Session 07

Package and Exception Handling (2)

A. REQUIREMENTS

I. Write a program using ArithmeticException and InputMismatchException: Require user to enter 2 numbers then print out result of division.

- If enter wrong numbers, require user to try again.
- If division has exception divide by 0, require user to try again.

Just exit the program when the division completed.

II. Create an Exception named CustomException return a custom error message.

III. Write TestThrowException class have number field (integer type).

- Add a method named setNumber (int pNumber) to set number value. If number <= 0, throw an CustomException exception
- Add main method to test program

IV. Write a assertion program to check number > 0

B. STEPS BY STEPS

I. Using try-catch to catch some basic runtime exceptions

```
package exceptiondemo;

import java.util.InputMismatchException;
import java.util.Scanner;

/**
 *
 * @author KhanhVH@fe.edu.vn
 */
public class CatchRuntimeExceptions {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int a;
        int b;
        Scanner input = new Scanner(System.in);

        while (true) {
            try {
                System.out.print("Enter number a: ");
                a = input.nextInt();
                System.out.print("Enter number b: ");
                b = input.nextInt();

                int result = a / b;
                System.out.println("Result a div b is " + result);

                break;
            } catch (InputMismatchException ex1) {
                System.out.println("Please enter an integer number!");
            } catch (ArithmeticException ex2) {
                System.out.println("b must be different from 0!");
            } finally {
                input.nextLine();
            }
        }
    }
}
```

II. Using customize exception

CustomException.java

```
public class CustomException extends Exception {
    String message;

    public CustomException(String message) {
        this.message = message;
    }

    @Override
    public String getMessage(){
        return message;
    }
}
```

TestThrowException.java

```
public class TestThrowException {
    int number;

    void setNumber(int pNumber) throws CustomException{
        if(pNumber <= 0){
            throw new CustomException("Number must be greater then 0");
        }
        number = pNumber;
    }

    public static void main(String[] args) {
        TestThrowException obj = new TestThrowException();
        try {
            obj.setNumber(-2);
        } catch (CustomException ex) {
            Logger.getLogger(TestThrowException.class.getName())
                  .log(Level.SEVERE, null, ex);
        }
    }
}
```

III. Using assertion to test your class

```
public class TestAssertion {
    public static void main(String[] args) {
        int a;
        Scanner input = new Scanner(System.in);

        System.out.println("Please enter number a: ");
        a = input.nextInt();

        assert (a > 0 && a < 10) :
                "The number must be greater than 0 and less than 10";
        System.out.println("Number is ok!");
    }
}
```