/** Original Paper_Doll_Project_2_22_13.pde file was converted into a PDF on 2016-11-30 for ingest into DigitalGeorgetown.*/

/* OpenProcessing Tweak of *@*http://www.openprocessing.org/sketch/79178*@* */

/* !do not delete the line above, required for linking your tweak if you re-upload */

//Paper Dolls - Alternate Version - now with OOP power

//I consulted the following while writing this code:


//any numbers needed

float placeX, placeY; //used in randomizing the placement of dolls during the instances screen

int instanceNum = 199; // max number of instances during the instances screen - the higher the number, the more memory required, but the more fun you can have mashing the make one button

//declare our objects

Doll a, b, c, methodMan; // declare our object

Doll[] random = new Doll[instanceNum]; //this array will create randomly placed instances to fill the screen during the instantiation

//declare the types of screens we will have

boolean intro, define, defineTwo, instances, arguments, methods; // each of these arguments (and corresponding screens) represents a lesson

Screen text;

boolean[] instanceExists = new boolean[instanceNum];

//declares any buttons that will appear on the screen

Button letsMoveOn, makeInstance;
//these arrays will let you make various dance moves
int danceNum = 3;
String[] danceText = new String[danceNum];
Button[] danceMoves = new Button[danceNum];

//these arrays will let you review previous material
int reviewScreens = 4;
String[] reviewType = new String[reviewScreens];
Button[] reviewButton = new Button[reviewScreens];

//define any fonts
PFont bigFont, smallFont;

//These two strings determine the shape of your representation.
//Choose as you will - feel free to change the name
String male = "Alan";
String female = "Ada";
String moveOn = "Continue";
String instance = "Make One";

//declare any other variables
float mashCounter; //this counter will help us go through the different flags

//global variables for the variable screen

void setup() {
    //size of screen
size(800, 600);

// smooths out lines
smooth();

// provides the color of the background
background(240);

// sets up any fonts
bigFont = createFont("Arial", 36, true);
m
smallFont = createFont("Arial", 24, true);

// assign variables for our object - change the variables to change the basic look of the doll & where it is positioned
// use male or female to determine gender and name - anything else will confuse the code.

a = new Doll(female, #B92525, width/2, height/2, 100, 100);
b = new Doll(female, #236FE5, width/2, height/2, 100, 100);
c = new Doll(female, #236FE5, width/2, height/2, 100, 100);
methodMan = new Doll(female, #B92525, width/2, height/2, 100, 100);
text = new Screen(female, bigFont, smallFont);

letsMoveOn = new Button (width-100, height-100, 100, 50, #7B64FF, moveOn);

makeInstance = new Button (width*.5, height*.6, 120, 120, #AE2EFF, instance);

// everything related the methods/dance screen

danceText[0] = "To the Left";
danceText[1] = "To the Right";
danceMoves[0] = new Button (width*.2, height*.53, 180, 70, #AE2EFF, danceText[0]);
danceMoves[1] = new Button (width*.2, height*.73, 180, 70, #AE2EFF, danceText[1]);

// initializing button color

danceMoves[0].setupColor();
danceMoves[1].setupColor();

danceMoves[2].setupColor();
danceMoves[2].setColor();

makeInstance.setColor();

// setting up review buttons & initializing color

reviewType[0] = "Introduction";
reviewType[1] = "Instances";
reviewType[2] = "Variables";
reviewType[3] = "Methods";

reviewButton[0] = new Button (width*.35, height*.5, 180, 120, #AE2EFF, reviewType[0]);
reviewButton[1] = new Button (width*.35, height*.75, 180, 120, #AE2EFF, reviewType[1]);
reviewButton[2] = new Button (width*.65, height*.5, 180, 120, #AE2EFF, reviewType[2]);
reviewButton[3] = new Button (width*.65, height*.75, 180, 120, #AE2EFF, reviewType[3]);

reviewButton[0].setColor();
reviewButton[1].setColor();
reviewButton[2].setColor();
reviewButton[3].setColor();

// setting up variables so that they are given proper values (meaning they only are valued as this once and can be changed later using methods)

a.setupDoll();
b.setupDoll();
c.setupDoll();
a.waveSetup();
a.variableSetup();
}

void draw() {

background(240);

