 0;

}

if (sub2 == subNumberPlayer) {

System.out.println("Your submarine has been sunk!");

playerLose++;

sub2 = 0;

}

if (des2 == desNumberPlayer) {

System.out.println("Your destroyer has been sunk!");

playerLose++;

des2 = 0;

}

//if five ships have been sunk for either side, the other side wins

if (playerLose == 5) {

move = false;

System.out.println("You have lost! Better luck next time...");

System.out.println("Writing new score to text file.");

ReadGamesFile();

battleshipCompScore++;

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(GamesTempFile));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (GamesTempFile.createNewFile()) {

System.out.println("New temporary games file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(battleshipCompScore);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!GamesFile.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!GamesTempFile.renameTo(GamesFile)) {

System.out.println("Could not rename temporary file.");

}

System.out.println("New score successfully written");

}

if (computerLose == 5) {

move = false;

System.out.println("You beat the computer on difficulty " + difficulty + "! Now try a harder difficulty.");

System.out.println("Writing new score to text file.");

ReadGamesFile();

battleshipUserScore++;

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(GamesTempFile));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (GamesTempFile.createNewFile()) {

System.out.println("New temporary games file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(battleshipUserScore);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!GamesFile.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!GamesTempFile.renameTo(GamesFile)) {

System.out.println("Could not rename temporary file.");

}

System.out.println("New score successfully written");

}

}

}

private static void ShipPlacement(String[][] board) {

//creates random number to determine if ship is vertical or horizontal

int carrierDirection = (int) (Math.random() \* 1.9999999999999999999999999999999999);

if (carrierDirection == 1) { //if it is 1

//generate co ordinates

int carrierX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

int carrierY = (int) (Math.random() \* 5.9999999999999999999999999999999999);

//eliminate problem if random number is 0

if (carrierX == 0) {

carrierX = 1;

}

if (carrierY == 0) {

carrierY = 1;

}

//replace array elements with ship

for (int i = 0; i <= 4; i++) { //repeats 5 times to make ship 5 long

board[carrierX][carrierY + i] = "[C]";

}

} else {

//generate co ordinates

int carrierX = (int) (Math.random() \* 5.9999999999999999999999999999999999);

int carrierY = (int) (Math.random() \* 10.9999999999999999999999999999999999);

//eliminate problem if random number is 0

if (carrierX == 0) {

carrierX = 1;

}

if (carrierY == 0) {

carrierY = 1;

}

//replace array elements with ship

for (int i = 0; i <= 4; i++) { //repeats 5 times to make ship 5 long

board[carrierX + i][carrierY] = "[C]";

}

}

//the same is done for every other ship

int battleshipDirection = (int) (Math.random() \* 1.9999999999999999999999999999999999);

if (battleshipDirection == 1) {

boolean restartOverlap = false;

int battleshipX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

int battleshipY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

if (battleshipX == 0) {

battleshipX = 1;

}

if (battleshipY == 0) {

battleshipY = 1;

}

while (!board[battleshipX][battleshipY].equals("[ ]")) {

battleshipX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

battleshipY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 3; i++) {

if (!board[battleshipX][battleshipY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 3; i++) board[battleshipX][battleshipY + i] = "[B]";

} else {

boolean restartOverlap = false;

int battleshipX = (int) (Math.random() \* 6.9999999999999999999999999999999999);

int battleshipY = (int) (Math.random() \* 10.9999999999999999999999999999999999);

if (battleshipX == 0) {

battleshipX = 1;

}

if (battleshipY == 0) {

battleshipY = 1;

}

while (!board[battleshipX][battleshipY].equals("[ ]")) {

battleshipX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

battleshipY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 3; i++) {

if (!board[battleshipX][battleshipY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 2; i++) board[battleshipX + i][battleshipY] = "[B]";

}

//placement = true;

int cruiserDirection = (int) (Math.random() \* 1.9999999999999999999999999999999999);

if (cruiserDirection == 1) {

boolean restartOverlap = false;

int cruiserX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

int cruiserY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

if (cruiserX == 0) {

cruiserX = 1;

}

if (cruiserY == 0) {

cruiserY = 1;

}

while (!board[cruiserX][cruiserY].equals("[ ]")) {

cruiserX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

cruiserY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 2; i++) {

if (!board[cruiserX][cruiserY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 2; i++) board[cruiserX][cruiserY + i] = "[R]";

} else {

boolean restartOverlap = false;

int cruiserX = (int) (Math.random() \* 6.9999999999999999999999999999999999);

int cruiserY = (int) (Math.random() \* 10.9999999999999999999999999999999999);

if (cruiserX == 0) {

cruiserX = 1;

}

if (cruiserY == 0) {

cruiserY = 1;

}

while (!board[cruiserX][cruiserY].equals("[ ]")) {

cruiserX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

cruiserY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 2; i++) {

if (!board[cruiserX][cruiserY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 2; i++) board[cruiserX + i][cruiserY] = "[R]";

}

int submarineDirection = (int) (Math.random() \* 1.9999999999999999999999999999999999);

if (submarineDirection == 1) {

boolean restartOverlap = false;

int submarineX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

int submarineY = (int) (Math.random() \* 7.9999999999999999999999999999999999);

if (submarineX == 0) {

submarineX = 1;

}

if (submarineY == 0) {

submarineY = 1;

}

while (!board[submarineX][submarineY].equals("[ ]")) {

submarineX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

submarineY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 2; i++) {

if (!board[submarineX][submarineY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 2; i++) board[submarineX][submarineY + i] = "[S]";

} else {

boolean restartOverlap = false;

int submarineX = (int) (Math.random() \* 7.9999999999999999999999999999999999);

int submarineY = (int) (Math.random() \* 10.9999999999999999999999999999999999);

if (submarineX == 0) {

submarineX = 1;

}

if (submarineY == 0) {

submarineY = 1;

}

while (!board[submarineX][submarineY].equals("[ ]")) {

submarineX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

submarineY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 2; i++) {

if (!board[submarineX][submarineY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 2; i++) board[submarineX + i][submarineY] = "[S]";

}

int destroyerDirection = (int) (Math.random() \* 1.9999999999999999999999999999999999);

if (destroyerDirection == 1) {

boolean restartOverlap = false;

int destroyerX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

int destroyerY = (int) (Math.random() \* 8.9999999999999999999999999999999999);

if (destroyerX == 0) {

destroyerX = 1;

}

if (destroyerY == 0) {

destroyerY = 1;

}

while (!board[destroyerX][destroyerY].equals("[ ]")) {

destroyerX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

destroyerY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 1; i++) {

if (!board[destroyerX][destroyerY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 1; i++) board[destroyerX][destroyerY + i] = "[D]";

} else {

boolean restartOverlap = false;

int destroyerX = (int) (Math.random() \* 8.9999999999999999999999999999999999);

int destroyerY = (int) (Math.random() \* 10.9999999999999999999999999999999999);

if (destroyerX == 0) {

destroyerX = 1;

}

if (destroyerY == 0) {

destroyerY = 1;

}

while (!board[destroyerX][destroyerY].equals("[ ]")) {

destroyerX = (int) (Math.random() \* 10.9999999999999999999999999999999999);

destroyerY = (int) (Math.random() \* 6.9999999999999999999999999999999999);

}

for (int i = 0; i <= 1; i++) {

if (!board[destroyerX][destroyerY].equals("[ ]")) {

restartOverlap = true;

break;

}

}

if (!restartOverlap) for (int i = 0; i <= 1; i++) board[destroyerX + i][destroyerY] = "[D]";

}

}

private static final String[] board = new String[14]; //creates board, makes it a global variable

private static final Random randomDice = new Random();

private static int DiceRoll() {

return randomDice.nextInt(6) + 1; //returns a random value up to the number of sides of the dice

}

//yahtzee, game where you roll dice and put them into groupings, to try and get the highest points possible

public static void Yahtzee() throws InterruptedException {

Scanner input = new Scanner(System.in);

//prints introduction, rules, choices, etc.

