Chapter 6 – Page 271 - #7 MDoctor

# Problem Statement

5 test scores need to be entered repetitively with input validation. The program will determine the test score’s grade based on the following:

|  |  |
| --- | --- |
| **Score** | **Letter Grade** |
| 90-100 | A |
| 80-89 | B |
| 70-79 | C |
| 60-69 | D |
| Below 60 | F |

The program should then display a letter grade for each score and the average test score. Additionally, a determineGrade function should be used as follows:

determineGrade -- This function should accept a test score as an argument and return a letter grade for the score (as a String), based on the following grading scale (use a CASE/switch - see P 36 - 38 in Java Language Companion):

# Algorithm

1. Ask the user to enter a test score
2. Validate input
3. Repeat step 1 and 2 until all scores are entered
4. Display:
   1. Letter grade for each score (determineGrade)
   2. Average test score

# IPO Diagrams

Main module

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| (constant SIZE = 5)  score | While count < SIZE:  get score  Validate score: Call isValid(score)  Add score to testScores[]  Call report(testScores[]) |  |

IsValid module

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| score | If score < 0 or score > 100 or NaN  Set isValid = false  Else  Set isValid = true | Return isValid |

DetermineGrade module

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| score | Select Case (score / 10):   * 10: grade = ‘A’ * 9: grade = ‘A’ * 8: grade = ‘B’ * 7: grade = ‘C’ * 6: grade = ‘D’ * Default: grade = ‘F’ | Return grade |

Report module

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| testScore[] | Set grade = Call determineGrade(testScore[var]) | For each var in testScore[]  Display: testScore[var]  grade  end For  Display total / SIZE |

# Hierarchy Chart

# Flowchart

# Pseudocode

// Program: Chapter 6 - Page 271 - #7 v1.0

// Author: Mark Doctor

// Course: ITech

void main ()

{

// SIZE controls array size

while (count < SIZE)

{

DISPLAY "Enter test score ", count, ":";

if (isValid(score))

{

Add score to testScore[];

count++;

}

}

report (testScore[]);

}

Boolean isValid (score)

{

if (if score >= 0 AND score <= 100)

SET isValid = true;

if (isValid == false)

DISPLAY "Please enter a valid test score";

return isValid;

}

String determineGrade (score)

{

switch (SELECT Case (score / 10))

{

case 10:

grade = 'A';

break;

case 9:

grade = 'A';

break;

case 8:

grade = 'B';

break;

case 7:

grade = 'C';

break;

case 6:

grade = 'D';

break;

default:

grade = 'F';

}

return grade;

}

void report (testScore[])

{

for (count = 0; count <= 4; count = count + 1)

{

SET grade = determineGrade (testScore[count]);

DISPLAY "Score ", count + 1, + ": ", testScore[count];

DISPLAY "Grade ", count + 1, ": ", grade;

SET total += testScore[count];

}

DISPLAY "Average score: ", total / SIZE;

}

# Java Source Code

1 //Mark Doctor, 10/13/16, iTechPM Section 6: Assignment 6a  
 2 //Purpose: Functions  
 3 //Filename: testScore.java  
 4 //Documentation: In Class Chapter 6 - P271 - #7 MDoctor.docx  
 5 import java.util.\*;  
 6 public class testScore  
 7 {  
 8 final static int SIZE = 5; //constant controls array size  
 9 public static void main(String[] args)  
 10 {  
 11 Scanner kb = new Scanner(System.in);  
 12 String score;  
 13 int count = 0;  
 14 int[] testScore = new int[SIZE];  
 15 //while loop to gather test score input  
 16 while(count < SIZE)  
 17 {  
 18 System.out.print("Enter test score " + (count + 1) +  
 19 ": ");  
 20 score = kb.nextLine();  
 21 //Input validation  
 22 if(isValid(score))  
 23 {  
 24 //Add score to testSCore and increment  
 25 testScore[count] = Integer.parseInt(score);  
 26 count++;  
 27 }  
 28 }//end of while loop  
 29   
 30 //Call report module to display scores, grades (through determineGrade method)  
 31 //and average score  
 32 report(testScore);  
 33   
 34 }//end of main  
 35   
 36 //isValid method returns true if score is a valid test score.  
 37 //allows while loop to increment  
 38 public static boolean isValid(String score)  
 39 {  
 40 boolean isValid = false;  
 41 try  
 42 {  
 43 if (Integer.parseInt(score) >= 0 && Integer.parseInt(score) <= 100)  
 44 {  
 45 isValid = true;  
 46 }//end of if statement  
 47 }//end of try  
 48 catch(NumberFormatException e)  
 49 {  
 50 }//end of catch  
 51 if (!isValid)  
 52 System.out.println("Please enter a valid test score");  
 53 return isValid;  
 54 }//end of isValid function  
 55   
 56 //report module displays scores, grades through determineGrade method,  
 57 //and average score  
 58 public static void report(int[] testScore)  
 59 {  
 60 char grade;  
 61 int total = 0;  
 62 for(int count = 0; count < SIZE; count++)  
 63 {  
 64 grade = determineGrade(testScore[count]);  
 65 System.out.println("Score " + (count + 1) + ": " +  
 66 testScore[count] + " --> " + grade);  
 67 total += testScore[count];  
 68   
 69 }//end of for loop  
 70 System.out.println("Average score: " + (total / SIZE));  
 71 }//end of report module  
 72   
 73 //determineGrade function calculates the grade from the testScore using  
 74 //a case structure  
 75 public static char determineGrade(int score)  
 76 {  
 77 char grade = ' ';  
 78 switch(score / 10)  
 79 {  
 80 case 10: ;  
 81 case 9: grade = 'A'; break;  
 82 case 8: grade = 'B'; break;  
 83 case 7: grade = 'C'; break;  
 84 case 6: grade = 'D'; break;  
 85 default: grade = 'F'; break;  
 86 }//end of case structure  
 87 return grade;  
 88 }//end of determineGrade function  
 89 }//end of class  
 90 /\*  
 91  ----jGRASP exec: java testScore  
 92 Enter test score 1: hundred  
 93 Please enter a valid test score  
 94 Enter test score 1: 100  
 95 Enter test score 2: -50  
 96 Please enter a valid test score  
 97 Enter test score 2: 80  
 98 Enter test score 3: 60  
 99 Enter test score 4: 30  
100 Enter test score 5: 89  
101 Score 1: 100 --> A  
102 Score 2: 80 --> B  
103 Score 3: 60 --> D  
104 Score 4: 30 --> F  
105 Score 5: 89 --> B  
106 Average score: 71  
107   
108  ----jGRASP: operation complete  
109 \*/  
110