Exception Handling

Basic Exception Handling

- An **exception** is an event that occurs during the execution of a program that disrupts the normal flow of instructions.
- **Exception handling** is the process used to change the normal flow of code execution if a specified exception occurs.
- Exceptions that occur during compilation are called **checked exceptions**.

Exception	Description
`ClassNotFoundException`	The class is not found.
`IllegalAccessException`	Access to a class is denied.
`InstantiationException`	Attempt to create an object of an abstract class or an interface.
`NoSuchMethodException`	A requested method does not exist.

- **Unchecked exceptions** are exceptions that occur during execution. These are also known as **runtime exceptions**.

Exception	Description
`ArithmeticException`	Arithmetic error, such as an integer divided by 0
`ArrayIndexOutOfBoundsException`	Accessing an invalid index of the array
`ArrayStoreException`	Assigning a value to an array index that does not match the expected data type
`InputMismatchException`	Entering a value that does not match the expected data type
`NullPointerException`	Invalid use of a null reference
`NumberFormatException`	Invalid conversion of a string to a numeric format
`StringIndexOutOfBoundsException`	Accessing an invalid index (character) of a string

try, catch, and *finally*

- A ***try* block** is a block of code that might throw an exception that can be handled by a matching *catch* block.
- A ***catch* block** is a segment of code that can handle an exception that might be thrown by the *try* block that precedes it.

```
import java.util.*;
public class ExceptionSample
{
    public static void main (String[]args)
    {
        Scanner s = new Scanner(System.in);
        int dividend, divisor, quotient;
        System.out.print("Enter dividend: ");
        dividend = s.nextInt();
        System.out.print("Enter divisor: ");
        divisor = s.nextInt();
        try
        {
            quotient = dividend/divisor;
            System.out.println(dividend + " / " +
                divisor +   " = " + quotient);
        }
        catch(ArithmeticException ex)
        {
            System.out.println("Divisor cannot be 0.");
            System.out.println("Try again.");
        }
    }
}
```

- The ***getMessage()*** method can be used to determine Java's message about the exception.
  Syntax: **`System.out.println(exceptionName.getMessage());`**
- Only one (1) *try* block is accepted in a program but there can be multiple *catch* blocks.

```
                try
                {
                    System.out.print("Enter dividend: ");
                    dividend = s.nextInt();
                    System.out.print("Enter divisor: ");
                    divisor = s.nextInt();
                    quotient = dividend/divisor;
                    System.out.println(dividend + " / " +
                        divisor +   " = " + quotient);
                }
                catch(ArithmeticException ex)
                {
                    System.out.println("Divisor cannot be 0.");
                    System.out.println("Try again.");
                }
                catch(InputMismatchException ex)
                {
                    System.out.println("Invalid data type");
                    System.out.println("Please enter an integer.");
                }
            }
        }
```

- A user-defined exception is created by extending the *Exception* class.
- The ***finally* block** contains statements which are executed whether or not an exception is thrown. There can only be one (1) *finally* block after a *try-catch* structure but it is not required.

```
                try
                {
                    System.out.print("Enter dividend: ");
                    dividend = s.nextInt();
                    System.out.print("Enter divisor: ");
                    divisor = s.nextInt();
                    quotient = dividend/divisor;
                    System.out.println(dividend + " / " +
                        divisor + " = " + quotient);
                }
                catch(ArithmeticException ex)
                {
                    System.out.println("Divisor cannot be 0.");
                }
                catch(InputMismatchException ex)
                {
                    System.out.println("Invalid data type");
                }
                finally
                {
                    System.out.println("Thank you for your time.");
                }
            }
        }
```

<u>User-Defined Exceptions</u>

- A **user-defined exception** is created by extending the *Exception* class.

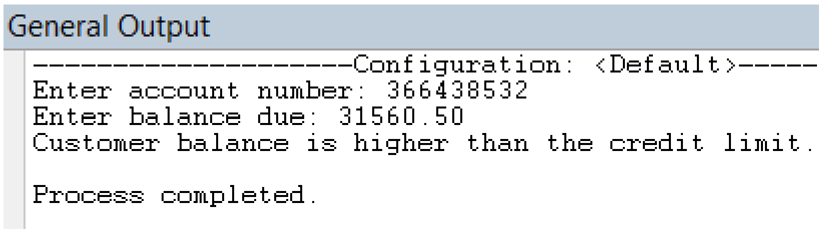
```java
public class HighBalanceException extends Exception
{
    public HighBalanceException()
    {
        super("Customer balance is higher than the credit limit.");
    }
}
```

- A **throw statement** sends an exception out of a block or a method so it can be handled elsewhere.

```java
import java.util.*;
public class CustomerAccount
{
    private Scanner s = new Scanner(System.in);
    private int acctNum;
    private double bal;
    public static double HIGH_CREDIT_LIMIT = 30000.00;

    public static void main(String[]args)
    {
        CustomerAccount c = new CustomerAccount();
        c.input();
    }
    public void input()
    {
        try
        {
        System.out.print("Enter account number: ");
        acctNum = s.nextInt();
        System.out.print("Enter balance due: ");
        bal = s.nextDouble();
        if(bal > HIGH_CREDIT_LIMIT)
            throw new HighBalanceException();
        }
        catch (HighBalanceException hbe)
        {
            System.out.println(hbe.getMessage());
        }
    }
}
```

Output:



```
General Output
--------------------Configuration: <Default>-----
Enter account number: 366438532
Enter balance due: 31560.50
Customer balance is higher than the credit limit.

Process completed.
```

References:

Baesens, B., Backiel, A. & Broucke, S. (2015). *Beginning java programming: The object-oriented approach*. Indiana: John Wiley & Sons, Inc.

Farrell, J. (2014). *Java programming, 7th edition*. Boston: Course Technology, Cengage Learning

Savitch, W. (2014). *Java: An introduction to problem solving and programming, 7th edition*. California: Pearson Education, Inc.