// letsMoveOn.buttonDraw();
//draw the doll and any text screens relating to the doll
if (mashCounter==0) { //this draws the intro screen
  text.writeIntro();
pushMatrix();
  scale(.5);
  translate(width/2, -height/4);
a.wave();
popMatrix();
}
if (mashCounter==1) { //this draws the first screen defining OOP
  text.defineOOP1();
pushMatrix();
  translate(-width/4, -height/8);
a.wave();
popMatrix();
}
if (mashCounter==2) {
pushMatrix();
  translate(-width/4, -height/8);
a.wave();
popMatrix();
text.defineOOP2();
}
if (mashCounter ==3) { //in this screen a button lets you create an unlimited number of dolls by pressing the create button, graphically explaining instances
  text.instances();
  makeInstance.buttonDraw();
  for (int i = 1; i<makeInstance.mashCounter(); i = i+1 ) {

random[i].make();

makeInstance.buttonDraw();

if (mashCounter == 4) { // this screen is for understanding variables
    makeInstance.reset();
    pushMatrix();
    scale(.75, .75);
    translate(width*.55, height*.15);
    b.make();
    popMatrix();
    pushMatrix();
    scale(.75, .75);
    translate(width*-.25, height*.15);
    c.make();
    popMatrix();
    a.drawVariables();
    text.variables();
}

if (mashCounter == 5) { // this screen is for understanding methods
    text.methods();
    println(danceMoves[0].mashCounter() + " and " + danceMoves[1].mashCounter() + " and " + danceMoves[2].mashCounter());
    if (danceMoves[0].mashCounter() % 2 == 0) {
        if (danceMoves[1].mashCounter() % 2 == 0) {
            if (danceMoves[2].mashCounter() % 2 == 0) {
                danceMoves[0].buttonDraw();
            }
        }
    }
}
danceMoves[1].buttonDraw();
danceMoves[2].buttonDraw();
pushMatrix();

scale(.6, .6);
translate(width*.52, height*.49);
//methodMan.clearChanges();
methodMan.make();

popMatrix();

danceMoves[0].bumpCounter();
danceMoves[1].bumpCounter();
danceMoves[2].bumpCounter();

methodMan.dancingIsForbidden();
}
}
}

if (danceMoves[0].mashCounter() > 1 && danceMoves[0].mashCounter() % 2 == 1) {
    pushMatrix();
    scale(.6, .6);
    translate(width*.52, height*.49);
    methodMan.toTheLeft();
    popMatrix();

danceMoves[0].turnOn();
danceMoves[1].buttonDraw();
danceMoves[2].buttonDraw();
danceMoves[1].putBabyInACorner();
danceMoves[2].putBabyInACorner();
}
else if (danceMoves[1].mashCounter() > 1 && danceMoves[1].mashCounter() %2 ==1) {
    pushMatrix();
    scale(.6, .6);
    translate(width*.52, height*.49);
    methodMan.toTheRight();
    popMatrix();
    danceMoves[1].turnOn();
    danceMoves[0].buttonDraw();
    danceMoves[2].buttonDraw();
    danceMoves[0].putBabyInACorner();
    danceMoves[2].putBabyInACorner();
}
else if (danceMoves[2].mashCounter() > 1 && danceMoves[2].mashCounter() %2 ==1) {
    pushMatrix();
    scale(.6, .6);
    translate(width*.52, height*.49);
    methodMan.nowKick();
    popMatrix();
    danceMoves[2].turnOn();
    danceMoves[0].buttonDraw();
    danceMoves[1].buttonDraw();
    danceMoves[1].putBabyInACorner();
    danceMoves[0].putBabyInACorner();
}
}

if ( mashCounter == 6) { //this is the final screen, just providing a table of contents and ways to go back to particular
    text.finishHim();
reviewButton[0].buttonDraw();
reviewButton[1].buttonDraw();
reviewButton[2].buttonDraw();
reviewButton[3].buttonDraw();

if (reviewButton[0].mashCounter()>=1) { //this button takes you back to the beginning
    mashCounter = 0;
    reviewButton[0].reset();
    reviewButton[1].reset();
    reviewButton[2].reset();
    reviewButton[3].reset();
}