System.out.println("<------------------------------------------------------------------------------------->\n" +

" 0 0 0 0 0 00000 00000 00000 00000 0 \n" +

" 0 0 0 0 0 0 0 0 0 0 0 0 \n" +

" 0 0 0 00000 0 0 00000 00000 0 0 0 \n" +

" 0 00000 0 0 0 0 0 0 0 0 0 0\n" +

" 0 0 0 0 0 0 00000 00000 00000 0 0 \n" +

"<------------------------------------------------------------------------------------->\n");

TimeUnit.SECONDS.sleep(2);

System.out.println("Welcome to Yahtzee! Here are the rules: \n" +

"On your turn, you are given three rolls to score the highest combination you can in a category. \n" +

"In each roll, you can choose to re-roll however many dice you want. After your roles are finished, \n" +

"you must choose a category from one of the 13 to put your score or a zero into " +

"(for if no categories are satisfied). \n" +

"The game ends when all categories have been filled, and you can only enter one roll set for each category. \n" +

"\nThe thirteen categories consist of the upper section and lower section, where the upper section \n" +

"consists of the sum of all same numbers in a given roll (ones, twos, threes, fours, fives, and sixes). \n" +

"If you score 63 points or higher in the upper section, a 35 point bonus is rewarded \n" +

"And the lower section consists of: \n" +

"-The sum of all die on the condition that there is a three of four of a kind, \n" +

"-A full house (two of a kind plus a three of a kind, 25 points) \n" +

"-A small straight (any sequence of four numbers, one random, 30 points) \n" +

"-A large straight (complete sequence of numbers, 40 points) \n" +

"-A YAHTZEE (five of a kind, 50 points, 100 if YAHTZEE already gotten) \n" +

"-Chance (sum of all five dice) \n" +

"Note: Getting a second YAHTZEE does not count towards your 13 rolls, only the first does." +

"AND REMEMBER TO CHOOSE RE-ROLLS CAREFULLY, THERE IS NO REDOING! \n");

for (int i = 0; i <= 13; i++) { //fills the board and makes it empty

board[i] = "Empty";

}

System.out.println("Please select what you would like to do. :D \n" +

"1. Single player (Play the game by yourself and see if you can beat the high-score!) \n" +

"2. Practice (Way to practice your rolls and test different strategies! (Doesn't score rolls)) \n" +

"3. Exit the program");

int choice;

boolean error;

do {

try {

choice = input.nextInt();

if (choice == 1) {

if (GetGamePoints() == 0) {

System.out.println("You don't have enough points to play.");

return;

} else {

SubtractGamePoints();

SubtractGamePoints();

}

YahtzeeSinglePlayerMethod(); //calls singlePlayerMethod method

System.out.println("Returning you to menu.");

return;

} else if (choice == 2) {

YahtzeePractice(); //calls practice method

error = false;

} else if (choice == 3) {

return;

} else {

System.out.println("Please enter a valid number. -.-");

error = true; //error guard

}

} catch (Exception e) { //if any error then tell user to input correct value

System.out.println("Please enter a valid number. -.-");

error = true;

input.next();

}

} while (error); //repeats code if error is detected

}

//singlePlayerMethod manages entire game for single player

private static void YahtzeeSinglePlayerMethod() throws IOException {

Scanner input = new Scanner(System.in);

System.out.println("Welcome to single player! Here is your game card!");

System.out.println("UPPER HALF: \n" +

"1. Ones: " + board[0] + "\n" +

"2. Twos: " + board[1] + "\n" +

"3. Threes: " + board[2] + "\n" +

"4. Fours: " + board[3] + "\n" +

"5. Fives: " + board[4] + "\n" +

"6. Sixes: " + board[5] + "\n" +

"LOWER HALF: \n" +

"7. Three of a kind: " + board[6] + "\n" +

"8. Four of a kind: " + board[7] + "\n" +

"9. Full House: " + board[8] + "\n" +

"10. Small Straight: " + board[9] + "\n" +

"11. Large Straight: " + board[10] + "\n" +

"12. Yahtzee: " + board[11] + "\n" +

"13. Chance: " + board[12] + "\n");

int rollRepeat = 0;

while (rollRepeat <= 12) { //repeats for the 13 rolls needed to finish the game

int dice1Result = DiceRoll();

int dice2Result = DiceRoll();

int dice3Result = DiceRoll();

int dice4Result = DiceRoll();

int dice5Result = DiceRoll();

String choice;

boolean error;

String rollNumber = null;

switch (rollRepeat) { //changes number depending on what roll number

case 0 -> rollNumber = "first";

case 1 -> rollNumber = "second";

case 2 -> rollNumber = "third";

case 3 -> rollNumber = "fourth";

case 4 -> rollNumber = "fifth";

case 5 -> rollNumber = "sixth";

case 6 -> rollNumber = "seventh";

case 7 -> rollNumber = "eighth";

case 8 -> rollNumber = "ninth";

case 9 -> rollNumber = "tenth";

case 10 -> rollNumber = "eleventh";

case 11 -> rollNumber = "twelfth";

case 12 -> rollNumber = "thirteenth";

}

for (int i = 0; i <= 1; i++) { //prints results of the dice roll

System.out.println("Here are the results of your " + rollNumber + " roll:");

System.out.println("Dice 1: " + dice1Result + "\n" + "Dice 2: " + dice2Result + "\n" + "Dice 3: " + dice3Result

+ "\n" + "Dice 4: " + dice4Result + "\n" + "Dice 5: " + dice5Result);

System.out.println("Do you want to re-roll dice 1? (type yes or no) \n" +

"If you would like to re-roll none, enter 0. If you would like to exit the game, type \"exit\")");

do { //asks user if they want to re-roll, skip rolls, or exit, repeats for all dice

error = false;

choice = input.nextLine();

try {

if (choice.equalsIgnoreCase("Yes")) {

dice1Result = DiceRoll();

} else if (choice.equalsIgnoreCase("0")) {

i = 1;

} else if (choice.equalsIgnoreCase("Exit")) {

i = 1;

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please remember to enter a valid value.");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

if (choice.equalsIgnoreCase("0")) {

System.out.println("Rolls skipped.");

} else if (choice.equalsIgnoreCase("Exit")) {

System.out.println("Exiting this game. :c");

return;

} else {

System.out.println("Do you want to re-roll dice 2? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice2Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

System.out.println("Do you want to re-roll dice 3? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice3Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

System.out.println("Do you want to re-roll dice 4? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice4Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

System.out.println("Do you want to re-roll dice 5? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice5Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

}

}

System.out.println("Your final roll was " + dice1Result + ", " + dice2Result + ", " + dice3Result + ", " +

dice4Result + ", " + dice5Result);

System.out.println("Here is your card."); //prints player's card, requests for category

System.out.println("UPPER HALF: \n" +

"1. Ones: " + board[0] + "\n" +

"2. Twos: " + board[1] + "\n" +

"3. Threes: " + board[2] + "\n" +

"4. Fours: " + board[3] + "\n" +

"5. Fives: " + board[4] + "\n" +

"6. Sixes: " + board[5] + "\n" +

"LOWER HALF: \n" +

"7. Three of a kind: " + board[6] + "\n" +

"8. Four of a kind: " + board[7] + "\n" +

"9. Full House: " + board[8] + "\n" +

"10. Small Straight: " + board[9] + "\n" +

"11. Large Straight: " + board[10] + "\n" +

"12. YAHTZEE: " + board[11] + "\n" +

"13. Chance: " + board[12] + "\n" +

"14. YAHTZEE bonus: " + board[13] + "\n");

System.out.println("Please select a category to score into.");

do { //checks user input, ensures they are being honest and the input is correct

error = false;

try {

int choiceCard = input.nextInt();

if (choiceCard >= 1 && choiceCard <= 14) {

if (choiceCard == 1) {

if (board[0].equals("Empty")) {

int onesSum = 0;

if (dice1Result == 1) {

onesSum += 1;

}

if (dice2Result == 1) {

onesSum += 1;

}

if (dice3Result == 1) {

onesSum += 1;

}

if (dice4Result == 1) {

onesSum += 1;

}

if (dice5Result == 1) {

onesSum += 1;

}

board[0] = String.valueOf(onesSum);

System.out.println(onesSum + " has been entered in square 1.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 1 has already been filled.");

error = true;

}

}

if (choiceCard == 2) {

if (board[1].equals("Empty")) {

int twosSum = 0;

if (dice1Result == 2) {

twosSum += 2;

}

if (dice2Result == 2) {

twosSum += 2;

}

if (dice3Result == 2) {

twosSum += 2;

}

if (dice4Result == 2) {

twosSum += 2;

}

if (dice5Result == 2) {

twosSum += 2;

}

board[1] = String.valueOf(twosSum);

System.out.println(twosSum + " has been entered in square 2.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 2 has already been filled.");

error = true;

}

}

if (choiceCard == 3) {

if (board[2].equals("Empty")) {

int threesSum = 0;

if (dice1Result == 3) {

threesSum += 3;

}

if (dice2Result == 3) {

threesSum += 3;

}

if (dice3Result == 3) {

threesSum += 3;

}

if (dice4Result == 3) {

threesSum += 3;

}

if (dice5Result == 3) {

threesSum += 3;

}

board[2] = String.valueOf(threesSum);

System.out.println(threesSum + " has been entered in square 3.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 3 has already been filled.");

error = true;

