**Lista 05**

**Nome**: Arthur Rocha Amaral **Matricula**: 625034

**Questão 01:**

A classe StopWatch funciona como um cronometro e é utiliza para marcar o tempo de duração de determinada ação e pode ser pausado e zerado à qualquer instante.

**Questão 02:**

package Questao02;  
  
public class Fatorial {  
 public static long fat(int n) {  
 long soma = 1;  
 if(n!=0) {  
 for (int i = 1; i <= n; i++) {  
 soma = i \* soma;  
 }  
 }  
 return soma;  
 }  
  
 public static long fatr(int n) {  
 if (n==1||n==0) {  
 return 1;  
 } else {  
 return n \* *fatr*(n-1);  
 }  
 }  
 public static void imprfat(int n) {  
 long r = *fat*(n);  
 System.*out*.println(n+"! = "+r);  
 }  
  
 public static void imprfatr(int n) {  
 long r = *fatr*(n);  
 System.*out*.println(n+"! = "+r);  
 }  
  
 public static void main(String[] args) {  
 final int qnt = 20;  
 StopWatchNano tmp = new StopWatchNano();  
  
 for (int i = 0; i<=qnt;i++) {  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 tmp.start();  
 *imprfat*(i);  
 tmp.stop();  
 System.*out*.println("time: "+tmp.getElapsedTime()+" nano segundos");  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 }  
  
 System.*out*.println();  
 System.*out*.println("---------------------------------------------");  
 System.*out*.println();  
  
 for (int i = 0; i<=qnt;i++) {  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 tmp.start();  
 *imprfatr*(i);  
 tmp.stop();  
 System.*out*.println("time: "+tmp.getElapsedTime()+" nano segundos");  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 }  
 }  
}

**Questão 03:**

package Questao03;  
  
import Questao02.StopWatchNano;  
  
public class Fibonacci {  
 private static long fibonacciRec(int n) {  
 if (n < 2) {  
 return n;  
 } else {  
 return *fibonacciRec*(n-1) + *fibonacciRec*(n-2);  
 }  
 }  
  
 public static long fibonacciRec(int n, boolean imprime) {  
 long r = *fibonacciRec*(n);  
 if (imprime)  
 System.*out*.println("Fibonacci("+n+") = "+r);  
 return r;  
 }  
  
 private static long fibonacci(int n) {  
 int f = 0;  
 int ant = 0;  
  
 for (int i = 1; i <= n; i++) {  
 if (i == 1) {  
 f = 1;  
 ant = 0;  
 } else {  
 f += ant;  
 ant = f - ant;  
 }  
 }  
 return f;  
 }  
  
 public static long fibonacci(int n, boolean imprime) {  
 long r = *fibonacci*(n);  
 if (imprime)  
 System.*out*.println("Fibonacci("+n+") = "+r);  
 return r;  
 }  
  
 public static void main(String[] args) {  
 final int MAX = 46;  
 StopWatchNano tmp = new StopWatchNano();  
  
 for (int i = 0; i < MAX + 1; i++) {  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 tmp.start();  
 Fibonacci.*fibonacci*(i, true);  
 tmp.stop();  
 System.*out*.println("time: " + tmp.getElapsedTime() + " nano segundos");  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 }  
  
 System.*out*.println();  
 System.*out*.println("---------------------------------------------");  
 System.*out*.println();  
  
 for (int i = 0; i < MAX + 1; i++) {  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 tmp.start();  
 Fibonacci.*fibonacciRec*(i, true);  
 tmp.stop();  
 System.*out*.println("time: " + tmp.getElapsedTimeMili() + " mili segundos");  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 }  
 }  
}  
  
*/\*\* Maior fator para long Fibonacci(46) = 1836311903.  
 \* Não aconteceu nenhum erro.  
 \*  
 \* Recursivo  
 \* ---------------------------  
 \* Fibonacci(46) = 1836311903  
 \* time: 8339 mili segundos  
 \* ---------------------------  
 \*  
 \* Iterativa  
 \* ---------------------------  
 \* Fibonacci(46) = 1836311903  
 \* time: 57800 nano segundos  
 \* ---------------------------  
 \*/*

**Questão 04:**

package Questao04;  
  
import Questao02.StopWatchNano;  
  
public class Multiplicacao {  
 public static long multRec(long x, long n) {  
 if (n==1) {  
 return x;  
 } else {  
 return x + *multRec*(x,n-1);  
 }  
 }  
  
 public static long mult(long x, long n) {  
 long soma = 0;  
 for (int i = 0; i<n;i++)  
 soma += x;  
 return soma;  
 }  
  
 public static void main(String[] args) {  
 StopWatchNano tmp = new StopWatchNano();  
 tmp.start();  
 System.*out*.println("mult(3,715827881) = " + *mult*(3, 715827881));  
 tmp.stop();  
 System.*out*.println("tmp.getElapsedTime() = " + tmp.getElapsedTimeMili());  
  
 System.*out*.println();  
 System.*out*.println();  
  
 tmp.start();  
 System.*out*.println("multRec(3,715827881) = " + *multRec*(3, 715827881));  
 tmp.stop();  
 System.*out*.println("tmp.getElapsedTime() = " + tmp.getElapsedTimeMili());  
 }  
 */\*\* Iterativo  
 \* -----------------------------------  
 \* mult(3,715827881) = 2147483643  
 \* tmp.getElapsedTime() = 300  
 \*  
 \*  
 \* Recursivo  
 \* -----------------------------------  
 \* Lança um erro de SrackOverflowError  
 \* ou seja estoura a pilha de execução  
 \*/*}

**Classe adicional StopWatchNano:**

package Questao02;  
  
public class StopWatchNano {  
  
 private long startTime = 0;  
 private long stopTime = 0;  
 private boolean running = false;  
  
  
 public void start() {  
 this.startTime = System.*nanoTime*();  
 this.running = true;  
 }  
  
  
 public void stop() {  
 this.stopTime = System.*nanoTime*();  
 this.running = false;  
 }  
  
  
 //elaspsed time in nanoseconds  
 public long getElapsedTime() {  
 long elapsed;  
 if (running) {  
 elapsed = (System.*nanoTime*() - startTime);  
 } else {  
 elapsed = (stopTime - startTime);  
 }  
 return elapsed;  
 }  
  
  
 //elaspsed time in seconds  
 public long getElapsedTimeSecs() {  
 long elapsed;  
 if (running) {  
 elapsed = ((System.*nanoTime*() - startTime) / 1000000000);  
 } else {  
 elapsed = ((stopTime - startTime) / 1000000000);  
 }  
 return elapsed;  
 }  
  
  
 //elaspsed time in miliseconds  
 public long getElapsedTimeMili() {  
 long elapsed;  
 if (running) {  
 elapsed = ((System.*nanoTime*() - startTime) / 1000000);  
 } else {  
 elapsed = ((stopTime - startTime) / 1000000);  
 }  
 return elapsed;  
 }  
}