if (reviewButton[1].mashCounter()>=1) { //this returns to instances screen
    mashCounter=3;
    reviewButton[0].reset();
    reviewButton[1].reset();
    reviewButton[2].reset();
    reviewButton[3].reset();
}

if (reviewButton[2].mashCounter()>=1) { //this returns to variables screen
    mashCounter=4;
    reviewButton[0].reset();
    reviewButton[1].reset();
    reviewButton[2].reset();
    reviewButton[3].reset();
}

if (reviewButton[3].mashCounter()>=1) { //this returns to the methods screen
    mashCounter = 5;
reviewButton[0].reset();
reviewButton[1].reset();
reviewButton[2].reset();
reviewButton[3].reset();

if (mashCounter>6) {//this should take you back to the contents screen if you continue to press M
    mashCounter = 6;
    reviewButton[0].reset();
    reviewButton[1].reset();
    reviewButton[2].reset();
    reviewButton[3].reset();
}

//press M and you continue in the program
void keyReleased() {
    if ((key == 'm' || key == 'M')) {
        mashCounter++;
    }
}

void mouseReleased() {
    makeInstance.buttonMash();
    if (mashCounter==3) {
        instanceExists[makeInstance.mashCounter()] = true;
        randomizePlacement();
        random[makeInstance.mashCounter()] = new Doll (female, #B92525, placeX, placeY, 100, 100);
        random[makeInstance.mashCounter()].setupDoll();
if (mashCounter==5) {
    //a.dancingIsForbidden();
    // danceMoves[0].putBabyInACorner();
    // danceMoves[1].putBabyInACorner();
    //danceMoves[2].putBabyInACorner();
}
if (mashCounter==6) {

    if (mouseX<width*.5) {

        if (mouseY<height*.65) {

            reviewButton[0].buttonMash();
        }
    }
}
else {
    reviewButton[1].buttonMash();
}
else {
    if (mouseY<height*.65) {
        reviewButton[2].buttonMash();
    }
    else {
        reviewButton[3].buttonMash();
    }
}
void randomizePlacement() {
    placeX = random(0, width);
    placeY = random(height/2, height);
}

void mouseDragged() {
    if (mashCounter == 4) {
        if (mousePressed) {
            // waist joint
            //if (mouseX >= a.waistX() - a.pressurePoint() && mouseX <= a.waistX() + a.pressurePoint()) {
            //  if (mouseY >= a.waistY() - a.pressurePoint() && mouseY <= a.waistY() + a.pressurePoint()) {

                // a.dragWaist();
                //}
            //}
            // head - x values - manipulate the width of the head
            if (mouseX >= a.headWidthPointX() - a.pressurePointRad && mouseX <= a.headWidthPointX() + a.pressurePointRad) {
                if (mouseY >= a.headWidthPointY() - a.pressurePointRad && mouseY <= a.headWidthPointY() + a.pressurePointRad) {
                    headSizeX = brainX + headSizeX*.5 - mouseX;
                    headWidthPointX = mouseX;
                //}
            //}
            // head - y values - manipulate the width of the head
            if (mouseX >= a.headLengthX() - a.pressurePoint() && mouseX <= a.headLengthX() + a.pressurePoint()) {
                if (mouseY >= a.headLengthY() - a.pressurePoint() && mouseY <= a.headLengthY() + a.pressurePoint()) {
                    a.dragChin();
                }
if (mouseX>=a.headWidthX()-a.pressurePoint() && mouseX<=a.headWidthX()+a.pressurePoint()) {
    if (mouseY>=a.headWidthY()-a.pressurePoint() && mouseY<=a.headWidthY()+a.pressurePoint()) {
        a.dragEar();
    }
}

if (mouseY>=a.waistY()-a.pressurePoint() && mouseY<=a.waistY()+a.pressurePoint()) {
    a.dragWaist();
}

void mousePressed() { 
    if (mashCounter == 5) {
        if (mouseY<height*.53+70) {
            danceMoves[0].buttonMash();
        }
        if (mouseY>height*.73-70 && mouseY<height*.73+70) {
            danceMoves[1].buttonMash();
        }
        if (mouseY>height*.93-70) {
            danceMoves[2].buttonMash();
        }
    }
}