}

}

if (choiceCard == 4) {

if (board[3].equals("Empty")) {

int foursSum = 0;

if (dice1Result == 4) {

foursSum += 4;

}

if (dice2Result == 4) {

foursSum += 4;

}

if (dice3Result == 4) {

foursSum += 4;

}

if (dice4Result == 4) {

foursSum += 4;

}

if (dice5Result == 4) {

foursSum += 4;

}

board[3] = String.valueOf(foursSum);

System.out.println(foursSum + " has been entered in square 4.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 4 has already been filled.");

error = true;

}

}

if (choiceCard == 5) {

if (board[4].equals("Empty")) {

int fivesSum = 0;

if (dice1Result == 5) {

fivesSum += 5;

}

if (dice2Result == 5) {

fivesSum += 5;

}

if (dice3Result == 5) {

fivesSum += 5;

}

if (dice4Result == 5) {

fivesSum += 5;

}

if (dice5Result == 5) {

fivesSum += 5;

}

board[4] = String.valueOf(fivesSum);

System.out.println(fivesSum + " has been entered in square 5.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 5 has already been filled.");

error = true;

}

}

if (choiceCard == 6) {

if (board[5].equals("Empty")) {

int sixesSum = 0;

if (dice1Result == 6) {

sixesSum += 6;

}

if (dice2Result == 6) {

sixesSum += 6;

}

if (dice3Result == 6) {

sixesSum += 6;

}

if (dice4Result == 6) {

sixesSum += 6;

}

if (dice5Result == 6) {

sixesSum += 6;

}

board[5] = String.valueOf(sixesSum);

System.out.println(sixesSum + " has been entered in square 6.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 6 has already been filled.");

error = true;

}

}

//creates a list of the die results, and sorts it with bubble sort

int[] dieList = {dice1Result, dice2Result, dice3Result, dice4Result, dice5Result};

for (int i = 0; i < dieList.length; i++) {

for (int j = 1; j < (dieList.length - i); j++) {

if (dieList[j - 1] > dieList[j]) {

int placeholder = dieList[j - 1];

dieList[j - 1] = dieList[j];

dieList[j] = placeholder;

}

}

}

String diceSum = String.valueOf(dice1Result + dice2Result + dice3Result + dice4Result + dice5Result);

if (choiceCard == 7) { //only runs code if three die are the same, else user chooses again

if (board[6].equals("Empty")) {

if (((dieList[0] == dieList[1]) && (dieList[1] == dieList[2])) || ((dieList[1] == dieList[2])

&& (dieList[2] == dieList[3])) || ((dieList[2] == dieList[3]) && (dieList[3] == dieList[4]))) {

System.out.println("Three of a kind!");

board[6] = diceSum;

System.out.println(diceSum + " has been entered in square 7.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. You don't seem to have a three of a kind. :(");

System.out.println("Would you like to enter a 0?");

boolean zeroRepeat;

do {

choice = input.nextLine();

zeroRepeat = false;

if (choice.equalsIgnoreCase("Yes")) {

board[6] = "0";

System.out.println("0 has been entered in square 7.");

rollRepeat++;

} else if (choice.equalsIgnoreCase("No")) {

System.out.println("Then please choose another category.");

error = true;

} else {

System.out.println("Please enter yes or no. >:(");

zeroRepeat = true;

}

} while (zeroRepeat);

}

} else {

System.out.println("Uh oh. Square 7 has already been filled.");

error = true;

}

}

if (choiceCard == 8) { //only runs if four die are the same

if (board[7].equals("Empty")) {

if (((dieList[0] == dieList[1]) && (dieList[1] == dieList[2]) && (dieList[2] == dieList[3]))

|| ((dieList[1] == dieList[2]) && (dieList[2] == dieList[3]) && (dieList[3] == dieList[4]))) {

System.out.println("Four of a kind!");

board[7] = diceSum;

System.out.println(diceSum + " has been entered in square 8.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. You don't seem to have a four of a kind. :(");

System.out.println("Would you like to enter a 0?");

boolean zeroRepeat;

do {

choice = input.nextLine();

zeroRepeat = false;

if (choice.equalsIgnoreCase("Yes")) {

board[7] = "0";

System.out.println("0 has been entered in square 8.");

rollRepeat++;

} else if (choice.equalsIgnoreCase("No")) {

System.out.println("Then please choose another category.");

error = true;

} else {

System.out.println("Please enter yes or no. >:(");

zeroRepeat = true;

}

} while (zeroRepeat);

}

} else {

System.out.println("Uh oh. Square 8 has already been filled.");

error = true;

}

}

if (choiceCard == 9) { //only runs if there is a pair of two, and a pair of three

if (board[8].equals("Empty")) {

if (((dieList[0] == dieList[1]) && (dieList[1] == dieList[2]) && (dieList[3] == dieList[4]))

|| ((dieList[0] == dieList[1]) && (dieList[2] == dieList[3]) && (dieList[3] == dieList[4]))) {

System.out.println("Full house!!");

board[8] = "25";

System.out.println("25 has been entered in square 9.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. You don't seem to have a full house. :(");

System.out.println("Would you like to enter a 0?");

boolean zeroRepeat;

do {

choice = input.nextLine();

zeroRepeat = false;

if (choice.equalsIgnoreCase("Yes")) {

board[8] = "0";

System.out.println("0 has been entered in square 9.");

rollRepeat++;

} else if (choice.equalsIgnoreCase("No")) {

System.out.println("Then please choose another category.");

error = true;

} else {

System.out.println("Please enter yes or no. >:(");

zeroRepeat = true;

}

} while (zeroRepeat);

}

} else {

System.out.println("Uh oh. Square 9 has already been filled.");

error = true;

}

}

//system that reads sorted list and finds the smallest values to remove ay repeats

if (choiceCard == 10) {

if (board[9].equals("Empty")) {

int smallestDice = dieList[0];

int secondSmallestDice = dieList[1];

int thirdSmallestDice;

int fourthSmallestDice;

int fifthSmallestDice = dieList[4];

if (secondSmallestDice == smallestDice) {

secondSmallestDice = dieList[2];

thirdSmallestDice = dieList[3];

fourthSmallestDice = dieList[4];

} else {

thirdSmallestDice = dieList[2];

if (thirdSmallestDice == secondSmallestDice) {

thirdSmallestDice = dieList[3];

fourthSmallestDice = dieList[4];

} else {

fourthSmallestDice = dieList[3];

if (fourthSmallestDice == thirdSmallestDice) {

fourthSmallestDice = dieList[4];

}

}

}

//only runs if there are four die that have values in succession

if (((secondSmallestDice == smallestDice + 1) && (thirdSmallestDice == smallestDice + 2) &&

(fourthSmallestDice == smallestDice + 3)) || ((thirdSmallestDice == secondSmallestDice + 1)

&& (fourthSmallestDice == secondSmallestDice + 2) && (fifthSmallestDice == secondSmallestDice + 3))) {

System.out.println("Small straight!!!");

System.out.println("30 has been entered in square 10.");

board[9] = "30";

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. You don't seem to small straight. :(");

System.out.println("Would you like to enter a 0?");

boolean zeroRepeat;

do {

choice = input.nextLine();

zeroRepeat = false;

if (choice.equalsIgnoreCase("Yes")) {

board[9] = "0";

System.out.println("0 has been entered in square 10.");

rollRepeat++;

} else if (choice.equalsIgnoreCase("No")) {

System.out.println("Then please choose another category.");

error = true;

} else {

System.out.println("Please enter yes or no. >:(");

zeroRepeat = true;

}

} while (zeroRepeat);

}

} else {

System.out.println("Uh oh. Square 10 has already been filled.");

error = true;

}

}

if (choiceCard == 11) { //only runs if there is a complete sequence of 5

if (board[10].equals("Empty")) {

if (((dieList[0] == 1) && (dieList[1] == 2) && (dieList[2] == 3) && ((dieList[3] == 4)) &&

(dieList[4] == 5)) || ((dieList[0] == 2) && (dieList[1] == 3) && (dieList[2] == 4)

&& ((dieList[3] == 5)) && (dieList[4] == 6))) {

System.out.println("Large straight!!!!!");

System.out.println("40 has been entered in square 12.");

board[10] = "40";

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. You don't seem to large straight. :(");

System.out.println("Would you like to enter a 0?");

boolean zeroRepeat;

do {

choice = input.nextLine();

zeroRepeat = false;

if (choice.equalsIgnoreCase("Yes")) {

board[10] = "0";

System.out.println("0 has been entered in square 11.");

rollRepeat++;

} else if (choice.equalsIgnoreCase("No")) {

System.out.println("Then please choose another category.");

error = true;

