CS106A Midterm Review

Fall 2013

With problems adapted from Eric Roberts' CS106A Midterm Review Deck 2010

Topics/Types of problems

- Karel
- Expression Evaluation
- Code tracing
- Simple Java Programs
- Graphics
- String manipulation
- Heap/Stack

Order of Operations

()

* , / , %

+ , -

Operators in same precedence category evaluated left to right.

Order of Comparators

```
!           “not”
!p          “not” p


&&          and
p && q      only true if p and q are true


||          or
p || q      true if p or q or both are true


Other value comparisons: ==  !=  >  <  >=  <=
```

Evaluating Expressions

```
5.0 / 4 - 4 / 5
```

```
7 < 9 - 5 && 3 % 0 == 3
```

```
"B" + 8 + 4
```

```
"E" - "A"
```

Remember these from lecture?

Short circuit evaluations:

```
p = (x != 0) && ((y / x) == 0);
```

```
p = (5 > 3) || (4 <= 2);
```

Code Tracing Strategies

```
void run(){
    for (int x=5; x<9; x++){
        int y = mystery1(x*x, x%3);
        println("x = " + x + ", y = " + y);
    }
}

int mystery1(int x, int y){
    if((x%2)==0){
        return (y + mystery2(x,x+y));
    }

    return (x + mystery2(x+y, y));
}

int mystery2(int x, int y){
    if(x<y){
        return (y-x+5);
    }

    return ((x-y) % 4);
}
```

Code Tracing Solution

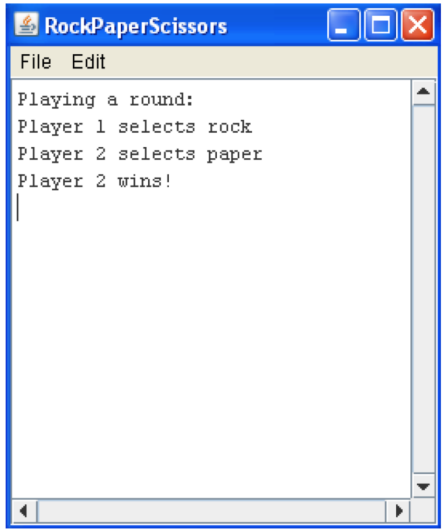
```
x = 5, y = 26

x = 6, y = 0

x = 7, y = 50

x = 8, y = 9
```

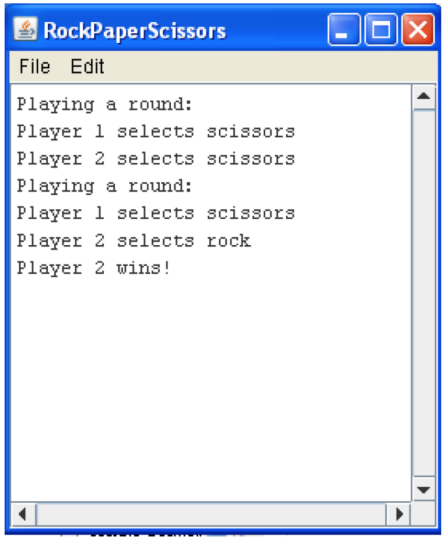
Simple Java: Rock Paper Scissors

For this program you will play a game of "Rock Paper Scissors". The program will play a game and print the winner. In case of a tie, another turn is played, until one of the players has won. You can assume you have a method that returns the String corresponding to the numeric constant for a selection. The method looks as shown below:

```
private String resultString(int play){}
```

So,
resultingString(ROCK)will return "rock"





Here is the starter code:

```
public class RockPaperScissors extends C

    private static final int ROCK = 0;
    private static final int PAPER = 1;
    private static final int SCISSORS = 2;

    public void run() {

    }
}
```

Simple Java: RPS Solution

```
/* Determine selections for each player and then return winner (or
tie) */

    private int playRound() {
        println("Playing a round:");
        int player1 = rgen.nextInt(ROCK, SCISSORS);
        println("Player 1 selects " + resultString(player1));
        int player2 = rgen.nextInt(ROCK, SCISSORS);
        println("Player 2 selects " + resultString(player2));
        return (determineWinner(player1, player2));
    }

/* Return string corresponding the numeric constant for a selection.
*/
    private String resultString(int play) {
        switch (play) {
            case ROCK: return "rock";
            case PAPER: return "paper";
            default: return "scissors";
        }
    }
```

