JC2002 Java Programming - Practical 5 (Day 6)

The goal of this practical is to practice using recursion and exceptions in your Java programs.

1. ***Collatz conjecture***, a famous unproven mathematical conjecture, is stated as follows: Take any positive integer *n*. If *n* is even, generate the next number by dividing *n* by 2. If *n* is odd, generate the next number by multiplying *n* by 3 and adding 1 (i.e. 3*n*+1). After a finite number of steps, the sequence of numbers will always reach 1. Write a Java program to print Collatz sequence, starting from parameter *n*, using recursion. *Hint:* you can take the Fibonacci example in today's lecture as a starting point. Test your program with different initial values of *n*.

2. Collatz conjecture works only if the initial *n* is a positive integer. Extend your program so that it throws **IllegalArgumentException** if *n* is not valid. In addition, use **try … catch** structure to handle the exception. For example, you can print text "Please use valid n" in case exception is caught.

3. Modify the program from the previous tasks so that it does not print out the sequence but counts the number of steps required to reach the end and prints out the count after the recursion is finished, e.g., "For initial n = 21, it took 7 steps to reach 1." Test the program with different initial values of *n*, including large and small values.

4. Get back to the calculator program you implemented in the last practical. You may have noticed that if you type in invalid data, e.g., letters instead of numbers when the **nextInt()** method of **Scanner** object reads user input, the execution of the program ends as **InputMismatchException** is thrown. Add **try…catch** exception handler in **RunApp** class to handle the exception (for example, you can print a message "Invalid input" in the **catch** block).

   Note that if you want to continue the loop after handling the exception, you need to have the **try** block inside the loop where user input is read. In this case, you also need to clear the scanner buffer using nextLine() method inside the catch block, otherwise new exceptions will be thrown.

5. Add the fourth method **divide()** in the Calculator class that is invoked if user chooses division operation '/'. Add another **catch** block to handle **ArithmeticException** in method **main()** of **RunApp** that is thrown because of division by zero. You can for example print text "Division by zero".

6. Since our calculator operates using integers, it does not return fractional numbers. Define custom exception **NotDivisibleException** that is thrown in method **divide()** in case the modulo of the division operation is not zero. Add one more **catch** block to handle **NotDivisibleException** in method **main()** of **RunApp** that is thrown because the first integer is not divisible by the second one. You can for example print text "Not divisible".