} else {

System.out.println("Please enter yes or no. >:(");

zeroRepeat = true;

}

} while (zeroRepeat);

}

} else {

System.out.println("Uh oh. Square 11 has already been filled.");

error = true;

}

}

if (choiceCard == 12) { //only runs if all five die are the same

if (board[11].equals("Empty")) {

if ((dieList[0] == dieList[1]) && (dieList[1] == dieList[2]) && (dieList[2] == dieList[3])

&& ((dieList[3] == dieList[4]))) {

System.out.println("YAHTZEE!!!!!!!!!!");

System.out.println("50 has been entered in square 12.");

board[11] = "50";

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. You don't seem to YAHTZEE. :(");

System.out.println("Would you like to enter a 0?");

boolean zeroRepeat;

do {

choice = input.nextLine();

zeroRepeat = false;

if (choice.equalsIgnoreCase("Yes")) {

board[11] = "0";

System.out.println("0 has been entered in square 12.");

rollRepeat++;

} else if (choice.equalsIgnoreCase("No")) {

System.out.println("Then please choose another category.");

error = true;

} else {

System.out.println("Please enter yes or no. >:(");

zeroRepeat = true;

}

} while (zeroRepeat);

}

} else {

System.out.println("Uh oh. Square 12 has already been filled.");

error = true;

}

}

if (choiceCard == 13) { //takes sum of all die

if (board[12].equals("Empty")) {

board[12] = diceSum;

System.out.println(diceSum + " has been entered in square 13.");

rollRepeat++;

error = false;

} else {

System.out.println("Uh oh. Square 13 has already been filled.");

error = true;

}

}

if (choiceCard == 14) { //only runs if yahtzee has already been gotten, or already bonus yahtzee

if (board[13].equals("100")) {

board[13] = "200";

} else if (board[11].equals("50")) {

System.out.println("YAHTZEE BONUS!!!!!!!!!!!!!!!!!!!!!!!!!!");

System.out.println("100 has been entered in square 14.");

board[13] = "100";

error = false;

} else {

System.out.println("Oh dear. You haven't gotten a YAHTZEE yet...");

error = true;

}

}

} else {

System.out.println("Please enter a number on the board. :(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a number on the board. ;(");

input.next();

error = true;

}

} while (error);

System.out.println("Here is your new card."); //prints new card with new values

System.out.println("UPPER HALF: \n" +

"1. Ones: " + board[0] + "\n" +

"2. Twos: " + board[1] + "\n" +

"3. Threes: " + board[2] + "\n" +

"4. Fours: " + board[3] + "\n" +

"5. Fives: " + board[4] + "\n" +

"6. Sixes: " + board[5] + "\n" +

"LOWER HALF: \n" +

"7. Three of a kind: " + board[6] + "\n" +

"8. Four of a kind: " + board[7] + "\n" +

"9. Full House: " + board[8] + "\n" +

"10. Small Straight: " + board[9] + "\n" +

"11. Large Straight: " + board[10] + "\n" +

"12. YAHTZEE: " + board[11] + "\n" +

"13. Chance: " + board[12] + "\n" +

"14. YAHTZEE bonus: " + board[13] + "\n");

}

if (board[13].equals("Empty")) { //if no bonus yahtzee, place zero to allow for parse to int

board[13] = "0";

}

System.out.println("You've filled all squares! Calculating your score now!");

System.out.println("Here was your final card.");

System.out.println("UPPER HALF: \n" +

"1. Ones: " + board[0] + "\n" +

"2. Twos: " + board[1] + "\n" +

"3. Threes: " + board[2] + "\n" +

"4. Fours: " + board[3] + "\n" +

"5. Fives: " + board[4] + "\n" +

"6. Sixes: " + board[5] + "\n" +

"LOWER HALF: \n" +

"7. Three of a kind: " + board[6] + "\n" +

"8. Four of a kind: " + board[7] + "\n" +

"9. Full House: " + board[8] + "\n" +

"10. Small Straight: " + board[9] + "\n" +

"11. Large Straight: " + board[10] + "\n" +

"12. YAHTZEE: " + board[11] + "\n" +

"13. Chance: " + board[12] + "\n" +

"14. YAHTZEE bonus: " + board[13] + "\n");

int score = 0;

int[] intBoard = new int[14];

for (int i = 0; i < board.length; i++) {

intBoard[i] = Integer.parseInt(board[i]);

} //parses all values in board, makes them into integers

for (int i : intBoard) {

score += i;

} //adds together all values in the board

if ((intBoard[0] + intBoard[1] + intBoard[2] + intBoard[3] + intBoard[4] + intBoard[5]) >= 63) {

System.out.println("However, You earned at least 63 points in the upper half! You got a 32 point bonus. " +

"Good job. :)");

score += 32;

}

System.out.println("Your final score was " + score + "!");

System.out.println("That's very neat!");

ReadGamesFile();

System.out.println("The current high score is " + yahtzeeHighScore + ".");

if (score > yahtzeeHighScore) {

System.out.println("Which means you beat the high score! Congrats!");

System.out.println("Writing new score to text file.");

ReadGamesFile();

battleshipUserScore++;

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(GamesFile));

PrintWriter gamesPw = new PrintWriter(gamesBw);

gamesPw.println();

//creates temporary games file

try {

if (GamesTempFile.createNewFile()) {

System.out.println("New temporary games file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(battleshipUserScore);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!GamesFile.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!GamesTempFile.renameTo(GamesFile)) {

System.out.println("Could not rename temporary file.");

}

System.out.println("New score successfully written");

} else {

System.out.println("Which means you did not beat the high score. Better luck next time!");

}

}

//practice method, runs rolls for user to practice re-rolls

private static void YahtzeePractice() {

Scanner input = new Scanner(System.in);

int dice1Result = DiceRoll();

int dice2Result = DiceRoll();

int dice3Result = DiceRoll();

int dice4Result = DiceRoll();

int dice5Result = DiceRoll();

boolean error = true;

String choice;

boolean practice = true;

System.out.println("Hello! Welcome to the practice room!");

System.out.println("Do you want to practice? (Please enter yes or no)");

while (practice) {

do {

String practiceChoice;

practiceChoice = input.nextLine();

//interface for navigation

if (practiceChoice.equalsIgnoreCase("Yes")) { //if user enters yes, practice, else exit the method

for (int i = 0; i <= 1; i++) {

System.out.println("Neat! Here are the results of your practice roll:");

System.out.println("Dice 1: " + dice1Result + "\n" + "Dice 2: " + dice2Result + "\n" + "Dice 3: " + dice3Result

+ "\n" + "Dice 4: " + dice4Result + "\n" + "Dice 5: " + dice5Result);

System.out.println("Do you want to re-roll dice 1? (type yes or no) \n" +

"If you would like to re-roll none, enter 0.");

//runs dice rolls and re-rolls

do {

error = false;

choice = input.nextLine();

try {

if (choice.equalsIgnoreCase("Yes")) {

dice1Result = DiceRoll();

} else if (choice.equalsIgnoreCase("0")) {

i = 1;

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please remember to enter a valid value.");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

if (choice.equalsIgnoreCase("0")) {

System.out.println("Rolls skipped.");

} else {

System.out.println("Do you want to re-roll dice 2? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice2Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

System.out.println("Do you want to re-roll dice 3? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice3Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

System.out.println("Do you want to re-roll dice 4? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice4Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

System.out.println("Do you want to re-roll dice 5? (type yes or no)");

do {

choice = input.nextLine();

error = false;

try {

if (choice.equalsIgnoreCase("Yes")) {

dice5Result = DiceRoll();

} else if (choice.equalsIgnoreCase("No")) {

break;

} else {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a valid value. >:(");

error = true;

}

} while (error);

}

}

//prints final rolls, asks if user wants to go again

System.out.println("Your final roll was " + dice1Result + ", " + dice2Result + ", " + dice3Result + ", " +

dice4Result + ", " + dice5Result);

System.out.println("Do you want to practice again?");

} else if (practiceChoice.equalsIgnoreCase("No")) {

System.out.println("Oh that's too bad. Sending you back to the lobby. See you later. :(");

practice = false;

error = false;

} else {

System.out.println("Please enter a proper value! -.-");

error = true;

}

} while (error);

}

}

//bowling, game where you play a game of bowling with calculated trajectory and speed

public static void Bowling() throws InterruptedException {

Scanner input = new Scanner(System.in);

System.out.println("<-------------------------------------------------------------------------------------------------->\n" +

" 0000 000 0 0 0 00000 00 0 000 0 0 0 \n" +

" 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 \n" +

" 0000 0 0 0 0 0 0 0 0 0 0 0 00 000 000 000 000--- - \n" +

" 0 0 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 0--- -\n" +

" 0000 000 0 0 00000 00000 0 0 000 000 000 000 000--- - \n" +

"<-------------------------------------------------------------------------------------------------->\n");

