

To Be Submitted by Midnight, Dec. 9 (Monday)

40 Points

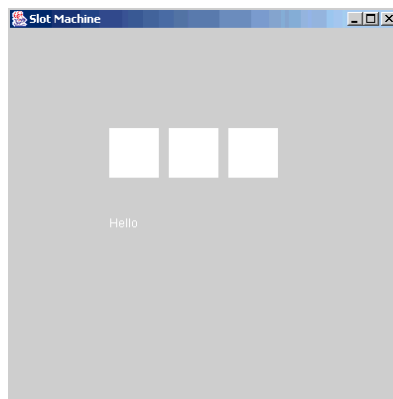
New Concepts: Searching algorithms (linear and binary search), using the vector class to store an unspecified number of objects, wrapper classes, graphics programming with java, using frames and panels, layout managers (FlowLayout, GridLayout, and BorderLayout), using the paintComponent method, using the Color, Font and FontMetrics classes, java drawing methods in the Graphics class, and basic concepts of event-driven programming.

PROBLEM STATEMENT:

Once again, we are expanding on the slot machine simulation theme. The slot machines are getting more complicated, and, in this final programming assignment, you will also build a graphical representation. You will be provided with 2 files that comprise a basic framework of the program. Your assignment is to expand the files to include the full functionality specified in this document. The files include:

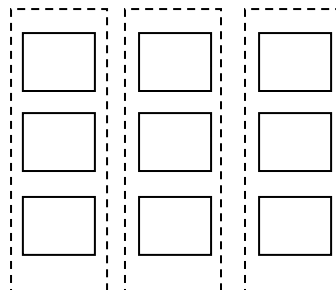
1. Assign6.java – contains the main method
2. DisplaySlotMachine.java – provides the frame and panel for the graphics window.

As provided in these files, the graphics window will look like the figure below:



Note that there are 3 small squares and 1 text message “painted” on the panel. You will add more components to show a slot machine look with **9 display regions** that make up the output for the **3 wheels** of the slot machine. The output forms a matrix as shown below. Each column shows the output for one wheel.

wheel 1 wheel 2 wheel 3

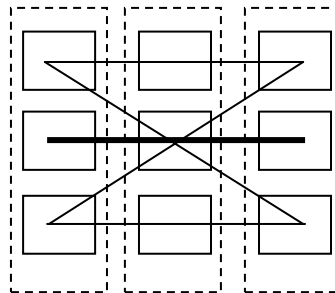


Here’s how the slot machines work. As with Program #5, there are 5 types of fruit as defined below with their respective indices:

Index	1	2	3	4	5
Fruit	Cherry	Orange	Apple	Banana	Plum

For each “pull” of the slot machine, 3 fruit will be selected randomly for each wheel and displayed in the output squares of the matrix. Thus, for each slot machine, you will need to generate 9 random numbers between 1 and 5 to represent the fruit selections. You may use any random number generator. Payoffs will be made for combinations across a row (any row) or across a diagonal (either diagonal). This means that for each pull, you will have to check 5 random number sequences (where each sequence has 3 numbers) for a possible winning combination.

wheel 1 wheel 2 wheel 3



The payoffs will be determined according to the table below. Use the payoff amounts as shown for the top and bottom rows and the two diagonals. *If a winning combination appears on the middle row, double the amounts shown on the table below.* If a pull results in more than one winning combination, then use the highest payoff amount.

Payoff level	Combination 1 One cherry on any wheel	Combination 2 2 matches of any type fruit	Combination 3 3 matches of any type fruit
Payoff level 1	\$2	\$3	\$5
Payoff level 2	\$3	\$4	\$10
Payoff level 3	\$4	\$5	\$15
Payoff level 4	\$5	\$8	\$25

As before, each slot machine will have a designated payoff level that will be used to determine the payoff. For example, 1-3-1 is equal to Cherry-Apple-Cherry (Combination 2). If the payoff level is 2, and the combination appears on the top row, the payoff amount is \$4; if the combination appears on the middle row, the payoff amount is \$8.

You should *include a button on the panel to “pull” the slot machine* (one pull at a time), and you should also have a *text box that displays the payoff amount for the current pull and another that displays the net dollars for all pulls thus far.* The net dollars made is the payoff minus the cost to play. (The net dollars might be negative.)

In this program, you will also read a list of slot machines from an input file, using our favorite InputReader class. As the file is read, check each line for valid input and add the entries into an array (for extra credit, use a Vector instead). Once the file is read, sort the slot machines by name (alphabetically) using your Bubble Sort method. *Include an input textbox on the graphics panel for entering the desired slot machine name.* For each new name entered, you must search the list of slot machines (using a binary search) to find the payoff level for that machine and the cost to play. If the entered slot machine name is not found, then the pull button should NOT perform the pull operation.

