Lab8

April 4, 2024

1 Lab 8

Deadline: **Week 10** in your respective lab session

1.0.1 Name: Hansel Rodrigues

1.0.2 Student ID: 230603866

In this lab we are looking for solutions that catch specific exceptions (e.g. IndexOutOfBoundException). Answers catching "generic" Exception will not be accepted.

1.1 Question 1 [1 mark]

Consider the method below, `getValue`. Identify a potential error that might occur when calling it and handle it using `try` and `catch`. The `catch` block should print out a message indicating that the error occurred and return `null`.

Write your answer below:

[12]:
```
static <T> T getValue(T[] array, int index)
{
    try
    {
        return array[index];
    }
    catch (ArrayIndexOutOfBoundsException e)
    {
        System.out.println("Error: Index out of bounds for array access.␣
 ↪Returning null.");
        return null;
    }
}
```

Run your program:

[13]:
```
Integer[] a = new Integer[3];
a[0] = 1;
```

```
a[1] = 2;
a[2] = 3;
System.out.println(getValue(a, -1));
System.out.println(getValue(a, 4));
System.out.println(getValue(a, 1));
```

```
Error: Index out of bounds for array access. Returning null.
null
Error: Index out of bounds for array access. Returning null.
null
2
```

1.2 Question 2 [1 mark]

Consider the method below, `getQuotient`. Identify potential errors that might occur when calling it and handle it using `try` and `catch`. The `catch` blocks should print out a message indicating what went wrong and return `0`.

Hint: We are looking to see two different exceptions being handled.

Write your answer below:

[14]:
```
import java.util.Scanner;

static int getQuotient(int dividend)
{
    Scanner s = new Scanner(System.in);
    System.out.println("divisor:");
    try
    {
        int divisor = Integer.parseInt(s.nextLine());
        return dividend / divisor;
    }
    catch (NumberFormatException e)
    {
        System.out.println("Error: Please enter a valid number for the divisor.␣
 ↪Returning 0.");
        return 0;
    }
    catch (ArithmeticException e)
    {
        System.out.println("Error: Cannot divide by zero. Returning 0.");
        return 0;
    }
}
```

Run your program:

[15]:
```
System.out.println(getQuotient(25));
```

```
divisor:
 0
Error: Cannot divide by zero. Returning 0.
0
```

1.3 Question 3 [1 mark]

Consider the method below, `getLength`. Identify a potential error that might occur when calling it and handle it using `try`, `catch` and `finally`. The `catch` block should print out a message indicating that the error occurred and set the value of `length` to `-1`. The `finally` block should print out a message indicating that we are in the `finally` block and return the `length`.

The return statement should only be within the `finally` block.

Write your answer below:

[16]:
```
static int getLength(String s)
{
    int length = 1;
    try
    {
        length = s.length();
    }
    catch (NullPointerException e)
    {
        System.out.println("Error: String is null. Setting length to -1.");
        length = -1;
    }
    finally
    {
        System.out.println("In finally block. Returning length.");
        return length;
    }
}
```

Run your program:

[17]:
```
System.out.println(getLength(null));
System.out.println(getLength("ABC"));
```

```
Error: String is null. Setting length to -1.
In finally block. Returning length.
-1
In finally block. Returning length.
3
```

1.4 Question 4 [1 mark]

Consider the class below, `CustomException`. It shows you how to define your own exception. Now, write your own exception.

Write your answer below:

[18]:
```
class CustomException extends Exception
{
    CustomException(String message)
    {
        super(message);
    }

    @Override
    public String getMessage()
    {
        return "Custom Exception: " + super.getMessage();
    }
}
class InvalidAgeException extends CustomException
{
  public InvalidAgeException(int age)
    {
        super("Invalid age has been provided: " + age + ". Age must be a␣
 ↪positive value.");
    }
}
```

[19]:
```
public int setAge(int age) throws InvalidAgeException
{
    if (age <= 0)
    {
        throw new InvalidAgeException(age);
    }
    System.out.println("Age set to: " + age);
    return age;
}
try
{
  int newAge = setAge(-67);
  System.out.println("New age after successful update: " + newAge);
}
catch (InvalidAgeException e)
{
  System.err.println("Error setting age: " + e.getMessage());
```

```
}
```

```
Error setting age: Custom Exception: Invalid age has been provided: -67. Age
must be a positive value.
```

1.5 Question 5 [1 mark]

Using your exception from Question 4, write a method incorporating it sensibly. If the error occurs, you should throw your exception and handle it appropriately using `try` and `catch`.

Write your answer below:

[20]:
```
public int setAge(int age) throws InvalidAgeException
{
    if (age <= 0)
    {
        throw new InvalidAgeException(age);
    }
    System.out.println("Age set to: " + age);
    return age;
}
```

Run your program:

[21]:
```
try
{
  int newAge = setAge(-67);
  System.out.println("New age after successful update: " + newAge);
}
catch (InvalidAgeException e)
{
  System.err.println("Error setting age: " + e.getMessage());
}
```

```
Error setting age: Custom Exception: Invalid age has been provided: -67. Age
must be a positive value.
```

[22]:
```
try
{
  int newAge = setAge(67);
  System.out.println("New age after successful update: " + newAge);
}
catch (InvalidAgeException e)
{
  System.err.println("Error setting age: " + e.getMessage());
}
```

```
Age set to: 67
New age after successful update: 67
```

[ ]: