Mehran Sahami
CS 106A

Handout #32
November 5, 2007

Solutions to Midterm Exam

Problem 1: Karel the Robot (15 points)

```
/*
 * File: TreasureHuntKarel.java
 * ----------------------------
 * This program has Karel run a treasure hunt.
 */

import stanford.karel.*;

public class TreasureHuntKarel extends SuperKarel {

   public void run() {
      /* Continue treasure hunt until we face wall (by treasure).  We
       * are guaranteed not to encounter a wall until we reach treasure.
       */
      while (frontIsClear()) {
         faceCorrectDirection();
         moveToNextPile();
      }
   }

   /* Turns Karel until he is facing East */
   private void faceEast() {
      while (notFacingEast()) {
         turnLeft();
      }
   }

   /* To face the correct direction based on a clue represented by a
    * pile of beepers, we first face East and then make one left turn
    * for each beeper in the pile.
    */
   private void faceCorrectDirection() {
      faceEast();
      while (beepersPresent()) {
         pickBeeper();
         turnLeft();
      }
   }

   /* Move forward until you reach next clue or treasure (pile of beepers) */
   private void moveToNextPile() {
      while (noBeepersPresent()) {
         move();
      }
   }
}
```

Problem 2: Simple Java expressions, statements, and methods (15 points)

(2a)

`7 / ('C' - 'A')`	`3`
`4 > 5 || 3 % 1 == 0 && 6 * 3 > 19`	`false`
`('b' - 'a') + "a"`	`"1a"`

(2b) What output is printed by the following program:

```
First result: 19
Second result: 4
```

Problem 3: Simple Java program using the random number library (20 points)

```
/*
 * File: CoinToss.java
 * -------------------
 * Number of times to toss a coin until it comes up heads 3 times in a row.
 */

import acm.program.*;
import acm.util.*;

public class CoinToss extends ConsoleProgram {

   private static final int NUM_CONSECUTIVE = 3;

   public void run() {
      int count = 0;
      int totalFlips = 0;
      while (count < NUM_CONSECUTIVE) {
         boolean isHeads = rgen.nextBoolean();  // value true means "heads"
         if (isHeads) {
            println("heads");       // print result of coin flip
            count++;                // increment count of consecutive heads
         } else {
            println("tails");       // print result of coin flip
            count = 0;              // reset count of consecutive heads
         }
         totalFlips++;              // increment total number of flips
      }
      println("It took " + totalFlips + " flips to get "+ NUM_CONSECUTIVE
            + " consecutive heads");
   }

   private RandomGenerator rgen = RandomGenerator.getInstance();
}
```

Problem 4: Using the graphics libraries (25 points)

Although there are a number of ways to solve this problem, there are two common approaches, the first of which would be considered stylistically nicer than the other. The stylistically nicer solution is to keep track of the selected object at all times. When an object is selected, the previous selected object is unselected and the new selection is recorded. This is shown below:

```
/* File: RadioButtonsProgram.java */

import acm.graphics.*;
import acm.program.*;
import java.awt.*;
import java.awt.event.*;

public class RadioButtonsProgram extends GraphicsProgram {

   private static final double DIAM = 100;
   private static final double SPACER = 120;

   public void run() {
      drawInitialCircles();
      addMouseListeners();
   }

   // Draw three circles centered in the graphics window.
   private void drawInitialCircles() {
      double cy = (getHeight() - DIAM) / 2;
      double startx = (getWidth() - ((DIAM * 3) + (SPACER * 2))) / 2;

      GOval oval1 = new GOval(startx, cy, DIAM, DIAM);
      oval1.setFilled(true);
      add(oval1);

      GOval oval2 = new GOval(startx + DIAM + SPACER, cy, DIAM, DIAM);
      oval2.setFilled(true);
      add(oval2);

      GOval oval3 = new GOval(startx + ((DIAM + SPACER) * 2), cy, DIAM,DIAM);
      oval3.setFilled(true);
      add(oval3);
   }

   // If use clicked on a object unselect current selection (if one exists)
   // and keep track of the new selection.
   public void mouseClicked(MouseEvent e) {
      GObject obj = getElementAt(e.getX(), e.getY());
      if (obj != null) {
         if (selected != null) {
            selected.setColor(Color.BLACK);
         }
         selected = obj;
         selected.setColor(Color.RED);
      }
   }

   private GObject selected = null; // Used to keep track of selected circle
}
```

Another possible solution keeps track of all three circles as instance variables. Whenever a circle is clicked, it unselects *all* the circles and then selects the newly selected circle. This code is stylistically less nice than the previous example, and is much less efficient (imagine if the user could select among one of 20 or 100 circles), but it still works for the problem we gave you.

```
/* File: RadioButtonsProgram.java */

import acm.graphics.*;
import acm.program.*;
import java.awt.*;
import java.awt.event.*;

public class AnotherCircleSelect extends GraphicsProgram {

   private static final double DIAM = 100;
   private static final double SPACER = 120;

   public void run() {
      drawInitialCircles();
      addMouseListeners();
   }

   // Draw three circles centered in the graphics window.
   private void drawInitialCircles() {
      double cy = (getHeight() - DIAM) / 2;
      double startx = (getWidth() - ((DIAM * 3) + (SPACER * 2))) / 2;

      oval1 = new GOval(startx, cy, DIAM, DIAM);
      oval1.setFilled(true);
      add(oval1);

      oval2 = new GOval(startx + DIAM + SPACER, cy, DIAM, DIAM);
      oval2.setFilled(true);
      add(oval2);

      oval3 = new GOval(startx + ((DIAM + SPACER) * 2), cy, DIAM, DIAM);
      oval3.setFilled(true);
      add(oval3);
   }

   // If user clicks a circle, then unselect all the circles and select
   // the newly clicked circle.
   public void mouseClicked(MouseEvent e) {
      GObject obj = getElementAt(e.getX(), e.getY());
      if (obj != null) {
         oval1.setColor(Color.BLACK);
         oval2.setColor(Color.BLACK);
         oval3.setColor(Color.BLACK);
         obj.setColor(Color.RED);
      }
   }

   private GOval oval1; // Instance variables used to keep track of all
   private GOval oval2; // three circles that a user could select.
   private GOval oval3;
}
```

Problem 5: Strings and characters (15 points)

```
// Returns true is ch is an end-of-sentence punctuation character
private boolean isPunctuation(char ch) {
   return (ch == '.' || ch == '!' || ch == '?');
}

private String cleanUpPunctuation(String str) {
   String result = "";

   // Keep track of whether we are currently in a sequence of
   // punctuation marks.
   boolean droppingPunctuation = false;
   for (int i = 0; i < str.length(); i++) {
      char ch = str.charAt(i);

      // If we find punctuation, then keep the character is we are
      // not already in a punctuation sequence.  In any case, we
      // are now in a punctuation sequence.
      if (isPunctuation(ch)) {
         if (!droppingPunctuation) {
            result += ch;
            droppingPunctuation = true;
         }
      } else {
      // If character is not punctuation, then we keep it and we cannot
      // be in a punctuation sequence.
         result += ch;
         droppingPunctuation = false;
      }
   }
   return result;
}
```