TimeUnit.SECONDS.sleep(2);

BowlingInstructionsMethod(); //prints instructions

System.out.println("\nPlease enter how you would like to play: \n" +

"1. Solo mode 2. Practice mode 3. Instructions 4. Exit Back to Menu");

int choice;

boolean error = true;

//loop asks for user inputs, what they would like to do

do {

try {

choice = input.nextInt();

if (choice == 1) {

if (GetGamePoints() >= 3) {

System.out.println("Subtracting three points.");

SubtractGamePoints();

SubtractGamePoints();

SubtractGamePoints();

} else {

System.out.println("You don't have enough points to play.");

return;

}

BowlingPlayerMethod();

} else if (choice == 2) {

error = false;

BowlingPracticeMode();

} else if (choice == 3) {

error = false;

BowlingInstructionsMethod();

} else if (choice == 4) {

System.out.println("Have a good day! Goodbye! :D");

return;

}

} catch (Exception e) { //if any error then tell user to input correct value

System.out.println("Please enter a valid number. -.-");

error = true;

input.next();

}

} while (error); //repeats code if error is detected

}

//instruction method, prints the instructions when called, returns nothing

private static void BowlingInstructionsMethod() {

System.out.println("Welcome to bowling simulator. Here we have bowling, and that's pretty much it. :D \n" +

"In this bowling simulator, the way you choose to bowl actually matters. So play carefully. \n" +

"Here are the rules for standard bowling (this simulator is the same!) \n" +

"You are awarded 10 points for a strike (plus the ones you get in the next frame), \n" +

"10 points for a spare, and if you don't get a spare or a strike you get 1 point per pin knocked over. \n" +

"Rules are standard, meaning that if you get a strike on your first throw, you get another two after. \n" +

"\nInstructions for how to play: \n" +

"You will have choices for where you want to throw from: Far left, left, middle, right, and far right \n" +

"The board that you start with looks like this: \n" +

"[7] [8] [9] [10] - Throwing in the middle increases chances of a strike, but also for a split (strikes 1)\n" +

" [4] [5] [6] - Throwing left or right wipes down a side generally, with a high chance of hitting 2,4 or 3,6, respectively \n" +

" [2] [3] - Throwing far left or far right increases chances of clearing splits, but doesn't do well for front pins\n" +

" [1] (far left and far right have a high chance of hitting the 4,7 and 6,10 pins, respectively)\n" +

"On top of all this, speed that you throw the ball can also be chosen, from 5 mph to 30 mph. \n" +

"Throwing too slow will not knock down many pins, but too fast will not allow for pins to cascade \n" +

"It is up to you to find the optimal speed, and to strategically choose the right shots get strikes and spares. \n" +

"Good luck and have fun!");

}

//practice mode method, runs twoThrows in a loop until terminated

private static void BowlingPracticeMode() {

int frame = 1;

int score = 0;

boolean repeat = true;

System.out.println("Hello! Welcome to practice bowling mode! Here, you can practice how to play or use this \n" +

"to learn optimal speeds and hone your skills! You can exit anytime by entering option 6 when choosing throw type.\n" +

"Let's begin!");

//repeats until terminated, infinite attempts to practice

while (repeat) {

while (frame <= 1) {

score += TwoThrows();

if (score >= 1000) {

repeat = false;

break;

}

frame++;

System.out.println("Your final points for this frame was " + score + ".");

}

}

}

//playerMethod, handles all instances where user wants to play games

private static void BowlingPlayerMethod() throws IOException {

int p1Score = 0;

int frame = 1;

//loop to break to exit method

System.out.println("Hey " + User.GetUsername() + "! It's your turn!");

System.out.println("Let's start frame one.");

while (frame <= 10) { //plays for 10 frames

p1Score += TwoThrows();

if (p1Score >= 1000) { //exit if return value is greater than 1000 (if exit entered while playing)

break;

}

System.out.println("Good job " + User.GetUsername() + ", your total points are now " + p1Score);

frame++;

System.out.println("Time for your next turn! It is now frame " + frame + ".");

}

System.out.println("Your final score was " + p1Score + "!");

System.out.println("The current high score is " + bowlingHighScore);

if (bowlingHighScore < p1Score) {

System.out.println("Which means you beat the high score!!!");

System.out.println("Writing new score to text file.");

Scanner gamesFileReader = new Scanner(new FileInputStream("Games.txt"));

BufferedWriter gamesBw = new BufferedWriter(new FileWriter(GamesTempFile));

PrintWriter gamesPw = new PrintWriter(gamesBw);

//creates temporary games file

try {

if (GamesTempFile.createNewFile()) {

System.out.println("New temporary games file created.");

}

} catch (Exception e) {

System.out.println("An error occurred while creating a temporary games file.");

}

gamesPw.println(gamesFileReader.nextLine());

gamesPw.println(gamesFileReader.nextLine());

gamesFileReader.nextLine();

gamesPw.println(nimUserScore);

while (gamesFileReader.hasNext()) {

gamesPw.println(gamesFileReader.nextLine());

}

gamesFileReader.close();

gamesBw.close();

gamesPw.close();

//deletes original games file with assignment we want to delete

if (!GamesFile.delete()) {

System.out.println("Could not delete original assignment file.");

}

//renames the file with the deleted file removed to the original file

if (!GamesTempFile.renameTo(GamesFile)) {

System.out.println("Could not rename temporary file.");

}

System.out.println("New score successfully written");

} else {

System.out.println("You did not get a new high score... Better luck next time!");

}

}

private static String pin1 = "[0]";

private static String pin2 = "[0]";

private static String pin3 = "[0]";

private static String pin4 = "[0]";

private static String pin5 = "[0]";

private static String pin6 = "[0]";

private static String pin7 = "[0]";

private static String pin8 = "[0]";

private static String pin9 = "[0]";

private static String pin10 = "[0]";

//get name method, returns players name

//two throws method, handles throws for every player

private static int TwoThrows() {

Scanner input = new Scanner(System.in);

int turn = 0;

int score = 0;

boolean strikeTurnOne = true;

boolean turnAgain = true;

pin1 = "[0]";

pin2 = "[0]";

pin3 = "[0]";

pin4 = "[0]";

pin5 = "[0]";

pin6 = "[0]";

pin7 = "[0]";

pin8 = "[0]";

pin9 = "[0]";

pin10 = "[0]";

System.out.println("Here's the state of the current lane.");

BoardPrint();

while (turn < 2) {

System.out.println("Please enter how you would like to throw!");

System.out.println("1.Far Left (4,7) 2.Left (2,4) 3.Middle (1) 3.Right (3,6) 5.Far Right (6,10) 6. EXIT");

int playerThrow;

int speed;

boolean error = true;

boolean error2;

//asks user to input throw type and speed, calls method according to throw type and passes speed

do {

try {

playerThrow = input.nextInt();

if (playerThrow == 6) {

System.out.println("Sad to see you go. :(");

System.out.println("Returning to lobby...");

return 1000;

} else if (playerThrow >= 1 && playerThrow <= 6) {

System.out.println("Great! Now please enter how fast you would like to throw the ball (from 5mph to 30mph)");

do {

try {

speed = input.nextInt();

if (speed >= 5 && speed <= 30) {

error = false;

error2 = false;

switch (playerThrow) {

//calls method according to user input for throw type, passes speed

case 1 -> FarLeftThrow(speed);

case 2 -> LeftThrow(speed);

case 3 -> MidThrow(speed);

case 4 -> RightThrow(speed);

case 5 -> FarRightThrow(speed);

}

//prints alley results

System.out.println("This is the alley after your toss.");

BoardPrint();

//if no strike and second turn, frame is over so add up points of knocked over pins

if (!StrikeCheck() && turn == 1) {

System.out.println("You knocked over " + PinCheck() + " pins in total. That's neat. :D");

if (pin1.equals("[ ]")) {

score++;

}

if (pin2.equals("[ ]")) {

score++;

}

if (pin3.equals("[ ]")) {

score++;

}

if (pin4.equals("[ ]")) {

score++;

}

if (pin5.equals("[ ]")) {

score++;

}

if (pin6.equals("[ ]")) {

score++;

}

if (pin7.equals("[ ]")) {

score++;

}

if (pin8.equals("[ ]")) {

score++;

}

if (pin9.equals("[ ]")) {

score++;

}

if (pin10.equals("[ ]")) {

score++;

}

turn++;

//if no strike then print the number of pins that have been knocked over

} else if (!StrikeCheck()) {

System.out.println("You knocked over " + PinCheck() + " pins. That's neat. :D");

if (turn == 0) {

System.out.println("Time for your second throw!");

