Alexander Laudino

CSC-162-IN1

Dr. Farrett

Lab Assignment 9

TestScores Class & TestScores Exception Class Reference Documents

Pseudo-code

Package withoutCustomException

This program demonstrates the TestScores class which prompts the user to enter a series of test scores, the program checks whether any of the scores are negative or greater than 100, if found, the class throws an IllegalArgumentException.

DemoTestScores

Begin main(args: String[])

1. Print " ~ Test valid scores. ~"
2. Create ArrayList<Double> for test scores as scores, call getTestScores() and initialize scores
3. Try
4. Create new TestScores object as s0 with scores as parameter
5. Print s0
6. Catch IllegalArgumentException
7. Print stack trace
8. Sleep for 1000 ms
9. Print "~ Test invalid score over 100. ~"
10. Create ArrayList<Double> for test scores as scores1, call getTestScores() and initialize scores1
11. Try
12. Create new TestScores object as s1 with scores1 as parameter
13. Print s1
14. Catch IllegalArgumentException
15. Print stack trace
16. Sleep for 1000 ms
17. Print "~ Test invalid score below 0. ~"
18. Create ArrayList<Double> for test scores as scores2, call getTestScores() and initialize scores2
19. Try
20. Create new TestScores object as s2 with scores2 as parameter
21. Print s2
22. Catch IllegalArgumentException
23. Print stack trace

End

The methods for creating TestScores objects

TestScores

1. Create new Scanner object as INPUT
2. Create double variable for average scores as averageScore
3. Create double[] array to hold scores as scores

Begin TestScores()

1. Empty constructor

End

Begin TestScores(testScores: ArrayList<Double>) throws IllegalArgumentException

1. Call setAverageScore(testScores)
2. Call setScores(testScores)

Begin setAverageScore(testScores: ArrayList<Double>) throws IllegalArgumentException

1. Create double variable for sum of test scores as sum and set to 0
2. for score in testScores
3. if score > 100 or score < 0
4. throw new IllegalArgumentException(“Score cannot be greater than 100 or negative”
5. else
6. sum += score
7. averageScore = sum / # of elements in ArrayList

End

Begin getScore()

1. Create Boolean variable for controlling loop as continueInput and set to true
2. Create double variable for test score as score and set to 0
3. Begin do-while loop
4. Begin try block
5. Print “Enter a test score: “
6. score = next double input
7. Print “The test score entered is “ + score
8. continueInput = false
9. Catch InputMisMatchException as ex
10. Print “Try again. (Incorrect input: an integer is required.)”
11. Discard current input line
12. While continueInput is true
13. Return score

End

Begin getTestScores()

1. Create ArrayList<Integer> variable for test scores as scores
2. Create Boolean variable for control of loop as continueInput and set to true
3. Begin do-while loop
4. Print “\nEnter another score? (Y/N): “
5. String i = next String input
6. if i toLowerCase equals “n”
7. continueInput = false
8. else if i toLowerCase equals “y”
9. double score = getScore()
10. Add score to scores
11. continue
12. else
13. Print “Try again. (Invalid input: Y or N required.)”
14. Discard current input line
15. End do-while loop when continueInput = false
16. Return scores

End

Begin setScores(testScores: ArrayList<Double>)

1. Set size of this.scores array to size of ArrayList
2. int index = 0
3. for test in testScores
4. this.score[index] = test
5. Index++

End

Begin displayScores()

1. String output = “”
2. int index = 1
3. for score in this.scores
4. output = output + “\nTest score #” + index + “: “ + score
5. index++
6. return output

End

Begin getAverageScore()

1. return averageScore

End

Begin getGrade()

1. if averageScore >= 93
2. return “A”
3. else if averageScore >= 90
4. return “A-“
5. else if averageScore >= 87
6. return “B+”
7. else if averageScore >= 83
8. return “B”
9. else if averageScore >= 80
10. return “B-“
11. else if averageScore >= 77
12. return “C+”
13. else if averageScore >= 70
14. return “C”
15. else if averageScore >= 60
16. return “D”
17. else
18. return “F”