Simple Java: RPS Solution

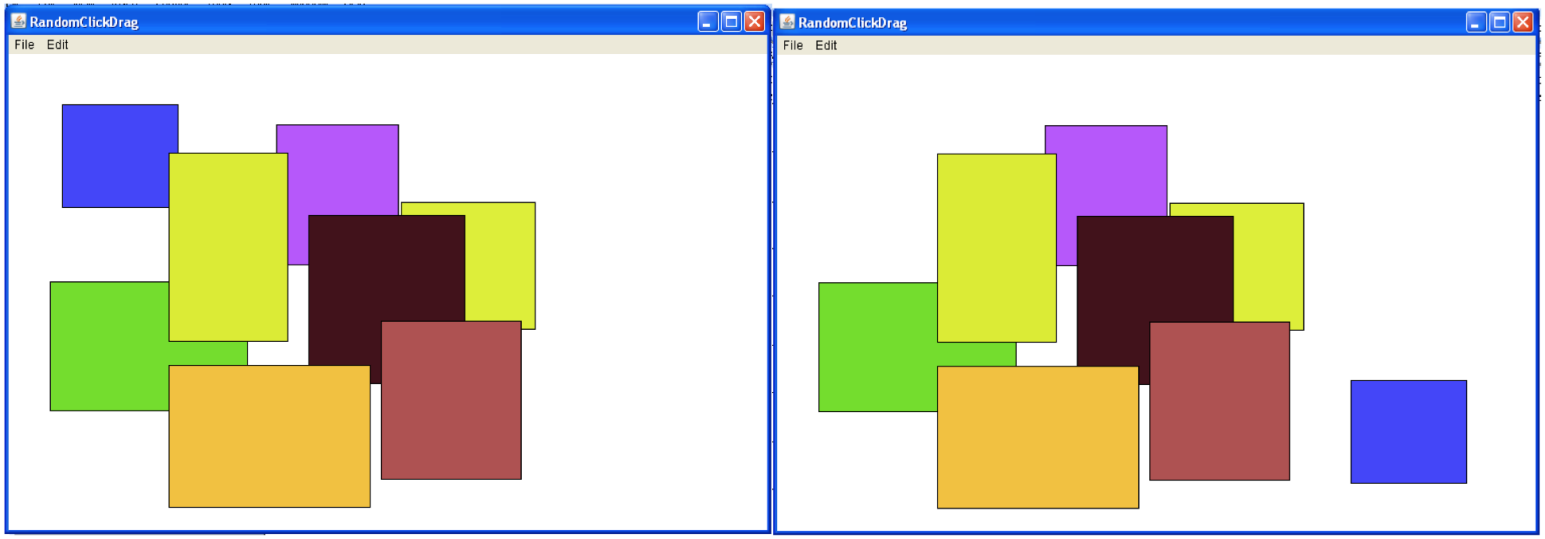
```
/* Return number of winning player (1 or 2), or 0 to indicate a tie.
*/
    private int determineWinner(int player1, int player2) {
        if (player1 == player2) {
            return 0;
        /* Player 1 wins if selection is 1 less than player 2's selection,
         * assuming that we "wrap around" given the 3 values we have.
         */
        } else if (player1 == (player2 + 1) % 3) {
            return 1;
        } else {
            return 2;
        }
    }
```

