Nick Troccoli Practice Final 2

CS 106A August 16, 2017

Solutions to Practice Final Exam 2

Based on handouts by Marty Stepp, Mehran Sahami, Eric Roberts and Patrick Young

**Problem 1: Java expressions, statements, and methods**

**a) [ok, dog, horse, horse]**

**b) [fruit, hyena, bird, hello, hello]**

**c) [hhh, gg, e]**

**d) [doggie, robot]**

**Problem 2: Nim**

**public class Nim extends GraphicsProgram {**

**private ArrayList<GOval> coinList;**

**private boolean isPlayer1sTurn;**

**public void run() {**

**isPlayer1sTurn = true;**

**setupCoins();**

**}**

**// This method draws N\_COINS horizontally and vertically centered.**

**private void setupCoins() {**

**double widthNeeded = N\_COINS \* COIN\_SIZE + (N\_COINS - 1) \* COIN\_SEP;**

**double x = (getWidth() - widthNeeded) / 2.0;**

**double y = (getHeight() - COIN\_SIZE) / 2.0;**

**coinList = new ArrayList<>();**

**for (int i = 0; i < N\_COINS; i++) {**

**GOval coin = new GOval(x, y, COIN\_SIZE, COIN\_SIZE);**

**coin.setFilled(true);**

**coin.setFillColor(Color.GRAY);**

**add(coin);**

**coinList.add(coin);**

**x += COIN\_SIZE + COIN\_SEP;**

**}**

**}**

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

**GObject obj = getElementAt(event.getX(), event.getY());**

**if (obj != null) {**

**int index = coinList.indexOf(obj);**

**if (index >= coinList.size() - 3) {**

**// Flip our turn boolean**

**isPlayer1sTurn = !isPlayer1sTurn;**

**// Remove coins from back to front**

**for (int i = coinList.size() - 1; i >= index; i--) {**

**remove(coinList.get(i));**

**coinList.remove(i);**

**}**

**if (coinList.size() == 0) {**

**if (isPlayer1sTurn) {**

**add(new GLabel("Player 1 wins!"), 50, 50);**

**} else {**

**add(new GLabel("Player 2 wins!"), 50, 50);**

**}**

**}**

**}**

**}**

**}**

**}**

**Problem 3: Sequences**

**// solution 1: nested loops**

**private boolean contains1(int[] a1, int[] a2) {**

**for (int i = 0; i <= a1.length - a2.length; i++) {**

**boolean found = true;**

**for (int j = 0; j < a2.length; j++) {**

**if (a1[i + j] != a2[j]) {**

**found = false;**

**}**

**}**

**if (found) {**

**return true;**

**}**

**}**

**return false;**

**}**

**// solution 2: uses a count instead of a boolean**

**private boolean contains2(int[] a1, int[] a2) {**

**for (int i = 0; i <= a1.length - a2.length; i++) {**

**int count = 0;**

**for (int j = 0; j < a2.length; j++) {**

**if (a1[i + j] == a2[j]) {**

**count++;**

**}**

**}**

**if (count == a2.length) {**

**return true;**

**}**

**}**

**return false;**

**}**

**// solution 3: a single while loop**

**private boolean contains3(int[] a1, int[] a2) {**

**int i1 = 0;**

**int i2 = 0;**

**while (i1 < a1.length && i2 < a2.length) {**

**if (a1[i1] != a2[i2]) { // doesn't match; start over**

**i2 = 0;**

**}**

**if (a1[i1] == a2[i2]) { // important NOT to use else-if here**

**i2++;**

**}**

**i1++;**

**}**

**return i2 >= a2.length;**

**}**

**// solution 4: for loop with inner while loop**

**private boolean contains4(int[] a1, int[] a2) {**

**for (int i = 0; i < a1.length; i++) {**

**int j = 0;**

**while (j < a2.length && i + j < a1.length && a1[i + j] == a2[j]) {**

**j++;**

**}**

**if (j == a2.length) {**

**return true;**

**}**

**}**

**return false;**

**}**

**Problem 4: Image Tiling**

**// solution 1: result image pixel based**

**private void tile1(GImage source, int width, int height) {**

**int[][] pixels = source.getPixelArray();**

**int[][] result = new int[height][width];**

**for (int row = 0; row < height; row++) {**

**for (int col = 0; col < width; col++) {**

**result[row][col] = pixels[row % pixels.length][col % pixels[0].length];**

**}**

**}**

**source.setPixelArray(result);**

**}**

**// solution 2: source image pixel based**

**private void tile2(GImage source, int width, int height) {**

**int[][] pixels = source.getPixelArray();**

**int[][] result = new int[height][width];**

**for (int row = 0; row < pixels.length; row++) {**

**for (int col = 0; col < pixels[0].length; col++) {**

**// tile the individual pixel of pixels[row][col]**

**for (int y = row; y < height; y += pixels.length) {**

**for (int x = col; x < width; x += pixels[0].length) {**

**result[y][x] = pixels[row][col];**

**}**

**}**

**}**

**}**

**source.setPixelArray(result);**

**}**

**Problem 5: Teacher**

**// solution 1**

**private HashMap<String, String> teacher1(HashMap<String, Integer>**

**students, HashMap<Integer, String> gradeMap) {**

**HashMap<String, String> result = new HashMap<>();**

**for (String student : students.keySet()) {**

**int studentPercent = students.get(student);**

**// figure out this student's grade from the mapping**