LIST OF THE REQUIRED GRAPHICS COMPONENTS:

1. The 3 by 3 matrix of the wheel output regions (for minimum points, display the fruit indices; for extra credit, display an image or figure that looks like the fruit, in each region)
2. pull button
3. input text box for entering the slot machine name
4. output text box for displaying the current payoff
5. output text box for displaying the net dollars made thus far for all slot machines played

THE INPUT FILE:

The first line in the input file will contain the number of possible slot machines (i.e., the number of lines contained in the rest of the input file). Each subsequent line contains the slot machine name (you may assume it will be one word long with no spaces), the cost to play the machine, and the payoff level. For example, take the input line:

```
LuckyDucky 5 3
```

“LuckyDucky” is the name of the slot machine, with a \$5 cost per pull, and a payoff level of 3.

SAMPLE INPUT:

```
7
LuckyDucky 5 3
Dubya Bush 3
Dubya_Bush 3 3
Dubya_Bush 4 5
SpruceGoose 20 5
SpruceGoose 20 4
Bubba_Gump 10 1
```

SAMPLE OUTPUT:

```
LuckyDucky ($5 per pull, level 3)
Invalid input line ignored: Dubya Bush 3
Dubya_Bush ($3 per pull, level 3)
Invalid input line ignored: Dubya_Bush 4 5
Invalid input line ignored: SpruceGoose 20 5
SpruceGoose ($20 per pull, level 4)
Bubba_Gump ($10 per pull, level 1)
```

SORT BY NAME

```
Bubba_Gump
Dubya_Bush
LuckyDucky
SpruceGoose
```

ERROR CHECKING:

Your program should include an error check of the *number of inputs on a line* and whether the inputs are *valid* and in the proper *order*. All valid inputs besides the slot machine names should be integers. *Valid payoff levels are 1 through 4 only*. Invalid input lines should be ignored and an error message should be displayed with the invalid input line, such as the examples above.

INPUTS FOR PROGRAM GRADING:

As in Assignments #3 and #4, the actual input file used for program grading will not be provided to students. You should again make up your own input file to test your program. Include a variety of possible test cases to ensure that your program will handle any error included in the actual input file.

PROGRAM SUBMISSION:

You will submit your program as usual, but it will not be executed as part of the submission process. Instead, each student will sign up for a time to demonstrate the program to a TA (Tuesday, Dec. 10 to Thursday, Dec. 12).

PROGRAM HEADER:

Include the following information in the program header, and fill in the information on the definition of the problem, the algorithm steps, the expected inputs, the generated outputs, and the error checking performed

/******

NAME: XXXXXXXXXX XXXXXXXXXXXX ASSIGNMENT # 6
STUDENT# XXXXXX
SECTION# x
LAB TA: XXXXXXXXXX

DEFINITION OF PROBLEM:

ALGORITHM STEPS FOR PROGRAM INPUT AND ERROR CHECKING:

INPUT:

OUTPUT:

ERROR CHECKING:

ALGORITHM STEPS FOR THE PULL METHOD:

*****/

GRADING:

8 points – Documentation (complete program header, informative identifier names, appropriate indentation and embedded comments)

8 points – Programming style (consistent with standard java conventions, consistent with requirements given, corresponds to algorithm steps provided in the header)

8 points – Output (correct values, including error handling, sorting)

16 points – Graphics window (required components, correct operation of the pull method and the corresponding output of the wheel display regions)

40 total points possible

COMPILER ERROR(S): 20 points deducted from your score

RUNTIME ERROR: 10 points deducted from your score

EXTRA CREDIT POINTS:

- Using a Vector instead of an array for storing slot machines (3 points)
- Displaying fruit-like images in the wheel output regions (5 points)
- Adding a slot machine-like background to the graphics panel (3 points)
- Adding *meaningful* graphics components beyond the required components (0-5 points, at the discretion of the TA)

GETTING HELP:

Students are required to correct most syntax errors and to bring a hardcopy of their program listing (complete with indicated errors and any output) with them when they wish to get help from the instructor or a Teaching Assistant (TA) or a Peer Learning Assistant (PLA) during office hours.

TA and PLA office hours are posted on the course web page:

<http://www.cecs.missouri.edu/~skubic/103>.

TAs, PLAs, and the instructor will assist students in understanding concepts, in finding elusive syntax errors, and in aiding correction of logic errors. However, they will NOT write or rewrite program code for students.