Simple Java: RPS Solution

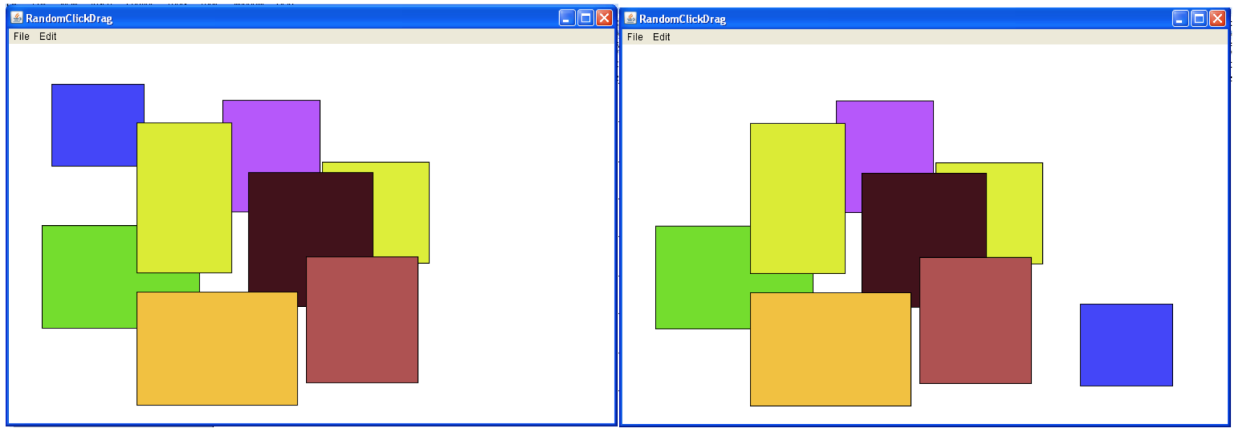
```
public class RockPaperScissors extends ConsoleProgram {

    private RandomGenerator rgen = new RandomGenerator();
    private static final int ROCK = 0;
    private static final int PAPER = 1;
    private static final int SCISSORS = 2;

     /* Keep playing rounds until a winner is determined. */

    public void run() {
        int winner = 0; // No winner to begin with.
        while (winner == 0) {
                winner = playRound();
          }
        println("Player " + winner + " wins!");
    }
```

Graphics: Random Squares

In this problem you will write a program that draws GRects of random sizes on the screen at the click of the mouse and allows the user to drag them around the canvas. The GRects can change colors and be dragged around the screen. For example, after a few clicks, your canvas will look as shown below at the left. However, if you press the mouse down on top of the square at the top left, hold, and move it to the bottom right corner, the square will move and the image will now look as shown below at the right.



Graphics: Random Squares



- Length and width and initial color are randomly chosen
- Clicking on an existing piece will simply choose a new color for that GRect
- Size is bounded by MAX_LEN and MIN_LEN

Here is the starter code:

```
public class RandomClickDrag extends GraphicsProgram{
     private static final double MAX_LEN = 200;
     private static final double MIN_LEN = 100;
     private RandomGenerator rgen = new RandomGenerator();
     public void run(){
          //Your code here

     }
}
```

Graphics: RandSq Solution

```
public void mouseClicked(MouseEvent e) {
    currentRect = getElementAt(e.getX(), e.getY());
    if(currentRect != null) {
        ((GRect)currentRect).setFillColor(rgen.nextColor());
    } else{
        GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
                               rgen.nextDouble(MIN_LEN, MAX_LEN));
        rect.setLocation(e.getX() - (rect.getWidth() / 2),
                         e.getY() - (rect.getHeight() / 2));
        rect.setFilled(true);
        rect.setFillColor(rgen.nextColor());
        add(rect);
    }
}

private static final double MAX_LEN = 200;
private static final double MIN_LEN = 100;
private GObject currentRect;
private RandomGenerator rgen = new RandomGenerator();
```

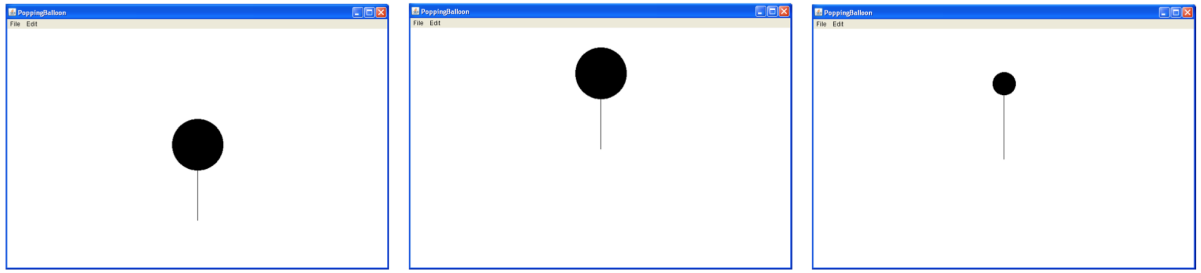
Graphics: RandSq Solution

```
public void mousePressed(MouseEvent e) {
    lastX = e.getX();
    lastY = e.getY();
    currentRect = getElementAt(e.getX(), e.getY());
}


public void mouseDragged(MouseEvent e) {
    if(currentRect != null) {
        currentRect.move(e.getX() - lastX, e.getY() - lastY);
        lastX = e.getX();
        lastY = e.getY();
    }
}

public void run(){
    addMouseListeners();
}

private double lastX;
private double lastY;
private GObject currentRect;
```

