한양대학교

OBJECT-ORIENTED SYSTEMS DESIGN (Lab9)

Heejin Park

Hanyang University

9-1 (Display 9.1)

Create a class *InputMismatchExceptionDemo* that prints the output below. The detailed description of the class is given on the next page.

<input and output>

```
Enter a whole number:
forty two
Not a correctly written whole number.
Try again.
Enter a whole number:
Fortytwo
Not a correctly written whole number.
Try again.
Enter a whole number:
42
You entered 42
```

9-1

[*InputMismatchExceptionDemo*]

1. Import 2 classes as follows.

    java.util.Scanner

    java.util.InputMismatchException

2. Write a method **main**:

    1. Create **Scanner** object:
    2. Create **int** *number* initialized to *0*.
    3. Create **boolean** *done* initialized to *false*.

4. Create a **while-statement** that is executed when *done* is false.

    1. Create **try** block.

        Print the output below.

        <output>

        Enter a whole number:

        Store an input integer into *number*.

        Store true in *done*.

2. Create **catch**(InputMismatchException **e**) block.

Perform the enter action using **nextLine().**

Print the output below.

<output>

Not a correctly written whole number.
Try again.

5. Print “You entered ” + *number*.

9-2 (Display 9.3)

Create a class *DanceLesson* that prints the output below.

The detailed description of the class is given on the next page.

<input and output>

Enter number of male dancers:
4
Enter number of female dancers:
6
Each man must dance with 1.5 women.
Begin the lesson.

<input and output>

Enter number of male dancers:
0
Enter number of female dancers:
5
Lesson is canceled. No men.

<input and output>

Enter number of male dancers:
0
Enter number of female dancers:
0
Lesson is canceled. No students.

<input and output>

Enter number of male dancers:
4
Enter number of female dancers:
0
Lesson is canceled. No women.

[*DanceLesson*]

1. Import a class **java.util.Scanner.**

2. Write a method **main**:

    1. Create a **Scanner** object:
    2. Print “Enter number of male dancers:”.
    3. Create **int** *men* and store an input integer into it.
    4. Print “Enter number of female dancers:”.
    5. Create **int** *women* and store an input integer into it.

6. Create **try** block.

1. If *men* and *women* are 0, throw an exception with a message “Lesson is canceled. No students.”. If *men* is 0, throw an exception with “Lesson is canceled. No men.”. If *women* is 0, throw an exception with “Lesson is canceled. No women.”.

2. If *women* is more than or equal to *men*, print “Each man must dance with ” + *women* /(double)*men* + “ women.”. Otherwise, print “Each woman must dance with ” + *men* /(double)*women* + “ men.”.

7. Create **catch**(Exception **e**) block.

    Create String *message* and store the thrown message into it using **getMessage()**. And then, print *message* and exit.

8. Print “Begin the lesson.”.

9-3 (Display 9.6)

Create a class *BadNumberException* as a derived class of the class *Exception*.

[*BadNumberException*]

1. The class *BadNumberException* extends *Exception.*
2. Create **private int** *badNumber*.
3. Create a constructor **public BadNumberException**(int number) :

   Invoke **super("BadNumberException")**.

   Store *number* in *badNumber.*

4. Create a no-arg constructor **public BadNumberException()** :

   Invoke **super("BadNumberException")**.

5. Create a method **public BadNumberException**(String message) :

   Invoke **super(message)**.

6. Create a method **public** int **getBadNumber()** :

   Return *badNumber*.

9-4 (Display 9.7)

Create a class *BadNumberExceptionDemo* that prints the output below using the class *BadNumberException*.

The detailed description of the class is given on the next page.

<input and output>

```
Enter one of the numbers 42 and 24:
42
Thank you for entering 42
End of program.
```

<input and output>

```
Enter one of the numbers 42 and 24:
44
44 is not what I asked for.
End of program.
```

[*BadNumberExceptionDemo*]

1. Write a method **main**:
    1. Create **try** block.
        1. Create **Scanner** object:
        2. Print “Enter one of the numbers 42 and 24:”.
        3. Create **int** *inputNumber* and store an input integer into it.
        4. If *inputNumber* is not equal 42 and 24, throw **BadNumberException(inputNumber**).
        5. Print “Thank you for entering ” + *inputNumber*.

2. Create **catch**(BadNumberException **e**) block.

Print *e.getBadNumber()* + " is not what I asked for.".

3. Print "End of program.".

9-5 (Display 9.4)

Create a class *DivisionByZeroException* as a derived class of the class *Exception*.

[*DivisionByZeroException*]

1. Create a no-arg constructor public **DivisionByZeroException** () :

   Invoke **super("Division by Zero!")**.

2. Create a constructor public **DivisionByZeroException** (String **message**) :

   Invoke **super(message)**.

9-6 (Display 9.9)

Create a class *NegativeNumberException* as a derived class of the class *Exception*.

[*NegativeNumberException*]

1. Create a no-arg constructor public **NegativeNumberException** () :

   Invoke **super("Negative Number Exception!")**.

2. Create a constructor public **NegativeNumberException** (String **message**) :

   Invoke **super(message)**.