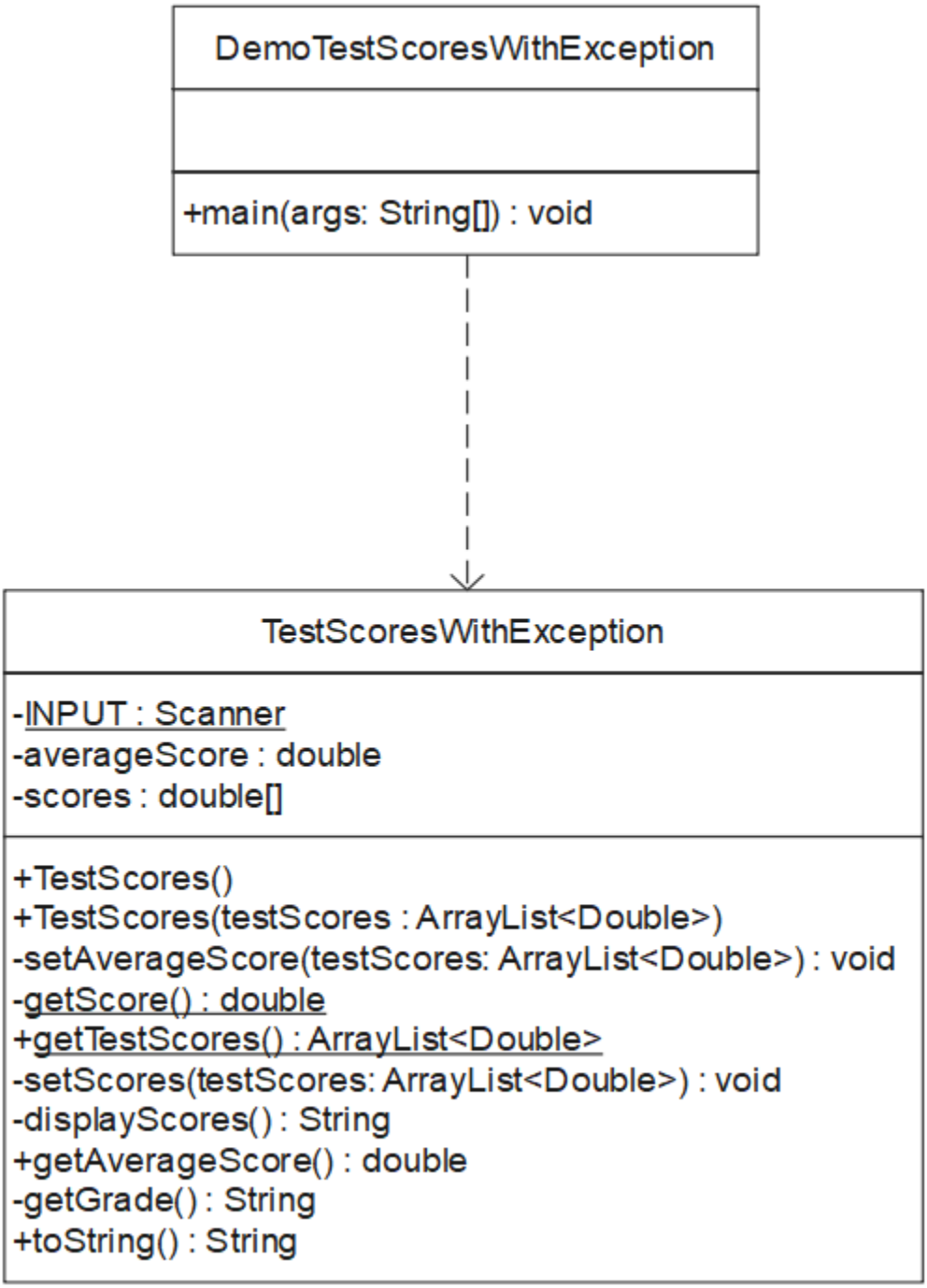
End

Begin toString()

1. return displayScores() + “Average score: “, getAverageScore() + “Grade: “ + getGrade()

End

UML


DemoTestScoresWithException
+main(args: String[]) : void
TestScoresWithException
-INPUT : Scanner
-averageScore : double
-scores : double[]
+TestScores()
+TestScores(testScores : ArrayList<Double>)
-setAverageScore(testScores: ArrayList<Double>) : void
-getScore() : double
+getTestScores() : ArrayList<Double>
-setScores(testScores: ArrayList<Double>) : void
-displayScores() : String
+getAverageScore() : double
-getGrade() : String
+toString() : String

Pseudo-code

Package withCustomException

This program demonstrates the TestScoresWithException class which prompts the user to enter a series of test scores, the program checks whether any of the scores are negative or greater than 100, if found, the class throws an InvalidTestScore exception.