}

strikeTurnOne = false;

TimeUnit.SECONDS.sleep(5);

turn++;

//if strike check (all pins knocked over) but strike possibility is not possible, spare

} else if (StrikeCheck() && !strikeTurnOne) {

System.out.println("Spare! Neat! You get 10 points.");

System.out.println("Now resetting the pins...");

TimeUnit.SECONDS.sleep(5);

score += 10;

strikeTurnOne = true;

turn++;

//if strike and possibility is possible, strike

} else if (StrikeCheck()) {

System.out.println("Strike!!! Very neat. You get 10 points! :D");

System.out.println("Now resetting the pins...");

score += 10;

if (turnAgain) {

System.out.println("You got a strike on your first throw, which means you get one more throw!");

turnAgain = false;

turn--;

}

//reset pins after a strike

pin1 = "[0]";

pin2 = "[0]";

pin3 = "[0]";

pin4 = "[0]";

pin5 = "[0]";

pin6 = "[0]";

pin7 = "[0]";

pin8 = "[0]";

pin9 = "[0]";

pin10 = "[0]";

System.out.println("The alley has been reset.");

TimeUnit.SECONDS.sleep(5);

BoardPrint();

strikeTurnOne = true;

turn++;

}

} else {

System.out.println("Please enter a proper speed. :(");

error2 = true;

input.next();

}

} catch (Exception e) {

System.out.println("Please enter a proper speed. :(");

error2 = true;

}

} while (error2);

} else {

System.out.println("Please enter a proper throw!");

error = true;

}

} catch (Exception e) {

System.out.println("Please enter a proper throw...");

error = true;

}

} while (error);

}

return score;

}

//boardPrint method, prints the board with current state of pins

private static void BoardPrint() {

System.out.println("| " + pin7 + " " + pin8 + " " + pin9 + " " + pin10 + " |\n" +

"| " + pin4 + " " + pin5 + " " + pin6 + " |\n" +

"| " + pin2 + " " + pin3 + " |\n" +

"| " + pin1 + " |\n" +

"| | | |\n" +

"| |\n" +

"| | | |\n" +

"| |\n" +

"| |\n" +

"| |\n" +

"| |\n" +

"| |\n" +

"| ^ |\n" +

"| ^ ^ |\n" +

"| ^ ^ |\n" +

"|^ ^|\n" +

"| ..... ..... |\n" +

"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

}

//strikeCheck method, checks to see if all pins have been knocked over

private static boolean StrikeCheck() {

boolean strike;

strike = pin1.equals(pin2) && pin2.equals(pin3) && pin3.equals(pin4) && pin4.equals(pin5) && pin5.equals(pin6) &&

pin6.equals(pin7) && pin7.equals(pin8) && pin8.equals(pin9) && pin9.equals(pin10) && pin10.equals("[ ]");

return strike;

}

//pinCheck method, returns number of pins that have been knocked over

private static int PinCheck() {

int pin = 0;

if (pin1.equals("[ ]")) {

pin += 1;

}

if (pin2.equals("[ ]")) {

pin += 1;

}

if (pin3.equals("[ ]")) {

pin += 1;

}

if (pin4.equals("[ ]")) {

pin += 1;

}

if (pin5.equals("[ ]")) {

pin += 1;

}

if (pin6.equals("[ ]")) {

pin += 1;

}

if (pin7.equals("[ ]")) {

pin += 1;

}

if (pin8.equals("[ ]")) {

pin += 1;

}

if (pin9.equals("[ ]")) {

pin += 1;

}

if (pin10.equals("[ ]")) {

pin += 1;

}

return pin;

}

//farLeftThrow method, takes speed and sets probability that pin will fall, according to random variables

private static void FarLeftThrow(int speed) {

double random1 = Math.random() \* 100;

double random2 = Math.random() \* 100;

double random3 = Math.random() \* 100;

double random4 = Math.random() \* 100;

double random5 = Math.random() \* 100;

double random6 = Math.random() \* 100;

double random7 = Math.random() \* 100;

double random8 = Math.random() \* 100;

double random9 = Math.random() \* 100;

if (speed >= 5 && speed <= 10) {

if (random1 < 80) {

pin7 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 70) {

pin2 = "[ ]";

}

if (random3 < 55) {

pin1 = "[ ]";

}

if (random4 < 75) {

pin8 = "[ ]";

}

if (random5 < 70) {

pin9 = "[ ]";

}

if (random6 < 65) {

pin10 = "[ ]";

}

if (random7 < 70) {

pin5 = "[ ]";

}

if (random8 < 55) {

pin6 = "[ ]";

}

if (random9 < 55) {

pin3 = "[ ]";

}

System.out.println("The bowling ball rolls in a straight line towards the gutters... Maybe a lot faster.");

} else if (speed >= 10 && speed <= 15) {

if (random1 < 85) {

pin7 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 75) {

pin2 = "[ ]";

}

if (random3 < 65) {

pin1 = "[ ]";

}

if (random4 < 85) {

pin8 = "[ ]";

}

if (random5 < 75) {

pin9 = "[ ]";

}

if (random6 < 70) {

pin10 = "[ ]";

}

if (random7 < 75) {

pin5 = "[ ]";

}

if (random8 < 60) {

pin6 = "[ ]";

}

if (random9 < 60) {

pin3 = "[ ]";

}

System.out.println("The bowling ball rolls with a microscopic curve.. Maybe faster.");

} else if (speed >= 15 && speed <= 20) {

if (random1 < 90) {

pin7 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 80) {

pin2 = "[ ]";

}

if (random3 < 70) {

pin1 = "[ ]";

}

if (random4 < 90) {

pin8 = "[ ]";

}

if (random5 < 80) {

pin9 = "[ ]";

}

if (random6 < 75) {

pin10 = "[ ]";

}

if (random7 < 80) {

pin5 = "[ ]";

}

if (random8 < 65) {

pin6 = "[ ]";

}

if (random9 < 65) {

pin3 = "[ ]";

}

System.out.println("The bowling ball is curving a bit. Maybe faster...");

} else if (speed >= 21 && speed <= 24) {

if (random1 < 95) {

pin7 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 85) {

pin2 = "[ ]";

}

if (random3 < 75) {

pin1 = "[ ]";

}

if (random4 < 95) {

pin8 = "[ ]";

}

if (random5 < 85) {

pin9 = "[ ]";

}

if (random6 < 80) {

pin10 = "[ ]";

}

if (random7 < 85) {

pin5 = "[ ]";

}

if (random8 < 70) {

pin6 = "[ ]";

}

if (random9 < 70) {

pin3 = "[ ]";

}

System.out.println("The bowling ball appears to be slightly too slow.");

} else if (speed == 25) {

if (random1 < 100) {

pin7 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 90) {

pin2 = "[ ]";

}

if (random3 < 80) {

pin1 = "[ ]";

}

if (random4 < 95) {

pin8 = "[ ]";

}

if (random5 < 90) {

pin9 = "[ ]";

}

if (random6 < 85) {

pin10 = "[ ]";

}

if (random7 < 90) {

pin5 = "[ ]";

}

if (random8 < 75) {

pin6 = "[ ]";

}

if (random9 < 75) {

pin3 = "[ ]";

}

System.out.println("The bowling ball curves to perfection...");

} else if (speed >= 26 && speed <= 30) {

if (random1 < 100) {

pin7 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 95) {

pin2 = "[ ]";

}

if (random3 < 85) {

pin1 = "[ ]";

}

if (random4 < 100) {

pin8 = "[ ]";

}

if (random5 < 95) {

pin9 = "[ ]";

}

if (random6 < 90) {

pin10 = "[ ]";

}

if (random7 < 95) {

pin5 = "[ ]";

}

if (random8 < 80) {

pin6 = "[ ]";

}

if (random9 < 80) {

pin3 = "[ ]";

}

System.out.println("The bowling ball accelerates towards the gutters!");

}

}

//leftThrow method, takes speed and sets probability that pin will fall, according to random variables

private static void LeftThrow(int speed) {

double random1 = Math.random() \* 100;

double random2 = Math.random() \* 100;

double random3 = Math.random() \* 100;

double random4 = Math.random() \* 100;

double random5 = Math.random() \* 100;

double random6 = Math.random() \* 100;

double random7 = Math.random() \* 100;

double random8 = Math.random() \* 100;

double random9 = Math.random() \* 100;

if (speed >= 5 && speed <= 10) {

System.out.println("The bowling ball rolls slowly down the lane... Maybe a lot faster.");

} else if (speed >= 10 && speed <= 15) {

if (random1 < 90) {

pin2 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 80) {

pin1 = "[ ]";