**String grade = "F";**

**for (int percent : gradeMap.keySet()) {**

**if (studentPercent >= percent) {**

**grade = gradeMap.get(percent);**

**}**

**}**

**if (gradeMap.size() > 0) {**

**result.put(student, grade);**

**}**

**}**

**return result;**

**}**

**// solution 2: count down**

**private HashMap<String, String> teacher2(HashMap<String, Integer>**

**students, HashMap<Integer, String> gradeMap) {**

**// find minimum value in grade map**

**int min = Integer.MAX\_VALUE;**

**for (int pct : gradeMap.keySet()) {**

**min = Math.min(min, pct);**

**}**

**HashMap<String, String> result = new HashMap<>();**

**if (gradeMap.size() > 0) {**

**for (String student : students.keySet()) {**

**int pct = students.get(student);**

**if (pct < min) {**

**result.put(student, "F");**

**} else {**

**// go down by 1% at a time until we find a % in the grade mapping**

**while (!gradeMap.containsKey(pct)) {**

**pct--;**

**}**

**result.put(student, gradeMap.get(pct));**

**}**

**}**

**}**

**return result;**

**}**

**// solution 3: gradeMap as outer loop**

**private HashMap<String, String> teacher3(HashMap<String, Integer>**

**students, HashMap<Integer, String> gradeMap) {**

**HashMap<String, String> result = new HashMap<>();**

**for (int pct : gradeMap.keySet()) {**

**String grade = gradeMap.get(pct);**

**for (String student : students.keySet()) {**

**int studentPct = students.get(student);**

**if (studentPct >= pct) {**

**result.put(student, grade);**

**} else if (!result.containsKey(student)) {**

**result.put(student, "F");**

**}**

**}**

**}**

**return result;**

**}**

**Problem 6: Let’s Go For A Drive (Part 1)**

**public class Car {**

**// The number of miles the car has been driven**

**private int mileage;**

**// The amount of gas left in the tank**

**private double gasLeft;**

**// Whether the car is broken down**

**private boolean isBroken;**

**// This constructor initializes a car with some gas and mileage.**

**public Car(int initialGasVolume, int existingMileage) {**

**gasLeft = initialGasVolume;**

**mileage = existingMileage;**

**isBroken = false;**

**}**

**public boolean turnOnAndDrive(int milestToDrive) {**

**double breakdownChance = (mileage / 10000) \* 0.1;**

**if (RandomGenerator.getInstance().nextBoolean(breakdownChance)) {**

**// If we break down...**

**isBroken = true;**

**return false;**

**} else if (gasLeft \* MILES\_PER\_GALLON >= milestToDrive) {**

**// If we have enough gas for the full trip...**

**mileage += milestToDrive;**

**gasLeft -= ((double)milestToDrive / MILES\_PER\_GALLON);**

**return true;**

**} else {**

**// We don't have enough gas**

**mileage += gasLeft \* MILES\_PER\_GALLON;**

**gasLeft = 0;**

**return false;**

**}**

**}**

**// This method returns the number of miles this car has driven.**

**public int getMileage() {**

**return mileage;**

**}**

**// This method returns whether the car is broken down or not.**

**public boolean isBrokenDown() {**

**return isBroken;**

**}**

**// This method sets the car as no longer broken down.**

**public void repair() {**

**isBroken = false;**

**}**

**// This method adds the given number of gallons of gas to the tank.**

**public void fillGas(int numberOfGallons) {**

**gasLeft += numberOfGallons;**

**}**

**}**

**Problem 6: Let’s Go For A Drive (Part 1)**

**private double testCar() {**

**Car c = new Car(10, 0);**

**while (!c.isBrokenDown()) {**

**if (!c.turnOnAndDrive(10)) {**

**c.fillGas(10);**

**}**

**}**

**return c.getMileage();**

**}**

**Problem 7: KooshBall**

**public class KooshBall extends GraphicsProgram {**

**// The most recent line added**

**private GLine lastLine;**

**private JTextField colorField;**

**public void init() {**

**add(new JLabel("Color: "), SOUTH);**

**colorField = new JTextField(16);**

**add(colorField, SOUTH);**

**colorField.setActionCommand("Add");**

**colorField.addActionListener(this);**

**add(new JButton("Add"), SOUTH);**

**add(new JButton("Remove Last"), SOUTH);**

**addActionListeners();**

**}**

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

**if (e.getActionCommand().equals("Add")) {**

**Color c = colorMap.get(colorField.getText().toLowerCase());**

**// If the color is in the map, add a new line**

**if (c != null) {**

**RandomGenerator rgen = RandomGenerator.getInstance();**

**int randomX = rgen.nextInt(getWidth());**

**int randomY = rgen.nextInt(getHeight());**

**lastLine = new GLine(randomX, randomY, getWidth() / 2,**

**getHeight() / 2);**

**lastLine.setColor(c);**

**add(lastLine);**

**}**

**} else {**

**if (lastLine != null) {**

**remove(lastLine);**

**lastLine = null;**

**}**

**}**

**}**

**}**