Mehran Sahami
CS 106A

Handout #33
November 5, 2018

# Solutions to Midterm Exam

**Problem 1: Karel the Robot**

```
/* File: MountainKarel.java */

import stanford.karel.*;

public class MountainKarel extends SuperKarel {

   public void run() {
      goToMountain();
      climbMountain();
      repositionOnMountainTop();
      putBeeper();      // plant flag (beeper) at top of mountain
   }

   /* Mountain is between avenues 4 and 5, so check first 4 corners for beeper. */
   private void goToMountain() {
      for(int i = 0; i < 3; i++) {
         if (beepersPresent()) {
            pickBeeper();
         }
         move();
      }
      if (beepersPresent()) {
         pickBeeper();
      }
   }

   /* The mountain is a series of steps, so to climb mountain we need to keep
    * climbing steps until we get to the top of the mountain (front is clear).  */
   private void climbMountain() {
      while (frontIsBlocked()) {
         climbStep();
      }
   }

   /* Karel climbs the step that it is currently facing and moves to the base
    * (facing East) of the next step (at most two avenues wide).                */
   private void climbStep() {
      turnLeft();
      while (rightIsBlocked()) {   // ascend the step
         move();
      }
      turnRight();                 // face the top of the step
      move();                      // traverse top of step (up to 2 avenues)
      if (frontIsClear()) {
         move();
      }
   }

   /* When traversing peak of the mountain, we may have moved one step too far
    * (if mountain peak was one avenue wide), so we need to back up one avenue. */
   private void repositionOnMountainTop() {
      if (rightIsClear()) {
         turnAround();
         move();
         turnAround();
      }
   }
}
```

**Problem 2: Simple Java expressions, statements, and methods**

**(2a)**

| | |
|---|---|
| `3 * ('d' - 'a') / 2` | **4** |
| `5 % 3 == 2 \|\| 3 == 7 / 2 && 2 + 2 < 3` | **true** |
| `'c' - 'a' + 1 + "A"` | **"3A"** |

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

```
First result: 14
Second result: 17
```

**Problem 3: Simple Java program**

There are several common solutions to this problem. Here, we present two. The first solution computers the total number of months to compute interest for and uses a single loop to count through the months.

Solution 1:

```
/* File: ComputeInterest.java                                              */
/* This program computes monthly compounded interest over time for a bank  */
/* account.                                                                */

import acm.program.*;

public class ComputeInterest extends ConsoleProgram {

   /* Monthly interest rate */
   private static final double INTEREST_RATE = 0.02;

   /* Number of months in a year */
   private static final int MONTHS_PER_YEAR = 12;


   public void run() {
      /* Read interest rate, start year/month, and end year/month from user.  */
      double balance = readDouble("Initial balance: ");
      int startYear = readInt("Start year: ");
      int startMonth = readInt("Start month: ");
      int endYear = readInt("End year: ");
      int endMonth = readInt("End month: ");

      /* Determine total number of months to compute interest for             */
      int totalMonths = ((endYear - startYear) * MONTHS_PER_YEAR)
                                              + (endMonth - startMonth);

      /* Variables to keep track of month and year as we iterate through loop */
      int year = startYear;
      int month = startMonth;

      for(int i = 0; i <= totalMonths; i++) {
         println("Year " + year + ", month " + month + " balance: " + balance);
         balance += balance * INTEREST_RATE;
         month++;

         /* If we reached end of the year, increment year and reset month     */
         if (month > MONTHS_PER_YEAR) {
            year++;
            month = 1;
         }
      }
}
```

The second solution has two loops, one for the year and one for the month. It needs to deal with the starting year and ending year as special cases when determining which months to loop through.

Solution 2:

```
/* File: ComputeInterest.java                                              */
/* This program computes monthly compounded interest over time for a bank  */
/* account.                                                                */

import acm.program.*;

public class ComputeInterest extends ConsoleProgram {

   /* Monthly interest rate */
   private static final double INTEREST_RATE = 0.02;

   /* Number of months in a year */
   private static final int MONTHS_PER_YEAR = 12;


   public void run() {
      /* Read interest rate, start year/month, and end year/month from user.  */
      double balance = readDouble("Initial balance: ");
      int startYear = readInt("Start year: ");
      int startMonth = readInt("Start month: ");
      int endYear = readInt("End year: ");
      int endMonth = readInt("End month: ");

      /* Make sure ending year/month is not earlier than starting end/month.  */
      if (!(endYear < startYear ||
                        (endYear == startYear && endMonth < startMonth))) {

         /* Iterate through the years */
         for (int year = startYear; year <= endYear; year++) {
            int currEndMonth = endMonth;

            /* If not in last year, then count through last month of the year */
            if (year != endYear) {
               currEndMonth = MONTHS_PER_YEAR;
            }

            /* If not in the start year, then start counting from first month */
            int currStartMonth = startMonth;
            if (year != startYear) {
               currStartMonth = 1;
            }

            /* Iterate through the months in that year                      */
            for (int month = currStartMonth; month <= currEndMonth; month++) {
               println("Year " + year + ", month " + month +
                                                    " balance: " + balance);
               balance += balance * INTEREST_RATE;
            }
         }
       }
    }
}
```

**Problem 4: Using the graphics libraries**

```
/* File: SquareAndCircle.java */

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

public class SquareAndCircle extends GraphicsProgram {

   // Size is both height and width of square and circle
   private static final int SIZE = 50;

   /* Put initial shape centered on the screen and listen for mouse events.   */
   public void run() {
      setup();
      addMouseListeners();
   }

   /* Create the square and circle, and display square centered on canvas.    */
   private void setup() {
      square = new GRect((getWidth() - SIZE) / 2, (getHeight() - SIZE) / 2,
                         SIZE, SIZE);
      square.setFilled(true);
      circle = new GOval(SIZE, SIZE);
      circle.setFilled(true);
      current = square;
      add(square);

   }

   /* Handle mouse click */
   public void mouseClicked(MouseEvent e) {
      int x = e.getX();
      int y = e.getY();
      GObject obj = getElementAt(x, y);
      if (obj == null) {            // User did not click on shape
         /* Move shape to be centered on mouse click.                        */
         current.setLocation(x - (SIZE / 2), y - (SIZE / 2));
      } else {                      // User did click on shape
         double objX = obj.getX();
         double objY = obj.getY();
         /* Remove old shape and place new (switched) shape in same location. */
         remove(current);
         if (current == square) {
            current = circle;
         } else {
            current = square;
         }
         add(current, objX, objY);
      }
   }

   /* private instance variables */
   private GRect square;
   private GOval circle;
   private GObject current;   // Shape (square or circle) currently displayed
}
```

**Problem 5: Strings and characters**

```
/* You can assume these packages have already been imported for you */
import acm.program.*;


   private boolean isValidWebsite(String str) {
      boolean isFirstLetter = true;

      for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);

         // if we at first letter of a token, make sure it is a letter
         if (isFirstLetter) {
            isFirstLetter = false;
            if (!Character.isLetter(ch)) return false;
         } else {
            // check to see if we ended a token
            if (ch == '.') {
               isFirstLetter = true;  // note: will be starting a new token
            } else {
               if (!Character.isLetterOrDigit(ch)) return false;
            }
         }
      }
      return true;
   }
```