CS159

Nathan Sprague

What's wrong with the following code?

```
1  /**********************************************
2  * Return the mean, or -1 if the array has length 0.
3  **********************************************/
4  public static double mean(double[] numbers)
5        {
6           double sum = 0;
7           double result;
8
9           if (numbers == null || numbers.length == 0)
10          {
11             result = -1;
12          }
13          else
14          {
15             for (int i = 0; i < numbers.length; i++)
16             {
17                sum += numbers[i];
18             }
19             result = sum / numbers.length;
20          }
21          return result;
22       }
```

Why Exceptions?

- Sometimes there is no appropriate return value that can be used to indicate an error has occurred. (Let's use exceptions to improve this code...)

Why Exceptions?

- Sometimes there is no appropriate return value that can be used to indicate an error has occurred. (Let's use exceptions to improve this code...)
- Exceptions provide flexibility in deciding where a particular problem should be handled. If an exception occurs, a method may:
  - "Pass the buck" by using the throws keyword. The exception will be handled somewhere higher-up in the call stack.
  - Deal with the exception using a try/catch block.

Fixed Mean Calculator

```
1   public static double mean(double[] numbers)
2   {
3       if (numbers == null || numbers.length == 0)
4       {
5           throw new IllegalArgumentException("Invalid array.");
6       }
7
8       double sum = 0;
9
10      for (int i = 0; i < numbers.length; i++)
11      {
12          sum += numbers[i];
13      }
14
15      return sum / numbers.length;
16  }
```

throw vs. throws

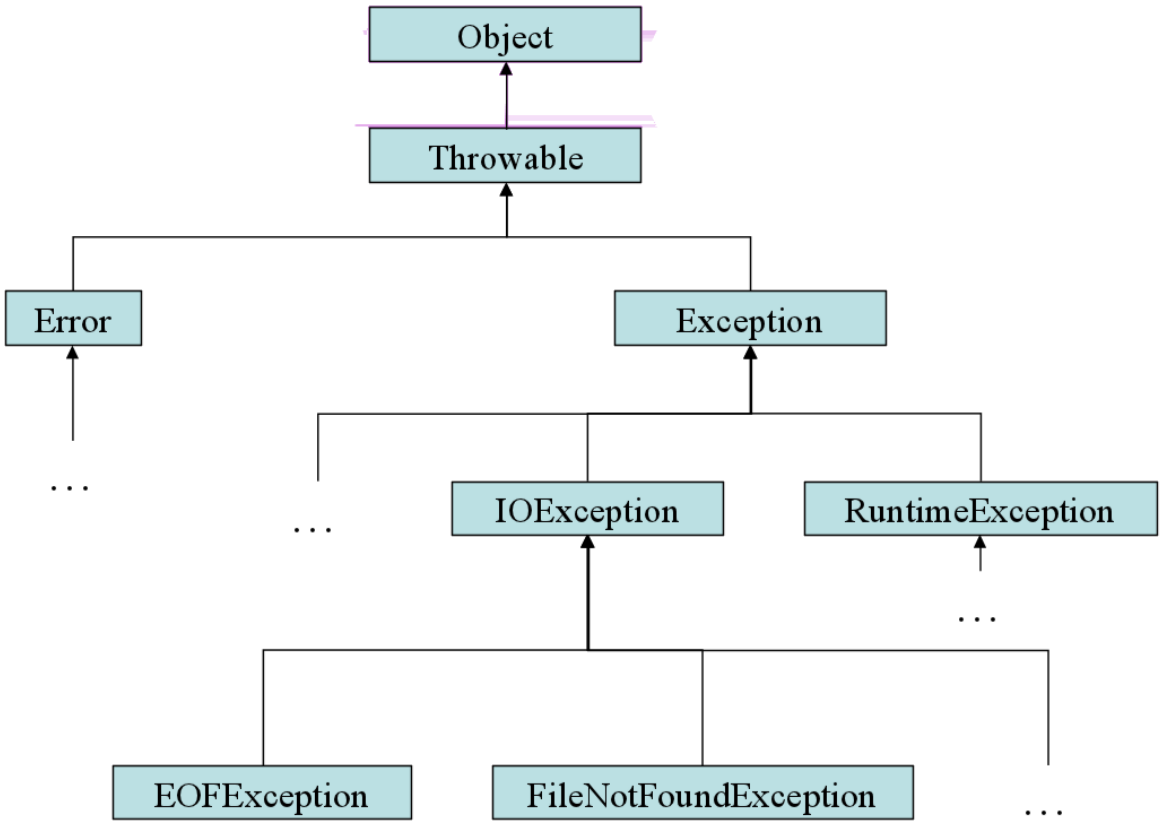
- The `throw` keyword is similar to `return`
  - Ends the method
  - Sends a result (exception object) to the caller
- The `throws` keyword is similar to a method's return type specification.

```
1    public double someMethod(double arg) throws SomeCheckedException
2    {
3        if (arg == 0.0)
4        {
5            throw new SomeCheckedException("Bad argument.");
6        }
7
8        return 10.0;
9    }
```

