

2. Analizador de código.

2.1 Analizadores de código

Os analizadores de código permiten revisar o código e obter informes que axudarán ao programador a optimizalo. Poden ser estáticos ou dinámicos.

Os analizadores estáticos permiten realizar a análise sen executar o código. Están enfocados normalmente á validación e optimización do código.

Os analizadores dinámicos permiten realizar a análise executando o código nun procesador real ou virtual e obtendo un informe de rendemento.

Criterios de valoración de un analizador

- Tipo de licenza e custo.
- Usabilidade. Deberán de valorarse a axuda e a documentación, o tempo de instalación, o tempo de configuración do contorno, as actualizacións e a facilidade para interpretar os resultados.
- Eficiencia. Valoraranse o tempo de execución e o consumo de recursos.
- Extensibilidade. Valorarase a posibilidade de engadir regras facilmente cunha linguaxe sinxela (XPath, java, JML).
- Precisión dos resultados obtidos.

Analizadores estáticos



Tarefa 1. Procura de analizadores estáticos de código no mercado e comparativas.

Analizadores dinámicos

Os analizadores dinámicos miden en tempo de execución o comportamento dun programa co fin de determinar o uso que fai dos recursos do sistema como CPU ou memoria ou os tempos de execución, obtendo un informe que permitirá ao programador optimizar o seu código.

Os analizadores dinámicos actúan durante a execución do código que se pretende analizar polo que é importante que:

- Elíxase un grupo de datos de entrada que permita obter resultados fiables para a análise.
- Minimícese o efecto que poden ter na execución as ferramentas ou instrumentos utilizados na execución.



Tarefa 2. Procura de analizadores dinámicos de código no mercado e comparativas.

Analizar dinámicamente código java con netbeans

Introdución

NetBeans IDE posúe o módulo NetBeans profiler que provee a NetBeans dunha funcionalidade completa para o análise dinámico de código Java que inclúe análise da CPU, memoria e subprocesos así como monitorización JVM básica permitindo aos programadores ser máis produtivos na solución dos problemas de memoria ou no rendemento da aplica-

ción. Pódense analizar proxectos Java SE, aplicacións web, proxectos Java EE projects, proxectos Java Freeform e módulos NetBeans.

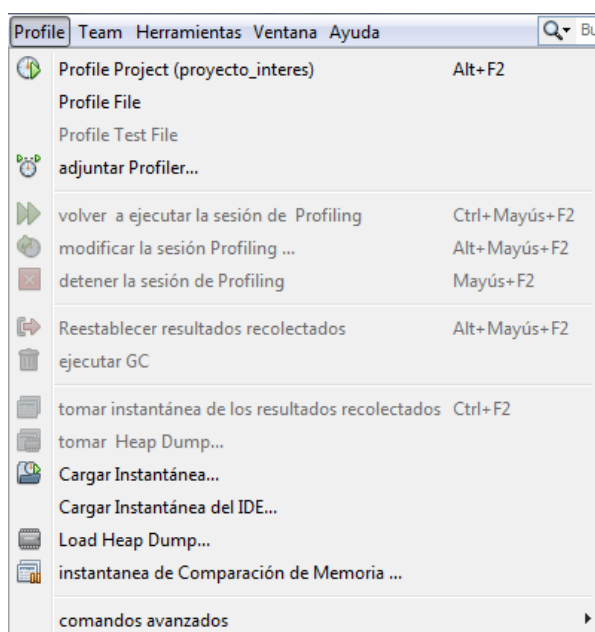
Esta análise permitirá identificar as instrucións do código que supoñen una maior carga (en tempo, memoria, recursos, etc.) para a aplicación, para que o programador optimice ese código e evite esa sobrecarga actuando en consecuencia.

É recomendable que no momento de realizar a análise non se estea executando ningunha outra aplicación no equipo xa que entón os resultados do análise de rendemento poden non ser exactos.