Graphics: Popping Balloon

For this problem you will be making a balloon that slowly moves up. When you click on the balloon, it will slowly start deflating and disappear along with the string holding onto it. The balloon should start at the center of the screen. The images below show:

(1) balloon at start
(2) balloon after a few seconds
(3) balloon deflating



Here is the starter code:

```
public class PoppingBalloon extends GraphicsProgram {

	private static final int BALLOON_WIDTH = 100;
	private static final int TAIL_LENGTH = 150;
	private static final int FLOAT_SPEED = 1;
	private static final int SHRINK_SPEED = 2;
	private static final int PAUSE_TIME = 30;

	public void run() { //Your code here

	}
}
```

Graphics: Balloon Solution

```
private static final int BALLOON_WIDTH = 100;
private static final int TAIL_LENGTH = 150;
private static final int FLOAT_SPEED = 1;
private static final int SHRINK_SPEED = 2;
private static final int PAUSE_TIME = 30;

private GOval balloon;
private GLine tail;
private int dy = 0;
private int deflation = 0;

public void mouseClicked(MouseEvent e) {
    if(getElementAt(e.getX(), e.getY()) != null) {
        dy = -FLOAT_SPEED;
        deflation = SHRINK_SPEED;
    }
}
```

Graphics: Balloon Solution

```
private void animateBalloon() {
    while(balloon.getWidth() > 0) {
        balloon.move(0, dy);
        tail.move(0, dy);
         balloon.setSize(balloon.getWidth() - deflation,
                         balloon.getHeight() - deflation);
        //Now adjust balloon after click
        balloon.move(deflation / 2.0, deflation);
        pause(PAUSE_TIME);
    }
    remove(balloon);
    remove(tail);
}

/* Reference:
 * (after mouse clicked, the following have these values)
 * dy = -FLOAT_SPEED;
 * deflation = SHRINK_SPEED;
 * (before mouse clicked, both are 0)
 */
```

Graphics: Balloon Solution

```
private static final int BALLOON_WIDTH = 100;
private static final int TAIL_LENGTH = 150;

private GOval balloon;
private GLine tail;

public void run() {
    int cx = getWidth()/2;
    int cy = getHeight()/2;
    balloon = new GOval(cx - (BALLOON_WIDTH / 2),
                cy - (BALLOON_WIDTH / 2),
                BALLOON_WIDTH, BALLOON_WIDTH);
    balloon.setFilled(true);
    add(balloon);
    tail = new GLine(cx, cy + (BALLOON_WIDTH / 2),
                cx, cy + (BALLOON_WIDTH / 2) + TAIL_LENGTH);

            add(tail);
            addMouseListeners();
    animateBalloon();
}
```

String Manipulation: Making Palindromes

```
/* This method will make a word into a palindrome by changing any
 * different letters between the two ends of the word. It will
 * print the number of letters changed to make the palindrome.
 */
public String makePalindrome(String str) {









}
```

String Manipulation: Making Palindromes

```
/* This method will make a word into a palindrome by changing any
 * different letters between the two ends of the word. It will
 * print the number of letters changed to make the palindrome.
 */
public String makePalindrome(String str) {
    int count = 0;
    int len = str.length();
    String firstHalf = "", secondHalf = "";
    for(int i = 0; i < len/2; i++) {
        if (str.charAt(i) != str.charAt(len-(i+1))) {
            count++;
        }

        firstHalf += str.charAt(i);
        secondHalf = str.charAt(i) + secondHalf;
    }

    println("The string " + str + " is " + count +
            " letters away from being a palindrome.");

    if ((len%2)!=0) return (firstHalf + str.charAt(len/2) +
                                secondHalf);
    return (firstHalf + secondHalf);
}
```

The End :)

Good luck on the exam!