Checked vs. Unchecked Exceptions

- The `throws` keyword is only required for "checked exceptions"
  - They must be handled within the method with a try-catch block OR
  - Declared thrown with `throws` (passing the buck)
- "Unchecked exceptions" inherit from `RuntimeException`
  - Often result from programming errors, not exceptional circumstances.

Exception Class Hierarchy



try/catch

```
1   public double timesTen(double arg)
2   {
3       double result = 0;
4       try
5       {
6           result = someMethod(arg) * 10.0;
7       }
8       catch (SomeCheckedException e)
9       {
10          // Code here that fixes the problem...
11          // Maybe some other way of
12          // calculating a good result.
13      }
14      return result;
15  }
```

throws

```
1    public double timesTen(double arg) throws SomeCheckedException
2    {
3        return someMethod(arg) * 10.0;
4    }
```

Example...

Let's write a Java program that reads a text file containing simple mathematical expressions, and writes the results:

- If the input file contains:

```
2 + 3
22 / 2
3 * 2
```

- Terminal output should be:

```
5
11
6
```

What will be printed?

```
1           fileName = "NONEXISTENTFILE.txt";
2           System.out.print("A ");
3           try
4           {
5               System.out.print("B ");
6               file = new File(fileName);
7               scanner = new Scanner(file);
8               System.out.print("C ");
9
10          }
11          catch (FileNotFoundException e)
12          {
13              System.out.print("D ");
14          }
15          finally
16          {
17              System.out.print("E ");
18          }
19          System.out.print("F ");
```

Question

Characterize the following code, assuming `numbers` is an array of doubles. (There are no syntax errors.)

```
1           double sum = 0;
2           try
3           {
4               for (int i = 0; i <= numbers.length; i++)
5               {
6                   sum += numbers[i];
7               }
8           }
9           catch (ArrayIndexOutOfBoundsException e)
10          {
11              //Do nothing.
12          }
13          System.out.println(sum);
```

1. Correct result, appropriate use of exception handling.
2. Incorrect result, appropriate use of exception handling.
3. Correct result, inappropriate use of exception handling
4. Incorrect result, inappropriate use of exception handling.

Question

Characterize the following code, assuming `numbers` is an array of doubles. (There are no syntax errors.)

```
1           double sum = 0;
2           try
3           {
4               for (int i = 0; i <= numbers.length; i++)
5               {
6                   sum += numbers[i];
7               }
8           }
9           catch (ArrayIndexOutOfBoundsException e)
10          {
11              //Do nothing.
12          }
13          System.out.println(sum);
```

1. Correct result, appropriate use of exception handling.
2. Incorrect result, appropriate use of exception handling.
3. Correct result, inappropriate use of exception handling
4. Incorrect result, inappropriate use of exception handling.

3.

Handling vs. Throwing

The specification for the following method indicates that it should "throw an exception of type IllegalArgumentException if amount is less than 0." Is this implementation correct?

```
1   public void increaseValue(int amount)
2    {
3        try
4        {
5            if (amount < 0)
6            {
7                throw new IllegalArgumentException();
8            }
9            value += amount;
10       }
11       catch (IllegalArgumentException e)
12       {
13           System.out.println("Negative amount not allowed!");
14       }
15   }
```

Handling vs. Throwing

The specification for the following method indicates that it should "throw an exception of type IllegalArgumentException if amount is less than 0." Is this implementation correct?

```
1   public void increaseValue(int amount)
2    {
3        try
4        {
5            if (amount < 0)
6            {
7                throw new IllegalArgumentException();
8            }
9            value += amount;
10       }
11       catch (IllegalArgumentException e)
12       {
13           System.out.println("Negative amount not allowed!");
14       }
15   }
```

No.