9-7 (Display 9.8)

Create a class *MoreCatchBlockDemo* that prints the output below using the classes *NegativeNumberException* and *DivisionByZeroException*.

The detailed description of the class is given on the next page.

<input and output 1>

```
How many pencils do you have?
5
How many erasers do you have?
2
Each eraser must last through 2.5 pencils
End of program.
```

<input and output 2>

```
How many pencils do you have?
-2
Cannot have a negative number of pencils
End of program.
```

<input and output 3>

```
How many pencils do you have?
5
How many erasers do you have?
0
Do not make any mistakes.
End of program.
```

[*MoreCatchBlockDemo*]

Write a method **main**:

1. Create a **try** block defined as follows.

   Print out “How many pencils do you have?”.

   Create a variable **int** *pencils* and store an input integer in it.

   If *pencils* is less than 0, throw **NegativeNumberException (“pencils”)**.

Print out “How many erasers do you have?”.

Create a variable **int** *erasers* and store an input integer in it.

If *erasers* is less than 0, throw **NegativeNumberException (“erasers”)**.

Create a variable **double** *pencilsPerEraser*.

If *erasers* is not equal to 0, store *pencils*/(**double**)*erasers* in *pencilsPerEraser*.

Otherwise, throw **DivisionByZeroException ()**.

Print out “Each eraser must last through ” + *pencilsPerEraser* + “ pencils.”.

2. Create two **catch** blocks defined as follows.

catch (NegativeNumberException e):

Print out “Cannot have a negative number of ” + **e.getMessage()**.

catch (DivisionByZeroException e):

Print out “Do not make any mistakes.”.

3. Print out “End of program.”.

9-8 (Display 9.10)

Create a class *DivisionDemo* that prints the output below using the class *DivisionByZeroException*.

The detailed description of the class is given on the next page.

<input and output 1>

```
Enter numerator:
11
Enter denominator:
5
11/5 = 2.2
End of program.
```

<input and output 2>

```
Enter numerator:
11
Enter denominator:
0
Division by Zero!
Try again.
Enter numerator:
11
Enter denominator:
5
11/5 = 2.2
End of program.
```

<input and output 3>

```
Enter numerator:
11
Enter denominator:
0
Division by Zero!
Try again.
Enter numerator:
11
Enter denominator:
0
I cannot do division by zero.
Aborting program.
```

[*DivisionDemo*]

1. Write a method **main**:

    1. Create a **try** block defined as follows:

        Print out “Enter numerator:”.

        Create a variable **int** *numerator* and store an input integer in it.

        Print out “Enter denominator:”.

        Create a variable **int** *denominator* and store an input integer in it.

        Create a variable **double** *quotient* and

        store **safeDivide(numerator, denominator)** in it.

        Print out *numerator* + “/” + *denominator* + “ = ” + *quotient*.

2. Create a **catch** block with a parameter **DivisionByZeroException e** defined as follows:

    Print out **e.getMessage()**.

    Invoke **secondChance()**.

3. Print out “End of program.”.

2. Create a method **public static double safeDivide(int top, int bottom) throws DivisionByZeroException:**

If **bottom** is 0, throw **new DivisionByZeroException ()**.

Return **top**/(double)**bottom**.

3. Create a method **public static void secondChance()**:

    1. Create a **try** block defined same as the one in the **main** method.

    2. Create a **catch** block with a parameter **DivisionByZeroException e** defined as follows:

        Print out “I cannot do division by zero.” and “Aborting program.” and exit.

9-9 (Display 9.13)

Create classes *ScoreNotSetException* as a derived class of the class *Exception* and *HighScore* that prints the output below.

The detailed description of the class is given on the next page.

<output>

Score not set
100

9-9

[*ScoreNotSetException*]

1. Create a no-arg constructor **public ScoreNotSetException ()** :

   Invoke **super("Score not set")**.

2. Create a constructor **public ScoreNotSetException** (String **message**) :

   Invoke **super(message)**.

9-9

[*HighScore*]

1. Create two instance variables as follows.

   private int *score* initialized to 0.

   private boolean *scoreSet* initialized to false.

2. Create a no-arg constructor **public HighScore()** :

   Store 0 into *score* and false into *scoreSet*.

3. Create a method **public void setScore(int newScore)** :

Copy **newScore** to *score* and store true into *scoreSet*.

4. Create a method **public int getScore() throws ScoreNotSetException** :

If not *scoreSet*, throw **new ScoreNotSetException ()**.

Otherwise, return *score*.

5. Write a method **main**:

    1. Create **HighScore** *highscore* by calling **HighScore**.

    2. Create a **try** block defined as follows.

        Print out **highscore.getScore()**.

    3. Create a **catch** block with a parameter **ScoreNotSetException e** defined as follows:

        Print out **e.getMessage()**.

4. Invoke **highscore.setScore**(100).

5. Create a **try** block defined as follows.

   Print out **highscore.getScore()**.

6. Create a **catch** block with a parameter **ScoreNotSetException e** defined as follows:

   Print out **e.getMessage()**.