DemoTestScoresWithException

Begin main(args: String[])

1. Print " ~ Test valid scores. ~"
2. Create ArrayList<Double> for test scores as scores, call getTestScores() and initialize scores
3. Try
4. Create new TestScores object as s0 with scores as parameter
5. Print s0
6. Catch InvalidTestScore
7. Print stack trace
8. Sleep for 1000 ms
9. Print "~ Test invalid score over 100. ~"
10. Create ArrayList<Double> for test scores as scores1, call getTestScores() and initialize scores1
11. Try
12. Create new TestScores object as s1 with scores1 as parameter
13. Print s1
14. Catch InvalidTestScore
15. Print stack trace
16. Sleep for 1000 ms
17. Print "~ Test invalid score below 0. ~"
18. Create ArrayList<Double> for test scores as scores2, call getTestScores() and initialize scores2
19. Try
20. Create new TestScores object as s2 with scores2 as parameter
21. Print s2
22. Catch InvalidTestScore
23. Print stack trace

End

The methods for creating TestScoresWithException objects

TestScoresWithException

1. Create new Scanner object as INPUT
2. Create double variable for average scores as averageScore
3. Create double[] array to hold scores as scores

Begin TestScoresWithException()

1. Empty constructor

End

Begin TestScoresWithException(testScores: ArrayList<Double>) throws InvalidTestScore

1. Call setAverageScore(testScores)
2. Call setScores(testScores)

Begin setAverageScore(testScores: ArrayList<Double>) throws InvalidTestScore

1. Create double variable for sum of test scores as sum and set to 0
2. for score in testScores
3. if score > 100 or score < 0
4. throw new IllegalArgumentException("Score cannot be greater than 100 or negative"
5. else
6. sum += score
7. averageScore = sum / # of elements in ArrayList

End

Begin getScore()

1. Create Boolean variable for controlling loop as continueInput and set to true
2. Create double variable for test score as score and set to 0
3. Begin do-while loop
4. Begin try block
5. Print "Enter a test score: "
6. score = next double input
7. Print "The test score entered is " + score
8. continueInput = false
9. Catch InputMisMatchException as ex
10. Print "Try again. (Incorrect input: an integer is required.)"
11. Discard current input line
12. While continueInput is true
13. Return score

End

Begin getTestScores()

1. Create ArrayList<Integer> variable for test scores as scores
2. Create Boolean variable for control of loop as continueInput and set to true
3. Begin do-while loop
4. Print “\nEnter another score? (Y/N): “
5. String i = next String input
6. if i toLowerCase equals “n”
7. continueInput = false
8. else if i toLowerCase equals “y”
9. double score = getScore()
10. Add score to scores
11. continue
12. else
13. Print “Try again. (Invalid input: Y or N required.)”
14. Discard current input line
15. End do-while loop when continueInput = false
16. Return scores

End

Begin setScores(testScores: ArrayList<Double>)

1. Set size of this.scores array to size of ArrayList
2. int index = 0
3. for test in testScores
4. this.score[index] = test
5. Index++

End

Begin displayScores()

1. String output = “”
2. int index = 1
3. for score in this.scores
4. output = output + “\nTest score #” + index + “: “ + score
5. index++
6. return output

End

Begin getAverageScore()

1. return averageScore

End

Begin getGrade()

1. if averageScore >= 93
2. return “A”
3. else if averageScore >= 90
4. return “A-“
5. else if averageScore >= 87
6. return “B+”
7. else if averageScore >= 83
8. return “B”
9. else if averageScore >= 80
10. return “B-“
11. else if averageScore >= 77
12. return “C+”
13. else if averageScore >= 70
14. return “C”
15. else if averageScore >= 60
16. return “D”
17. else
18. return “F”

End

Begin toString()

1. return displayScores() + “Average score: “, getAverageScore() + “Grade: “ + getGrade()

End

The methods for creating InvalidTestScore exceptions

InvalidTestScore

1. Create new double variable to hold invalid score as score

Begin InvalidTestScore(index: int, score: double)

1. super(“Invalid score! Test #” + index + “: “ + score + “\nScore cannot be greater than 100 or negative”
2. this.score = score

