

AP Computer Science

Chapter 9 Program Assignments

Write a Java program for each of the following assignments. Make the input and/or output look like the following.

- 1) Do Project 9-6 on page 347. This project is composed of 5 separate program files. All the code can be found in your textbook starting on page 327. The five programs are: “TestModel” on pages 329-330, “TestScoresApp” on page 331, “Student” on pages 332-334, “TestScoresModel” on pages 334-336, and “TestScoresView” on pages 336-337. You will be typing in a new “Student” class file that uses arrays to store the test scores. Be sure to backup and rename the old “Student” class file from chapter 5 to “StudentCh5.java” before typing in the new “Student” class file.
- 2) Do Project 9-1 on page 347. Save as “EvenOdd.java.java”.

```
Enter integer #1: 15
Enter integer #2: -3
Enter integer #3: 10
Enter integer #4: 45
Enter integer #5: 28
Enter integer #6: -8
Enter integer #7: -25
Enter integer #8: 77
Enter integer #9: 0
Enter integer #10: -91

Even numbers: 10, 28, 0
Odd numbers: 15, 45, 77
Negative numbers: -3, -8, -25, -91
```

- 3) Do Project 9-2 on page 347. Save as “GreaterThanAverage.java”. Read 50 floating-point numbers from a text file name “numbers.txt”. The method used will be located in the same program (not in a separate class) and will be called from the main method. Be sure to precede this method with the reserved word “static”. A sample output is show below.

```
The average of the 50 numbers in the list is 245.07.
```

```
The following numbers in the list were greater than the average:
```

```
287.5
384.3
.
.
.
467.5
```

- 4) Write a program that will read the name and age of an unknown number of individuals (no more than 100) from a text file called "NamesAges.txt". The program places the names and ages into two separate parallel one-dimensional arrays. Print out the names and ages into two columns as shown below. Also create a prompt asking for the name of an individual and print out the individual's name and age as shown below. The prompt should continue asking for names until it sees the word STOP. Use the toUpperCase() method to convert all strings to upper case when you search for the names. Save the program as "NamesAges.java".

```
Name           Age
----          -
Anderson      25
Baader        18
.
.
.
Woosley       56
```

Enter a name to search: White

Name: White
Age: 34

Enter a name to search: lee

Name: Not Found

Enter a name to search: SIVLEY

Name: Sivley
Age: 35

Enter a name to search: StoP

- 5) Write a program that conducts an experiment in probability. Simulate the rolling of two 6-sided dice by using a random object. After the roll, the sum of the two dice are added and counted. The frequency of the different possible outcomes is stored in a one-dimensional array. After rolling the die a predetermined number of times, a table showing the outcome, frequency, probability, and percentage is printed out as show below. Save as "Probability.java". Answers will vary from those shown below.

Enter the number of times you wish to roll the dice (0 to quit): -100
Enter the number of times you wish to roll the dice (0 to quit): 1000

Outcome	Frequency	Probability	Percentage
2	35	35/1000	3.5%
3	49	49/1000	4.9%
4	85	85/1000	8.5%
5	113	113/1000	11.3%
6	145	145/1000	14.5%
7	154	154/1000	15.4%
8	134	134/1000	13.4%
9	106	106/1000	10.6%
10	92	92/1000	9.2%
11	54	54/1000	5.4%
12	33	33/1000	3.3%

Enter the number of times you wish to roll the dice (0 to quit): 0

- 6) Do Project 9-7 on page 348. Save as "MagicSquare.java". Use loops to sum up the numbers in the rows, columns, and diagonals (see page 324 – "Sum the Rows").

```
Enter the 4 numbers in row #1: 16 3 2 13
Enter the 4 numbers in row #2: 5 10 11 8
Enter the 4 numbers in row #3: 9 6 7 12
Enter the 4 numbers in row #4: 4 15 14 1
```

The square you entered is a magic square with a sum of 34.

Do you wish to do another (Y/N)? y

```
Enter the 4 numbers in row #1: 12 5 8 10
Enter the 4 numbers in row #2: 10 5 18 45
Enter the 4 numbers in row #3: 25 1 3 5
Enter the 4 numbers in row #4: 10 2 30 40
```

The square you entered is NOT a magic square.

Do you wish to do another (Y/N)? N

- 7) Do Project 9-10 on page 349. Save as "PennyPitch.java". Include in this program two private static void methods. The first method called "initGame()" will create the two-dimensional array and initialize the array to the correct starting values. The second method called "displayList()" will print out the two-dimensional board with the number values and a "P" where a penny has landed. Additionally, you will need to use a separate class called "Square.java" that contains Square objects. The Square object is made up of two variables: one to store the number value of the square and one to store if a penny has landed on the number. The class also contains methods that will be made use of in the main program. Only one penny can land on each number on the board.

```
Penny Pitch Board Game
=====
```

```
1 1 1 1 1
1 2 2 2 1
1 2 3 2 1
1 2 2 2 1
1 1 1 1 1
```

Press enter to toss a penny:

```
1 1 1 1 1
1 P 2 2 1
1 2 3 2 1
1 2 2 2 1
1 1 1 1 1
```

Toss #1

Total amount: 2

Press enter to toss a penny:

```
1 1 1 1 1
1 P 2 2 1
1 2 P 2 1
1 2 2 2 1
1 1 1 1 1
```

Toss #2

Total amount: 5

.
.
.