/\*

RockPaperScissors.java

Purpose: Simulates a Rock Paper Scissors game with a user against the computer

Written by: Diana Eastman

Modified date: February 11, 2013

\*/

import java.util.\*;

public class RockPaperScissors

{

// private member variables

private int computerHand;

private int userHand;

private int numWins;

private int numTies;

private int numLosses;

private int gameStatus;

// Constructor

public RockPaperScissors ()

{

computerHand = 1; // Initialize to rock

userHand = 1; // Initiatilize to rock

gameStatus = 1; // Initialize to win

numWins = numTies = numLosses = 0;

}

// Methods

public void intro()

{

System.out.println("Welcome to Rock Paper Scissors");

}

public boolean GenerateHands()

{

// Generate the computer hand by getting a random number between 0 and 2

Random random = new Random();

computerHand = random.nextInt(3) ;

// Generate user hand

Scanner in = new Scanner(System.in);

String input = "";

// Ask for human input and valid the input given; if invalid (i.e., not in 0-3), prompt again

while (true)

{

System.out.print("Select your weapon or quit the game: 0 = Rock, 1 = Scissors, 2 = Paper, 3 = QUIT GAME");

input = in.nextLine();

try

{

userHand = Integer.parseInt(input);

if ((userHand >= 0) && (userHand <= 3))

break;

}

catch (NumberFormatException e)

{

// Do nothing and ask for input again

}

}

if (userHand!=3) { // Quit the game

return true;

}

else {

return false;

}

}

//Compare computerHand and UserHand, determine who won, and then increment the running totals

public void CompareHands() // Rock = 0, Scissors = 1, Paper = 2

{

if (computerHand == userHand) {

numTies++;

gameStatus = 0;

}

else if ((computerHand - userHand + 3) % 3 == 1)

{

numWins++;

gameStatus = 1;

}

else {

numLosses++;

gameStatus = 2;

}

}

// Rock beats scissors (0 over 1)

// Paper beats rock (2 over 0)

// Scissors beats paper (1 over 2)

private String statustoString(int value)

{

String status = "";

switch (value)

{

case 1: status = "Win";

break;

case 0: status = "Tie";

break;

case 2: status = "Lose";

break;

}

return (status);

}

public String getGameStatus()

{

return (statustoString(gameStatus));

}

private String handtoString(int value)

{

String hand = "";

switch (value)

{

case 0: hand = "Rock";

break;

case 1: hand = "Scissors";

break;

case 2: hand = "Paper";

break;

}

return (hand);

}

public String getComputerHand()

{

return (handtoString(computerHand));

}

public String getUserHand()

{

return (handtoString(userHand));

}

//toString method will print out the running totals when the user wants to exit the game

public String toString()

{

return ("Num Wins: " + numWins + "\n" +

"Num Ties: " + numTies + "\n" +

"Num Losses: " + numLosses + "\n");

}

public static void main (String[] args)

{

RockPaperScissors myGame = new RockPaperScissors();

boolean keepGoing = true;

myGame.intro();

while (true)

{

keepGoing = myGame.GenerateHands();

if (keepGoing == false)

{

System.out.println(myGame);

break;

}

myGame.CompareHands();

System.out.println("The computer's hand is: " + myGame.getComputerHand());

System.out.println("Your hand is: " + myGame.getUserHand());

System.out.println("You " + myGame.getGameStatus() + " this round.");

}

}

}

/\*

Triangle.java

Purpose: A class that describes the properties of a triangle and includes methods

that perform actions on these properties, such as computing the area.

CS230 Homework Assignment P2

Written by: Diana Eastman

Modified date:

\*/

public class Triangle

{

// Private data declarations

private double side1;

private double side2;

private double side3;

private double area;

// Constructor

public Triangle()

{

side1 = 0;

side2 = 0;

side3 = 0;

area = 0;

}

// Methods

/\*\* Given an input string of three integers, splits the string into the three

\* constituent integers at the white space delimiter and determines if three values

\* have been inputted

\* @param input the side lengths of a triangle as one string

\* @return boolean true for three sides inputted

\*/

public boolean getSides(String sideLength)

{

String[] triangleSideArray;

double[] sides = null;

triangleSideArray = (sideLength.split("\\s+"));;

sides = new double[triangleSideArray.length];

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

{

sides[i] = Double.parseDouble(triangleSideArray[i]);

//System.out.println("Array:" + sides[index]);

}

if (validateSides(sides))

{

side1 = sides[0];

side2 = sides[1];

side3 = sides[2];

return true;

}

else

{

return false;