}

if (random3 < 85) {

pin7 = "[ ]";

}

if (random4 < 80) {

pin8 = "[ ]";

}

if (random5 < 75) {

pin5 = "[ ]";

}

if (random6 < 75) {

pin3 = "[ ]";

}

if (random7 < 55) {

pin9 = "[ ]";

}

if (random8 < 65) {

pin6 = "[ ]";

}

if (random9 < 50) {

pin10 = "[ ]";

}

System.out.println("The bowling ball rolls at a steady pace.. Maybe faster.");

} else if (speed >= 15 && speed <= 22) {

if (random1 < 95) {

pin2 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 85) {

pin1 = "[ ]";

}

if (random3 < 90) {

pin7 = "[ ]";

}

if (random4 < 90) {

pin8 = "[ ]";

}

if (random5 < 80) {

pin5 = "[ ]";

}

if (random6 < 80) {

pin3 = "[ ]";

}

if (random7 < 60) {

pin9 = "[ ]";

}

if (random8 < 70) {

pin6 = "[ ]";

}

if (random9 < 55) {

pin10 = "[ ]";

}

System.out.println("The bowling ball is picking up speed. Maybe faster...");

} else if (speed == 23) {

if (random1 < 100) {

pin2 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 90) {

pin1 = "[ ]";

}

if (random3 < 95) {

pin7 = "[ ]";

}

if (random4 < 95) {

pin8 = "[ ]";

}

if (random5 < 85) {

pin5 = "[ ]";

}

if (random6 < 85) {

pin3 = "[ ]";

}

if (random7 < 65) {

pin9 = "[ ]";

}

if (random8 < 75) {

pin6 = "[ ]";

}

if (random9 < 60) {

pin10 = "[ ]";

}

System.out.println("The bowling ball appears to be at its optimal speed!");

} else if (speed >= 24 && speed <= 25) {

if (random1 < 100) {

pin2 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 85) {

pin1 = "[ ]";

}

if (random3 < 85) {

pin7 = "[ ]";

}

if (random4 < 90) {

pin8 = "[ ]";

}

if (random5 < 90) {

pin5 = "[ ]";

}

if (random6 < 80) {

pin3 = "[ ]";

}

if (random7 < 70) {

pin9 = "[ ]";

}

if (random8 < 80) {

pin6 = "[ ]";

}

if (random9 < 70) {

pin10 = "[ ]";

}

System.out.println("The bowling ball flies through the pins! Perhaps a bit too fast...");

} else if (speed >= 25 && speed <= 30) {

if (random1 < 100) {

pin2 = "[ ]";

pin4 = "[ ]";

}

if (random2 < 80) {

pin1 = "[ ]";

}

if (random3 < 80) {

pin7 = "[ ]";

}

if (random4 < 85) {

pin8 = "[ ]";

}

if (random5 < 95) {

pin5 = "[ ]";

}

if (random6 < 75) {

pin3 = "[ ]";

}

if (random7 < 80) {

pin9 = "[ ]";

}

if (random8 < 85) {

pin6 = "[ ]";

}

if (random9 < 75) {

pin10 = "[ ]";

}

System.out.println("The bowling ball won't smashes against the back! Definitely slower!");

}

}

//midThrow method, takes speed and sets probability that pin will fall, according to random variables

private static void MidThrow(int speed) {

double random1 = Math.random() \* 100;

double random2 = Math.random() \* 100;

double random3 = Math.random() \* 100;

double random4 = Math.random() \* 100;

double random5 = Math.random() \* 100;

double random6 = Math.random() \* 100;

double random7 = Math.random() \* 100;

double random8 = Math.random() \* 100;

double random9 = Math.random() \* 100;

if (speed >= 5 && speed <= 10) {

if (random1 < 85) {

pin1 = "[ ]";

pin5 = "[ ]";

}

if (random2 < 80) {

pin2 = "[ ]";

}

if (random3 < 80) {

pin3 = "[ ]";

}

if (random4 < 75) {

pin4 = "[ ]";

}

if (random5 < 75) {

pin6 = "[ ]";

}

if (random6 < 55) {

pin7 = "[ ]";

}

if (random7 < 70) {

pin8 = "[ ]";

}

if (random8 < 70) {

pin9 = "[ ]";

}

if (random9 < 55) {

pin10 = "[ ]";

}

System.out.println("The bowling ball rolls slowly down the lane... Maybe a lot faster.");

} else if (speed >= 10 && speed <= 15) {

if (random1 < 90) {

pin1 = "[ ]";

pin5 = "[ ]";

}

if (random2 < 85) {

pin2 = "[ ]";

}

if (random3 < 85) {

pin3 = "[ ]";

}

if (random4 < 80) {

pin4 = "[ ]";

}

if (random5 < 80) {

pin6 = "[ ]";

}

if (random6 < 60) {

pin7 = "[ ]";

}

if (random7 < 75) {

pin8 = "[ ]";

}

if (random8 < 75) {

pin9 = "[ ]";

}

if (random9 < 60) {

pin10 = "[ ]";

}

System.out.println("The bowling ball rolls at a steady pace.. Maybe faster.");

} else if (speed >= 15 && speed <= 20) {

if (random1 < 95) {

pin1 = "[ ]";

pin5 = "[ ]";

}

if (random2 < 90) {

pin2 = "[ ]";

}

if (random3 < 90) {

pin3 = "[ ]";

}

if (random4 < 85) {

pin4 = "[ ]";

}

if (random5 < 85) {

pin6 = "[ ]";

}

if (random6 < 70) {

pin7 = "[ ]";

}

if (random7 < 80) {

pin8 = "[ ]";

}

if (random8 < 80) {

pin9 = "[ ]";

}

if (random9 < 70) {

pin10 = "[ ]";

}

System.out.println("The bowling ball is picking up speed. Maybe faster...");

} else if (speed == 21) {

if (random1 < 100) {

pin1 = "[ ]";

pin5 = "[ ]";

}

if (random2 < 95) {

pin2 = "[ ]";

}

if (random3 < 95) {

pin3 = "[ ]";

}

if (random4 < 90) {

pin4 = "[ ]";

}

if (random5 < 90) {

pin6 = "[ ]";

}

if (random6 < 75) {

pin7 = "[ ]";

}

if (random7 < 85) {

pin8 = "[ ]";

}

if (random8 < 85) {

pin9 = "[ ]";

}

if (random9 < 75) {

pin10 = "[ ]";

}

System.out.println("The bowling ball appears to be at its optimal speed!");

} else if (speed >= 22 && speed <= 25) {

if (random1 < 100) {

pin1 = "[ ]";

pin5 = "[ ]";

}

if (random2 < 95) {

pin2 = "[ ]";

}

if (random3 < 95) {

pin3 = "[ ]";

}

if (random4 < 85) {

pin4 = "[ ]";

}

if (random5 < 85) {

pin6 = "[ ]";

}

if (random6 < 70) {

pin7 = "[ ]";

}

if (random7 < 90) {

pin8 = "[ ]";

}

if (random8 < 90) {

pin9 = "[ ]";

}

if (random9 < 70) {

pin10 = "[ ]";

}

System.out.println("The bowling ball flies through the pins! Perhaps a bit too fast...");

} else if (speed >= 25 && speed <= 30) {

if (random1 < 100) {

pin1 = "[ ]";

pin5 = "[ ]";

}

if (random2 < 95) {

pin2 = "[ ]";

}

if (random3 < 95) {

pin3 = "[ ]";

}

if (random4 < 80) {

pin4 = "[ ]";

}

if (random5 < 80) {

pin6 = "[ ]";

}

if (random6 < 65) {

pin7 = "[ ]";

}

if (random7 < 90) {

pin8 = "[ ]";

}

if (random8 < 90) {

pin9 = "[ ]";

}

if (random9 < 65) {

pin10 = "[ ]";

}

System.out.println("The bowling ball won't smashes against the back! Definitely slower!");

}

}

//rightThrow method, takes speed and sets probability that pin will fall, according to random variables

private static void RightThrow(int speed) {

double random1 = Math.random() \* 100;

double random2 = Math.random() \* 100;

double random3 = Math.random() \* 100;

double random4 = Math.random() \* 100;

double random5 = Math.random() \* 100;

double random6 = Math.random() \* 100;

double random7 = Math.random() \* 100;

double random8 = Math.random() \* 100;

double random9 = Math.random() \* 100;

if (speed >= 5 && speed <= 10) {

if (random1 < 90) {

pin6 = "[ ]";

pin3 = "[ ]";

}

if (random2 < 80) {

pin1 = "[ ]";

}

if (random3 < 85) {

pin10 = "[ ]";

}

if (random4 < 80) {

pin9 = "[ ]";