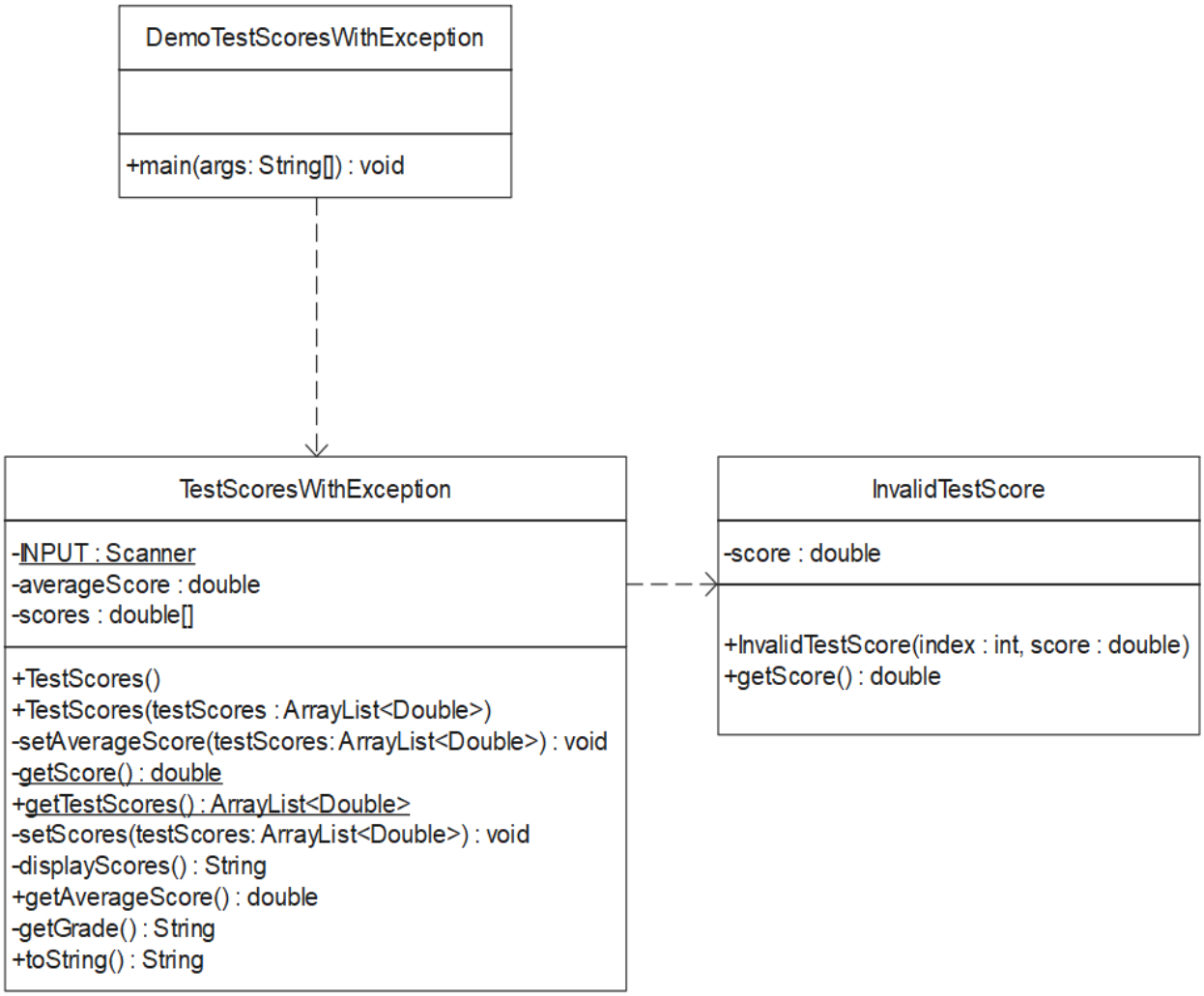
End

Begin getScore()

1. return score

End

UML


DemoTestScoresWithException
+main(args: String[]) : void
TestScoresWithException
-INPUT : Scanner
-averageScore : double
-scores : double[]
+TestScores()
+TestScores(testScores : ArrayList<Double>)
-setAverageScore(testScores: ArrayList<Double>) : void
-getScore() : double
+getTestScores() : ArrayList<Double>
-setScores(testScores: ArrayList<Double>) : void
-displayScores() : String
+getAverageScore() : double
-getGrade() : String
+toString() : String
InvalidTestScore
-score : double
+InvalidTestScore(index : int, score : double)
+getScore() : double