}

}

/\*\* Determines if the sidelengths provided make a valid

\* triangle and returns true if valid

\* @param input an array of three side lengths

\* @return boolean true for valid triangle

\*/

private boolean validateSides(double[] sides)

{

if (sides.length == 3)

{

if ((sides[0] + sides[1] > sides[2]) &&

(sides[1] + sides[2] > sides[0]) &&

(sides[0] + sides[2] > sides[1]))

{

return true;

}

}

return false;

}

/\*\* Computes and returns the area of a triangle

\* using Heron's Formula

\* @return triangle area

\*/

public double ComputeHeronsArea()

{

double semiPerimeter = (side1 + side2 + side3)/2;

area = Math.sqrt(semiPerimeter \* (semiPerimeter - side1) \*

(semiPerimeter - side2) \*

(semiPerimeter - side3));

// Round to nearest thousandth and return

//area = Math.round(1000\*area)/((double)(1000));

return (area);

}

/\*\* Gets the difference between two triangle areas

\* @param input first triangle area

\* @param input second triangle area

\* @return difference between two triangle areas

\*/

public static double getDifference(double firstArea, double secondArea)

{

double difference = Math.abs(firstArea - secondArea);

difference = Math.round(1000\*difference)/((double)(1000));

return difference;

}

/\*\* Compares the area of two triangles and determines if they are

\* essentially the same, given a methematical tolerance

\* @param input first triangle area

\* @param input second triangle area

\* @param input tolerance

\* @return int

\*/

public static int compareArea(double firstArea, double secondArea, double tolerance)

{

double difference = firstArea - secondArea;

if (Math.abs(difference) < tolerance)

return 0;

if (difference > 0)

return 1;

else

return 2;

}

}

/\*

HeronsFormula.java

Purpose: Compare the area of two triangles and determine if they are equal within 0.01 sq. units.

If the areas are within the tolerance, print that the areas are essentially the same.

CS230 Homework Assignment P2

Written by: Diana Eastman

Modified date:

\*/

import java.util.\*;

public class HeronsFormula

{

public static final double TOLERANCE = 0.01;

private static double difference = 0.01;

public static void main (String[] args)

{

Triangle triangle1 = new Triangle();

Triangle triangle2 = new Triangle();

System.out.println("Let's do some geometry. This program will compare the area of two " +

"triangles and determine if they are equal within 0.01 sq. units. Enter triangle 1's " +

"three side lengths, separated by spaces.");

Scanner in = new Scanner(System.in);

boolean invalid1 = true;

do

{

System.out.println("The lengths must make a valid triangle.");

String sideLength = in.nextLine();

if (triangle1.getSides(sideLength))

invalid1 = false;

} while (invalid1);

System.out.println("Enter triangle 2's three side lengths, separated by spaces.");

boolean invalid2 = true;

do

{

System.out.println("The lengths must make a valid triangle.");

String sideLength = in.nextLine();

if (triangle2.getSides(sideLength))

invalid2 = false;

} while (invalid2);

// Compare the areas and output result

difference = Triangle.getDifference(triangle1.ComputeHeronsArea(), triangle2.ComputeHeronsArea());

switch(Triangle.compareArea(triangle1.ComputeHeronsArea(), triangle2.ComputeHeronsArea(), TOLERANCE))

{

case 0: System.out.println("The two triangles have essentially the same area" +

" because their difference is " + difference +

" which is less than the permitted TOLERANCE of 0.01");

break;

case 1: System.out.println("The first triangle is " + difference + " bigger than the second");

break;

case 2: System.out.println("The second triangle is " + difference + " bigger than the first");

break;

}

}

}

/\*

StringOps.java

Purpose: Contains two methods that operate on strings - one to remove a specific character if

contained in a string and another to determine if two strings are anagrams. Assumes, as described

in the homework problem, that strings are in all lowercase.

CS230 Homework Assignment P2

Written by: Diana Eastman

Created date: February 11, 2013

\*/

import java.util.\*;

public class StringOps {

/\*\* Looks for the first instance of a given character in

\* a specified string; if found, the character is removed,

\* otherwise, the original string is returned.

