University of Technology, Jamaica
School of Computing and Information Technology

OOP Principles Tutorial Eight – Exception Handling

Objective:
The objectives of this tutorial are to allow students to be able to:

- program defensively using exception handling

Class Exercise

Consider the following code (in both C++ and Java) for the method called Process():

C++ 1

```
//Process size
int Process()
{
	int ls;
	int ss;
	int ans;
	System.out.println("Large size:");
	cin >> ls;
	System.out.println("Small size:");
	cin >> ss;
	ans = ls / ss;
	return ans;
}
```

Java

```
//Process size
public int Process()
{
	Scanner inp = new Scanner(System.in);
	int ls;
	int ss;
	int ans;
	System.out.println("Large size:");
	ls = inp.nextInt();
	System.out.println("Small size:");
	ss = inp.nextInt();
	ans = ls / ss;
	return ans;
}
```

From the code above it is apparent that if ss is equal to 0 (zero) then the method will crash.

1) Use exception handling to prevent the Process() method from terminating abnormally. Only modify the Process() method.

2) Rewrite the Process() method so that the so that an exception is generated when division by

zero occurs, but that exception is declared by way of an exception specifier. Do not handle the exception in this version of Process().

3) Write a method called Analyze() that calls the Process() method. In the Analyze() method, handle any specific exception thrown in the Process() method. Re-throw any exception handled in Analyze().

4) Write a main() method that calls Analyze(). Make provision in main() to handle <u>all</u> exceptions thrown.