Montgomery College, CMSC 203
Worksheet 3
Module 11

Objectives
- Exceptions

Concept Questions
1. What is the purpose of the *throw* and *throws* keywords?

ANSW: The *throws* keyword is used to specify that a method may raise an exception during its execution. It enforces explicit exception handling when calling the method.
The *throw* keyword allows us to throw an exception object to interrupt the normal flow of the program. This is most commonly used when a program fails to satisfy a given condition:

2. What is the *finally* block and when is it executed?

ANSW: The *finally* block is always executed after the *try* block exits, whether an exception was thrown or not inside it.

3. Rewrite this multi catch block in a different way.

```
try {

} catch (FileNotFoundException | EOFException ex) {

}
```

ANSW:

```
try {

} catch (FileNotFoundException ex) {

} catch (EOFException ex) {

}
```

4. What exception will be thrown executing the following code block?

```
Integer[][] ints = { { 1, 2, 3 }, { null }, { 7, 8, 9 } };
System.out.println("value = " + ints[1][1].intValue());
```

A) `ArrayIndexOutOfBoundsExceptio`*`n`*
B) `NullPointerException`
C) Both A and B
D) Will not compile

ANSW: A

5. Consider the following two classes. What is the output of the following code?

```
public class ExampleMain {

	public static void main(String[] args) {
		Example ex = new Example();
		ex.call();
	}

}

public class Example {
	private int count = 0;

	public void call() {
		add(2);
		add(3);
		System.out.println("in call method");
		div(0);
	}

	public void add(int val) {
		count += val;
		System.out.println("in add method");

	}

	public void div(int val) {

		count /= val;
		System.out.println("in div method");
	}
}
```

Answer:
in add method
in add method
in call method
```
Exception in thread "main" java.lang.ArithmeticException
	at Example.div(Example.java:19)
	at Example.call(Example.java:8)
	at ExampleMain.main(ExampleMain.java:6)
```

6. What is a stack trace? How does the stack trace help handle exceptions?

ANSW: A stack trace provides the names of the classes and methods that were called, from the start of the application to the point an exception occurred.
It's a very useful debugging tool since it enables us to determine exactly where the exception was thrown in the application and the original causes that led to it.

7. Can we write only try block without catch and finally blocks?

ANSW: No, It shows compilation error. The try block must be followed by either catch or finally block. You can remove either catch block or finally block but not both.

8. What is the error in this code?

```
public class ExceptionHandling
{
 public static void main(String[] args)
 {
      try
      {
           int i = Integer.parseInt("abc"); //This statement
throws NumberFormatException
      }
      catch(Exception ex)
      {
           System.out.println("handles all exception types");
      }
      catch(NumberFormatException ex)
      {

      }
 }
}
```

ANSW: Second catch block is unreachable since the exception above already handles everything

9. What are the correct combinations of try /catch and finally blocks?

1)

```
try
{

}
catch(Exception ex)
{

}
```

2)

```
 try
 {

 }
 finally
 {
```

```
}
```

3)

```
try
{

}
catch(Exception ex)
{

}
finally
{

}
```

A) Only 1 and 2
B) 1, 2 and 3 are all correct
C) 2 Only
D) Only 1 and 3

Answer: B

10. What is the output of the following program?

```
class Main {
     public static void main(String args[]) {
          try {
               throw 10;
          }
          catch(int e) {
               System.out.println("Got the Exception " + e);
          }
     }
}
```

A) Exception 10 is thrown
B) Exception 0 is thrown
C) IncorrectExceptionFormatException is thrown
D) Compiler error
ANSW: D

11. What is the output of the following program?

```
public class ExceptionExamples {

      public static String concatName(String s1, String s2  )  throws
NullPointerException
      {
            if  (s1 == null || s2 == null)
                  throw new NullPointerException("Null argument");
            else
                  return s1+s2;

      }

}


public class ExceptionDriver {
      public static void someMethod(String s1, String s2) {

            try
            {
                  String s=ExceptionExamples.concatName(s1,s2);
                  System.out.println(s);
            }

            finally {
                  System.out.println("In the finally Block of someMethod ");
            }

            System.out.println("End of someMethod");
            }

      public static void main(String[] args)
      {
            try {
                  someMethod(null,"Hello");
                  someMethod("Hi","Hello");
            }
          catch (Exception e)
            {
                  System.out.println(e.getMessage());
            }
            finally
            {
                  System.out.println("In the finally Block of main ");

            }
```

```
            System.out.println("End of main!");

      }
}
```

Answer:

```
In the finally Block of someMethod
Null argument
In the finally Block of main
End of main!
```

12. What is the output of the following program?

```
public class ExceptionExamples {

      public static String concatName(String s1, String s2  )  throws
NullPointerException
      {
            if  (s1 == null || s2 == null)
                  throw new NullPointerException("Null argument");
            else
                  return s1+s2;

      }

}


public class ExceptionDriver {
      public static void someMethod(String s1, String s2) {

            try
            {
                  String s=ExceptionExamples.concatName(s1,s2);
                  System.out.println(s);
            }
            catch (Exception e)
            {
                  System.out.println(e.getMessage());
            }


            finally {
                  System.out.println("In the finally Block of someMethod ");
            }

            System.out.println("End of someMethod");
            }

      public static void main(String[] args)
      {
            try {
                  someMethod(null,"Hello");
                  someMethod("Hi","Hello");
            }

            finally
```

```
            {
                  System.out.println("In the finally Block of main ");

            }
            System.out.println("End of main!");

      }
}
```

Answer:

```
Null argument
In the finally Block of someMethod
End of someMethod
HiHello
In the finally Block of someMethod
End of someMethod
In the finally Block of main
End of main!
```