\* @param name a String

\* @param numRounds a character

\*/

public static String removeChar (String S, char ch) {

String result="";

int index = -1;

index = S.indexOf(ch);

char[] charArray = S.toCharArray();

char[] charcopyArray = new char[charArray.length];

int i=0;

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

if (i!=index){

charcopyArray[i]=charArray[i];

}

}

result = new String(charcopyArray);

return result;

}

/\*\* Tests if two provided words are anagrams and

\* prints a sentence indicating the result of the

\* test. Two words that are the same word are not

\* considered anagrams

\* @param name a String

\* @param numRounds a String

\*/

public static void testAnagrams (String word1, String word2)

{

if ((!word1.equals(word2)) && (word1.length()==word2.length()))

{

char[] chars1 = word1.toCharArray();

char[] chars2 = word2.toCharArray();

Arrays.sort(chars1);

Arrays.sort(chars2);

String sorted1 = new String(chars1);

String sorted2 = new String(chars2);

if (sorted1.equals(sorted2))

System.out.println(word1 + " and " + word2 + " are anagrams.");

}

else {

System.out.println(word1 + " and " + word2 + " are not anagrams.");

}

}

//Testing method

public static void main (String args[]){

String teststring1;

String s1="e";

char c = s1.charAt(0);

teststring1= StringOps.removeChar("lemon", c);

String teststring2;

String s2="e";

char d = s2.charAt(0);

teststring2= StringOps.removeChar("java", c);

System.out.println("The result is: " + teststring1);

System.out.println("The result is: " + teststring2);

StringOps.testAnagrams("canter", "nectar");

StringOps.testAnagrams("hello", "hello");

StringOps.testAnagrams("java", "bean");

StringOps.testAnagrams("java", "");

StringOps.testAnagrams("java", "jav");

}

}

/\*\* CODE TEMPLATE

\* Creates the "hand" of a player by creating sn object of 5 random dice

\* @author Takis Metaxas

\* @version %I%, %G%

\*/

/\*

DicePlayer.java

CS230 Homework Assignment P4

Modified by: Diana Eastman

Modified date: February 11, 2013

\*/

import java.util.\*;

public class DicePlayer {

final private int hand = 5;

private Die[] FiveDice = new Die[hand];

String playerName;

/\*\* Constructor: Creates a player's hand by creating and rolling dice.

\*/

public DicePlayer(String name)

{

for (int i = 0; i < hand; i++)

{

FiveDice[i] = new Die();

}

playerName = name;

}

/\*\* Prints the contents of the hand that a player holds.

\*/

public String toString() {

String s = "";

int [] valueArray = new int[hand];

for (int i = 0; i < hand; i++)

{

valueArray[i] = FiveDice[i].getFaceValue();

}

s = playerName + " rolled: " + Arrays.toString(valueArray);

return s;

}

/\*\* Returns an array the contents of the hand that a player holds.

\* @return integer array of @param hand values corresponding to dice

\*/

public int[] getValues() {

int [] values = new int[hand];

int i = 0;

for (i = 0; i< values.length; i++)

{

values[i]=FiveDice[i].roll();

}

return values;

}

/\*\* Testing method.

\*/

public static void main (String args[]) {

DicePlayer hal = new DicePlayer("Hal");

hal.getValues();

System.out.println(hal);

DicePlayer dave = new DicePlayer("Dave");

dave.getValues();

System.out.println(dave);

}

}

/\*\* Simulates a Dice Poker game played between the computer and user.

\* This class definition contains a main() method that assumes

\* that the user enters a name and an integer in the command line, for example:

\* java PlayDice Wendy 7

\* If the user does not enter these arguments, however, default values are

\* provided. The default name is "Dave" and the default number of rounds is 5.

\* @author Takis Metaxas

\* @version %I%, %G%

\*/

/\*

DiceGame.java

CS230 Homework Assignment P4

Modified by: Diana Eastman

Modified date: February 11, 2013

\*/

import java.util.\*;

public class DiceGame {

/\*\* Creates a game object that contains the variables associated with a game.

\*/

String name; // Dave is supplied vy default

int numRounds; // 5 rounds by default

int pwin; //round wins of the player

int cwin; //round wins of the computer

private DicePlayer computerplayer;

private DicePlayer humanplayer;

//Provide an array of strings to map the ranks to the names of the hands

private static String[] ranks = {"Nothing", "One Pair", "Two Pair", "Three of a Kind", "Full House", "Four of a Kind", "Five of a Kind" };

public DiceGame(String name)

{

name = "Dave";

numRounds = 5;

pwin = 0;

cwin = 0;

humanplayer = new DicePlayer(name);

computerplayer = new DicePlayer("Hal");

}

/\*\* Counts how many distinct values appear in the input array

\* and stores each count onto the diceResults array.

\* PRE-CONDITION: diceResults[] should have enough length

\* to accomodate the values found in the input array.

\*

\* @param input the input array

\* @param diceResults holds the multiplicity of values found in input.

