1 **import** edu.sjcny.gpv1.\*;

2 **import** java.awt.\*;

3 **import** javax.swing.\*;

4 **import** java.util.Scanner;

5 **import** java.io.\*;

6

7 **public** **class** ExceptionBasics **extends** DrawableAdapter

8 {

9 **static** ExceptionBasics ge = **new** ExceptionBasics ();

10 **static** GameBoard gb = **new** GameBoard(ge, "Exception Basics");

11 **static** BoxedSnowman s1 = **new** BoxedSnowman(300, 200, Color.GREEN);

12 **static** BoxedSnowman s2 = **new** BoxedSnowman(30, 100, Color.BLACK);

13 **static** **int** score = 0;

14 **static** **int** count = 10;

15

16 **public** **static** **void** main(String[] args)

17 {

18 showGameBoard(gb);

19 }

20

21 **public** **void** draw(Graphics g) **// a call back method**

22 {

23 **int** w = 40;

24 **int** h = 77;

25 **int** s1X, s1Y, s2X, s2Y, temp;

26

27 s1X = s1.getX(); s1Y = s1.getY();

28 s2X = s2.getX(); s2Y = s2.getY();

29 g.setColor(Color.BLACK);

30 g.setFont(new Font("Arial", Font.BOLD, 18));

31 g.drawString("Time remaining: " + count, 260, 50);

32

33 **if**(count == 0) **// the game is over**

34 {

35 g.setColor(Color.BLACK);

36 g.drawString("Game Over", 205, 70);

38 g.drawString("Have a Good Day", 175, 90);

39

40 **try**

41 {

42 **int** highScore;

43 File fileObj = **new** File("HiScore.txt");

44 Scanner fileIn = **new** Scanner(fileObj);

45 highScore = fileIn.nextInt();

46 fileIn.close();

47

48 **if**(score >= highScore) **// a new high score**

49 {

50 g.drawString("Great, Your Score is the Highest Ever.," +

51 " It Will Be Saved", 10, 110);

52 FileWriter fileWriterObj = **new** FileWriter("HiScore.txt");

53 PrintWriter fileOut = **new** PrintWriter(fileWriterObj, false);

54

55 fileOut.println(score);

56 fileOut.close();

57 }

58 **else** **// not a new high score**

59 {

60 g.drawString("Best Score is: " + highScore +

61 ", Keep Practicing", 110, 110);

62 }

63 }

64 **catch**(IOException e)

65 {

66 g.drawString("Problems With High Score File", 120, 110);

67 }

68 }

69 **else** **if**( !(s2X > s1X + w || s2X + w < s1X || s2Y > s1Y + h ||

70 s2Y + h < s1Y) && s1.getVisible() == true)

71 {

72 score = score + 1;

73 s1.setVisible(false);

74 }

75 **else** **if**( s2X > s1X + w || s2X + w < s1X || s2Y > s1Y + h ||

76 s2Y + h < s1Y) **// no collision**

77 {

78 **if**(s1.getVisible() == false) **// not visible**

79 { temp = s1.getX();

80 s1.setX(s1.getY());

81 s1.setY(temp);

82 s1.setVisible(true);

83 }

84 }

85

86 s2.show(g);

87 **if**(s1.getVisible() == true)

88 {

89 s1.show(g);

90 }

91 g.setColor(Color.BLACK);

92 g.drawString("Score: " + score, 150, 50);

93 }

94

95 **public** **void** keyStruck(**char** key) **// a call back method**

96 {

97 **int** newX, newY;

98

99 **switch** (key)

100 {

101 **case** 'L':

102 {

103 newX = s2.getX() - 2;

104 s2.setX(newX);

105 **break**;

106 }

107 **case** 'R':

108 {

109 newX = s2.getX() + 2;

110 s2.setX(newX);

111 **break**;

112 }

113 **case** 'U':

114 {

115 newY = s2.getY() - 2;

116 s2.setY(newY);

117 **break**;

118 }

119 **case** 'D':

120 {

121 newY = s2.getY() + 2;

122 s2.setY(newY);

123 }

124 }

125 }

126 **public** **void** timer1() **// a call back method**

127 {

128 count = count - 1;

129 **if**(count == 0)

130 {

131 gb.stopTimer(1);

132 }

133 }

134 }

**Figure 4.23 The ExceptionBasics Application: A decision and exceptions case study.**