Chris Piech
CS 106A

Handout #30
Feb 23, 2017

Midterm Solutions

Problem 1: Karel Want to Be a GRect (12 points)

Write a program where Karel fills in the rectangle between where Karel starts, the bottom wall and the left wall. When the program finishes running each "corner" in the rectangle should have one (and only one beeper). Here are two examples:

Solution:

```
public void run(){
   makeGuideWall();
   while(beepersPresent()) {
      move();
      placeBeeperRow();
      moveToNextRow();
   }
}

private void makeGuideWall() {
   turnRight();
   placeBeeperRow();
   turnRight();
}

private void moveToNextRow() {
   turnAround();
   while(beepersPresent()) {
      move();
   }
   turnAround();
   repositionForNextRow();
}

private void repositionForNextRow() {
   move();
   turnRight();
   move();
   turnLeft();
}

private void placeBeeperRow() {
   while(frontIsClear()) {
      putBeeper();
      move();
   }
   putBeeper();
}
```

Problem 2: Simple Java expressions, statements, and methods (12 points)

(2a) Compute the value, **and be explicit about the type**, of each of the following Java expressions. If an error occurs during any of these evaluations, write "Error" on that line and explain briefly why the error occurs.

`1 + 3 / 2 + 1.5`	3.5 (double)
`7 * 2 < 9 || !true`	False (boolean)
`6 + 8 + "4"`	"144" (String)

(2b) Write a run method which prints out all multiples between 1 and 99 (inclusive) in ascending order (eg 3, 6, 9, 12 etc up to 99). Print the numebers one per line.

Solution:

```
public class MultiplesOfThree extends ConsoleProgram {
   public void run() {
       for(int i = 3; i <= 99; i += 3) {
           println(i);
       }
   }
}
```

(2c) Trace the following program and draw the output

- x = 150
- y = 100
- width = 100.0
- height = 33.333333333333336 ☺
- color is blue and square is filled

Problem 3: Simulate Weather (12 points)

Write a program that can simulate a year of sun and rain at Stanford and count how many days within the year were sunny. At Stanford the likelyhood of it raining is accurately predcted based on whether it is winter and if it rained on the previous day.

Solution:

```
public void run(){
   println("Simulating a year of sun and rain at Stanford");
   println("-----");
   int sunnyDays = 0;
   boolean rainingToday = false;
   for(int i = 0; i < 365; i++) {
      double probability = getRainProbability(rainingToday, i);
      rainingToday = rg.nextBoolean(probability);
      printSunnyOrRainy(rainingToday);
      if(!rainingToday) {
         sunnyDays++;
      }
   }
   println("Number of sunny days: " + sunnyDays);
}

private void printSunnyOrRainy(boolean rainingToday) {
   if(rainingToday) {
      println("Rain");
   } else {
      println("Sun");
   }
}

private double getRainProbability(boolean rainingToday, int dayIndex) {
   if(dayIndex == 0) {
      return 0.5;
   } else if(dayIndex > 60) {
      if(rainingToday) return 0.50;
      return 0.05;
   } else {
      if(rainingToday) return 0.85;
      return 0.20;
   }
}
```

Problem 4: Mouse Magnet (12 points)

```
private double lastX = 0;
private double lastY = 0;

public void run(){
   GOval oval = makeCircle();
   addMouseListeners();
   while(true) {
      moveCircle(oval);
      pause(DELAY);
   }
}

private void moveCircle(GOval oval) {
   double centerX = oval.getX() + oval.getWidth()/2;
   double centerY = oval.getY() + oval.getHeight()/2;
   double dx = lastX - centerX;
   double dy = lastY - centerY;
   double length = distance(dx, dy);
   if(length > MOVE_AMT) {
      double coefficient = MOVE_AMT / length;
      double moveX = dx * coefficient;
      double moveY = dy * coefficient;
      oval.move(moveX, moveY);
   }
}

private double distance(double dx, double dy) {
   return Math.sqrt(dx * dx + dy * dy);
}

private GOval makeCircle() {
   GOval oval = new GOval(CIRCLE_SIZE, CIRCLE_SIZE);
   add(oval);
   oval.setFilled(true);
   return oval;
}

public void mouseMoved(MouseEvent e) {
   lastX = e.getX();
   lastY = e.getY();
}
```

Problem 5: Gene Editing (12 points)

Write a method **`crispr`** that simulates the cutting of a DNA string.

```
private String crispr(String strand, String guide) {
   String newDna = "";
   for(int i = 0; i < strand.length(); i++) {
      newDna += strand.charAt(i);
      if(i != strand.length() - 1) {
         if(targetMatch(newDna, guide)) {
            newDna += "-";
         }
      }
   }
   return newDna;
}

// A simple target match
private boolean targetMatch(String strand, String guide) {
   int beginIndex = strand.length() - guide.length();
   if(beginIndex < 0) {
      return false;
   }
   String substr = strand.substring(beginIndex);
   return substr.equals(guide);
}
```

```
// A variant of target match
private boolean targetMatch2(String strand, String guide) {
   int beginIndex = strand.length() - guide.length();
   if(beginIndex < 0) {
      return false;
   }
   for(int i = 0; i < guide.length(); i++) {
      int strandIndex = beginIndex + i;
      if(strand.charAt(strandIndex) != guide.charAt(i)){
         return false;
      }
   }
   return true;
}

// A cheeky solution
private String crispr(String strand, String guide) {
   String temp = strand.replaceAll(guide, guide + "-");
   if(temp.charAt(temp.length()-1) == '-') {
      temp = temp.substring(0, temp.length() - 1);
   }
   return temp;
}
```