}

if (random5 < 75) {

pin5 = "[ ]";

}

if (random6 < 75) {

pin2 = "[ ]";

}

if (random7 < 55) {

pin8 = "[ ]";

}

if (random8 < 65) {

pin4 = "[ ]";

}

if (random9 < 50) {

pin7 = "[ ]";

}

System.out.println("The bowling ball rolls slowly down the lane... Maybe a lot faster.");

} else if (speed >= 10 && speed <= 15) {

if (random1 < 95) {

pin6 = "[ ]";

pin3 = "[ ]";

}

if (random2 < 85) {

pin1 = "[ ]";

}

if (random3 < 90) {

pin10 = "[ ]";

}

if (random4 < 85) {

pin9 = "[ ]";

}

if (random5 < 80) {

pin5 = "[ ]";

}

if (random6 < 80) {

pin2 = "[ ]";

}

if (random7 < 60) {

pin8 = "[ ]";

}

if (random8 < 70) {

pin4 = "[ ]";

}

if (random9 < 55) {

pin7 = "[ ]";

}

System.out.println("The bowling ball rolls at a steady pace.. Maybe faster.");

} else if (speed >= 15 && speed <= 22) {

if (random1 < 100) {

pin6 = "[ ]";

pin3 = "[ ]";

}

if (random2 < 90) {

pin1 = "[ ]";

}

if (random3 < 95) {

pin10 = "[ ]";

}

if (random4 < 90) {

pin9 = "[ ]";

}

if (random5 < 85) {

pin5 = "[ ]";

}

if (random6 < 85) {

pin2 = "[ ]";

}

if (random7 < 65) {

pin8 = "[ ]";

}

if (random8 < 75) {

pin4 = "[ ]";

}

if (random9 < 60) {

pin7 = "[ ]";

}

System.out.println("The bowling ball is picking up speed. Maybe faster...");

} else if (speed == 23) {

if (random1 < 100) {

pin6 = "[ ]";

pin3 = "[ ]";

}

if (random2 < 95) {

pin1 = "[ ]";

}

if (random3 < 95) {

pin10 = "[ ]";

}

if (random4 < 95) {

pin9 = "[ ]";

}

if (random5 < 90) {

pin5 = "[ ]";

}

if (random6 < 90) {

pin2 = "[ ]";

}

if (random7 < 75) {

pin8 = "[ ]";

}

if (random8 < 85) {

pin4 = "[ ]";

}

if (random9 < 75) {

pin7 = "[ ]";

}

System.out.println("The bowling ball appears to be at its optimal speed!");

} else if (speed >= 24 && speed <= 25) {

if (random1 < 100) {

pin6 = "[ ]";

pin3 = "[ ]";

}

if (random2 < 95) {

pin1 = "[ ]";

}

if (random3 < 90) {

pin10 = "[ ]";

}

if (random4 < 90) {

pin9 = "[ ]";

}

if (random5 < 95) {

pin5 = "[ ]";

}

if (random6 < 80) {

pin2 = "[ ]";

}

if (random7 < 85) {

pin8 = "[ ]";

}

if (random8 < 80) {

pin4 = "[ ]";

}

if (random9 < 85) {

pin7 = "[ ]";

}

System.out.println("The bowling ball flies through the pins! Perhaps a bit too fast...");

} else if (speed >= 25 && speed <= 30) {

if (random1 < 100) {

pin6 = "[ ]";

pin3 = "[ ]";

}

if (random2 < 90) {

pin1 = "[ ]";

}

if (random3 < 95) {

pin10 = "[ ]";

}

if (random4 < 85) {

pin9 = "[ ]";

}

if (random5 < 95) {

pin5 = "[ ]";

}

if (random6 < 75) {

pin2 = "[ ]";

}

if (random7 < 90) {

pin8 = "[ ]";

}

if (random8 < 85) {

pin4 = "[ ]";

}

if (random9 < 90) {

pin7 = "[ ]";

}

System.out.println("The bowling ball won't smashes against the back! Definitely slower!");

}

}

//farRightThrow method, takes speed and sets probability that pin will fall, according to random variables

private static void FarRightThrow(int speed) {

double random1 = Math.random() \* 100;

double random2 = Math.random() \* 100;

double random3 = Math.random() \* 100;

double random4 = Math.random() \* 100;

double random5 = Math.random() \* 100;

double random6 = Math.random() \* 100;

double random7 = Math.random() \* 100;

double random8 = Math.random() \* 100;

double random9 = Math.random() \* 100;

if (speed >= 5 && speed <= 10) {

if (random1 < 80) {

pin6 = "[ ]";

pin10 = "[ ]";

}

if (random2 < 70) {

pin3 = "[ ]";

}

if (random3 < 55) {

pin1 = "[ ]";

}

if (random4 < 75) {

pin9 = "[ ]";

}

if (random5 < 70) {

pin8 = "[ ]";

}

if (random6 < 65) {

pin7 = "[ ]";

}

if (random7 < 70) {

pin5 = "[ ]";

}

if (random8 < 55) {

pin4 = "[ ]";

}

if (random9 < 55) {

pin2 = "[ ]";

}

System.out.println("The bowling ball rolls in a straight line towards the gutters... Maybe a lot faster.");

} else if (speed >= 10 && speed <= 15) {

if (random1 < 85) {

pin6 = "[ ]";

pin10 = "[ ]";

}

if (random2 < 75) {

pin3 = "[ ]";

}

if (random3 < 60) {

pin1 = "[ ]";

}

if (random4 < 80) {

pin9 = "[ ]";

}

if (random5 < 75) {

pin8 = "[ ]";

}

if (random6 < 70) {

pin7 = "[ ]";

}

if (random7 < 75) {

pin5 = "[ ]";

}

if (random8 < 60) {

pin4 = "[ ]";

}

if (random9 < 60) {

pin2 = "[ ]";

}

System.out.println("The bowling ball rolls with a microscopic curve.. Maybe faster.");

} else if (speed >= 15 && speed <= 20) {

if (random1 < 90) {

pin6 = "[ ]";

pin10 = "[ ]";

}

if (random2 < 80) {

pin3 = "[ ]";

}

if (random3 < 65) {

pin1 = "[ ]";

}

if (random4 < 85) {

pin9 = "[ ]";

}

if (random5 < 80) {

pin8 = "[ ]";

}

if (random6 < 75) {

pin7 = "[ ]";

}

if (random7 < 80) {

pin5 = "[ ]";

}

if (random8 < 65) {

pin4 = "[ ]";

}

if (random9 < 65) {

pin2 = "[ ]";

}

System.out.println("The bowling ball is curving a bit. Maybe faster...");

} else if (speed >= 21 && speed <= 24) {

if (random1 < 95) {

pin6 = "[ ]";

pin10 = "[ ]";

}

if (random2 < 85) {

pin3 = "[ ]";

}

if (random3 < 70) {

pin1 = "[ ]";

}

if (random4 < 90) {

pin9 = "[ ]";

}

if (random5 < 85) {

pin8 = "[ ]";

}

if (random6 < 80) {

pin7 = "[ ]";

}

if (random7 < 85) {

pin5 = "[ ]";

}

if (random8 < 70) {

pin4 = "[ ]";

}

if (random9 < 70) {

pin2 = "[ ]";

}

System.out.println("The bowling ball appears to be slightly too slow.");

} else if (speed == 25) {

if (random1 < 100) {

pin6 = "[ ]";

pin10 = "[ ]";

}

if (random2 < 90) {

pin3 = "[ ]";

}

if (random3 < 75) {

pin1 = "[ ]";

}

if (random4 < 95) {

pin9 = "[ ]";

}

if (random5 < 90) {

pin8 = "[ ]";

}

if (random6 < 85) {

pin7 = "[ ]";

}

if (random7 < 90) {

pin5 = "[ ]";

}

if (random8 < 75) {

pin4 = "[ ]";

}

if (random9 < 75) {

pin2 = "[ ]";

}

System.out.println("The bowling ball curves to perfection...");

} else if (speed >= 26 && speed <= 30) {

if (random1 < 100) {

pin6 = "[ ]";

pin10 = "[ ]";

}

if (random2 < 95) {

pin3 = "[ ]";

}

if (random3 < 70) {

pin1 = "[ ]";

}

if (random4 < 100) {

pin9 = "[ ]";

}

if (random5 < 95) {

pin8 = "[ ]";

}

if (random6 < 90) {

pin7 = "[ ]";

}

if (random7 < 95) {

pin5 = "[ ]";

}

if (random8 < 70) {

pin4 = "[ ]";

}

if (random9 < 70) {

pin2 = "[ ]";

}

System.out.println("The bowling ball accelerates towards the gutters!");

}

}

}