Atividade

Informações

O super fatorial de um número N é definida pelo produto dos N primeiros fatoriais de N. Assim, o super fatorial de 4 é sf(4) = 1! \* 2! \* 3! \* 4! = 288.

Atividade

1. Crie uma interface e uma classe para o Cálculo do SuperFatorial. A classe deve ter um método público que recebe um parâmetro inteiro e retorna um inteiro (use big integer) representando um superfatorial. A classe pode ter vários métodos porém apenas este que recebe o número e retorna o superfatorial pode ser público.
   1. Super fatorial
      1. <http://googology.wikia.com/wiki/Superfactorial>
   2. Usar a classe BigInteger
      1. <https://www.tutorialspoint.com/java/math/java_math_biginteger.htm>
   3. Exemplo:

|  |
| --- |
| package br.edu.udesc.business;  import java.math.BigInteger;  import br.edu.udesc.exceptions.InputException;  import java.math.BigInteger;  /\*\*  \* Assinatura para o SuperFatorial  \* @author udesc  \*  \*/  public interface ISuperFatorial {    /\*\*  \* Recebe o fatorial do número e retorna o superfatorial dele  \* @param numero para o qual calcularemos o superfatorial  \* @return super fatorial  \* @throws InputException indicando que o valor digitado é inválido  \*/  public BigInteger getSuperFatorial(BigInteger numero) throws InputException;  } |

|  |
| --- |
| package br.edu.udesc.business;  package br.edu.udesc.business;  import java.math.BigInteger;  import br.edu.udesc.exceptions.InputException;  import br.edu.udesc.exceptions.NegativeValueEnteredException;  import br.edu.udesc.exceptions.ZeroEnteredException;  /\*\*  \* Implementação do super fatorial  \* @author udesc  \*  \*/  public class SuperFatorial implements ISuperFatorial {    public BigInteger getSuperFatorial(BigInteger numero) throws InputException{  if( ZERO )  return BigInteger.One;  else if( NEGATIVO )  throw new NegativeValueEnteredException();    // TODO Auto-generated method stub  return null;  }    protected BigInteger getFatoria( BigInteger numero ) {  return null;  }  } |

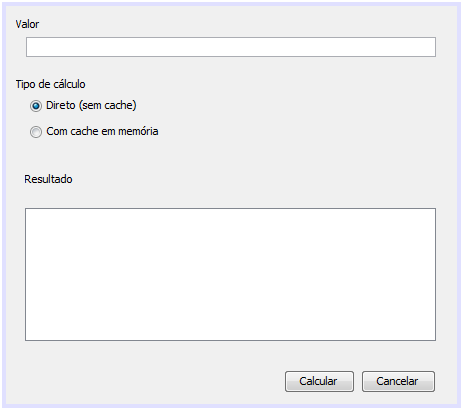
|  |
| --- |
| package br.edu.udesc.business;  import java.math.BigInteger;  import java.util.HashMap;  /\*\*  \* Antes de calcular o fatorial busca no cache  \* @author udesc  \*  \*/  public class SuperFatorialCached extends SuperFatorial {  /\*\*  \* Cache em memória para o super fatorial  \*/  private HashMap<Integer,BigInteger> cache = new HashMap();    protected BigInteger getFatoria( BigIntegernumero ) {  // procura no cache primeiro se existir retorna o valor  // se nao existir calcula e adiciona no cache  return null;  }  } |

|  |
| --- |
| package br.edu.udesc.exceptions;  /\*\*  \* Exceção gerada para quando os dados são entrados de forma errada  \* @author udesc  \*  \*/  public abstract class InputException extends Exception {  /\*\*  \* Mensagem do erro  \* @param msg  \*/  public InputException(String msg) {  super(msg);  }  } |

|  |
| --- |
| package br.edu.udesc.exceptions;  /\*\*  \* Lançada quando o valor é negativo  \* @author udesc  \*  \*/  public class NegativeValueEnteredException extends InputException {  private blic static final String NEGATIVE\_EXCEPTION = "Zero não possui fatorial";    /\*\*  \* Construtor da classe com a mensagem padrão  \*/  public NegativeValueEnteredException() {  super(NEGATIVE\_EXCEPTION);  }  } |

|  |
| --- |
| package br.edu.udesc.exceptions;  /\*\*  \* Lançada quando o valor é ZERO  \* @author udesc  \*  \*/  public class ZeroEnteredException extends InputException {  private static final String ZERO\_EXCEPTION = "Zero não possui fatorial";  /\*\*  \* Construtor da classe  \*/  public ZeroEnteredException() {  super(ZERO\_EXCEPTION);  }  } |

1. Crie testes unitários para o método público:
   1. Teste o superfatorial de um número negativo (deve gerar uma exception)
   2. Teste o superfatorial de zero
   3. Teste o superfatorial do número um
   4. Teste o superfatorial do número dez
   5. Teste o superfatorial do número 30
2. Crie uma tela (interface gráfica) com o usuário (pode usar o NetBeans para facilitar)
   1. Solicitar para o usuário o valor numérico para o qual se deseja calcular o SuperFatorial
   2. Solicitar para o usuário se deseja calcular usando cache ou não
   3. Permitir o usuário clicar em um botão que realiza o cálculo
   4. Mostrar o resultado do cálculo
   5. Exemplo de tela



1. Crie uma classe SuperFatorialCached
   1. A classe deve usar a memória para realizar cache leia sobre HashMap em Java
   2. Um valor que ainda não está no cache deve sempre ser adicionado nele
   3. Valores que já estão cache devem ser armazenados nele
2. Crie os testes unitários para sua classe SuperFatorialCached
3. Criar uma interface gráfica usando
   1. Um campo para digitar um valor para o qual o SuperFatorial será calculado
   2. Um campo para apresentar o resultado
4. Todos códigos devem ser armazenados no GitHub

## Período de desenvolvimento

* Em dupla
* Deve ser em Java
* Deve ser desenvolvido nas aulas durante a semana da computação e finalizado até dia dia 25/09