\* Note that this is actually the output parameter

\*/

private void accumulateValues(int[] input, int[] diceResults)

{

int count[] = new int[6];

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

{

int value = input[i];

count[value-1]++;

}

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

{

//An array that accumulates the number of occurrences mapped to the array's position

//(e.g., pairs are at position 2, three of a kind is at position 3, and so forth)

diceResults[count[i]]+=1;

}

}

/\*\* Given an input array storing five dice values,

\* determines the rank of the set of values

\* @param input the array with the five dice values

\* @return the rank: an integer between 0 and 6

\*/

private int getRank (int[] input)

{

int[] diceResults = new int[6];

accumulateValues(input, diceResults);

int rank = 0;

if (diceResults[2]==1 && diceResults[3]==0) {

rank = 1;

}

else if (diceResults[2]==2) {

rank = 2;

}

else if (diceResults[3]==1 &&diceResults[2]==0) {

rank = 3;

}

else if (diceResults[3]==1 && diceResults[2]==1) {

rank = 4;

}

else if (diceResults[4]==1) {

rank = 5;

}

else if (diceResults[5]==1) {

rank =6;

}

else {

rank = 0;

}

rankToString(rank);

return rank;

}

public static String rankToString (int rank)

{

return ranks[rank];

}

/\*\* Plays one round of the game: First the computer's turn,

\* then the player's turn

\* @param name the player's name

\* @return 0 if computer wins the round, 1 if player wins, 2 if a tie

\*/

private int playOneRound(String name) {

int compRank = 0;

String compRankString = "";

int humanRank = 0;

String humanRankString = "";

compRank = getRank(computerplayer.getValues());

humanRank = getRank(humanplayer.getValues());

compRankString = (computerplayer.toString() + " " + rankToString(compRank));

humanRankString = (humanplayer.toString() + " " + rankToString(humanRank));

System.out.println(compRankString);

System.out.println(humanRankString);

if (compRank > humanRank)

return 0;

if (humanRank > compRank)

return 1;

else return 2;

}

/\*\* Simulates the playing of numRounds of the Dice Poker game between

\* HAL and player name, and prints the winner at the end

\* @param name the player's name

\* @param numRounds the number of rounds to play

\*/

public void playDiceGame (String name, int numRounds) {

System.out.println("Good evening, " + name + ". Everything's running smoothly. And you?" + ".");

System.out.println("I'm completely operational and all my circuits are functioning perfectly.");

System.out.println("Would you like to play a game of Dice Poker? I play very well.");

int rounds=1;

do{

System.out.println("\*\*\* ROUND " + rounds);

int winner = playOneRound(name);

if(winner == 0) cwin++;

if(winner == 1) pwin++;

} while(rounds++ < numRounds);

// After all rounds played, determine the final winner of the game and print the results

if (pwin>cwin) System.out.print("The game was won by "+ name + " with a score of " + pwin + " to " + cwin);

else if (cwin>pwin) System.out.print("The game was won by the HAL with a score of " + cwin + " to " + pwin);

else System.out.print("The game was tied with a score of " + cwin + " to " + pwin);

System.out.println(" in " + numRounds + " rounds.");

System.out.println("Thank you for a very enjoyable game!");

}

public static void main (String args[]) {

// Create an instance of a new game and play the rounds

String name = (args.length > 0)? args[0] : "Dave";

DiceGame game = new DiceGame(name);

// 5 rounds by default

int numRounds = (args.length > 1)? Integer.parseInt(args[1]) : 5;

game.playDiceGame(name, numRounds);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Die.java Java Foundations

//

// Solution to Programming Project 5.10 & 5.11 (1st Ed, p. 232)

//

// Represents one die (singular of dice) with faces showing values

// between 1 and the number of faces on the die.

// CLASS NOT MODIFIED

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class Die

{

private final int MAX = 6; // maximum face value

private int faceValue; // current value showing on the die

//-----------------------------------------------------------------

// Constructor: sets the initial face value.

//-----------------------------------------------------------------

public Die()

{

faceValue = 1;

}

//-----------------------------------------------------------------

// Rolls the die and returns the result.

//-----------------------------------------------------------------

public int roll()

{

faceValue = (int) (Math.random() \* MAX) + 1;

return faceValue;

}

//-----------------------------------------------------------------

// Face value mutator.

//-----------------------------------------------------------------

public void setFaceValue (int value)

{

faceValue = value;

}

//-----------------------------------------------------------------

// Face value accessor.

//-----------------------------------------------------------------

public int getFaceValue()

{

return faceValue;

}

//-----------------------------------------------------------------

// Returns a string representation of this die.

//-----------------------------------------------------------------

public String toString()

{

String result = Integer.toString(faceValue);

return result;

}

}