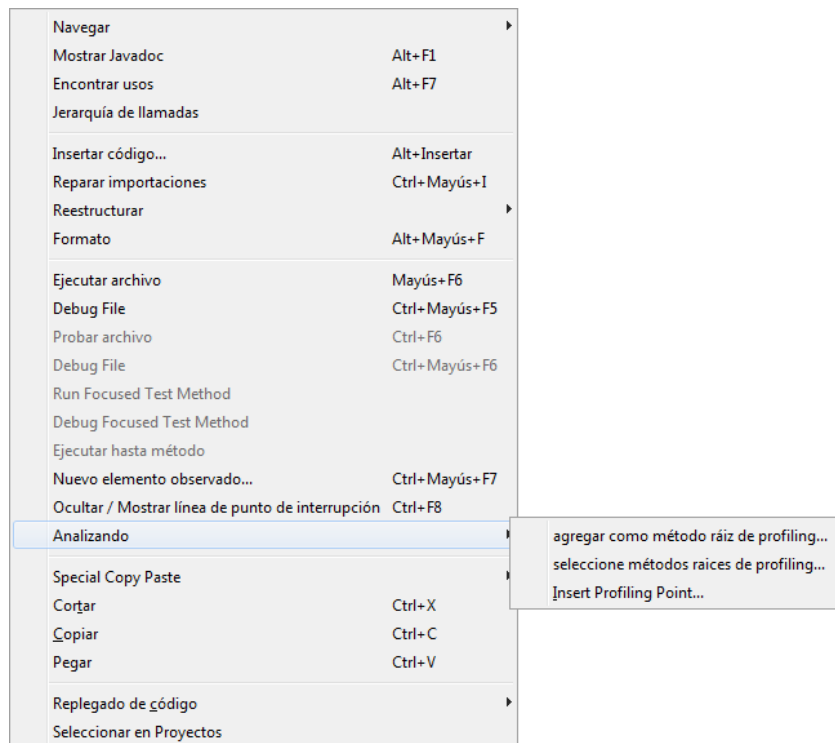
Os pasos para realizar unha análise son:

- Definir o elemento do proxecto sobre o que se vai facer a análise, seleccionar o tipo de medición que se quere realizar e as condicións adicionais que se necesiten.
- Executar a análise.
- Emitir resultados. NetBeans informa sobre as partes do código que están consumindo máis recursos do sistema durante a execución, é dicir, informa sobre a sobrecarga (overhead) na execución do código.

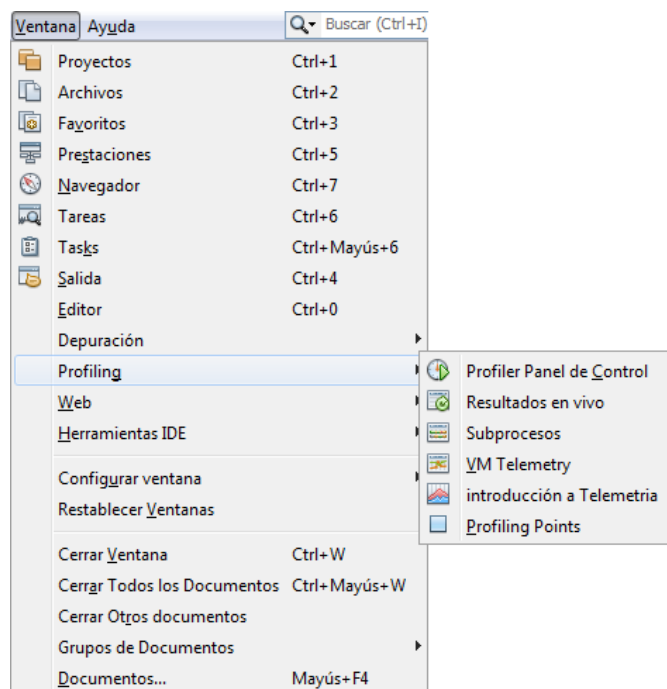
A opción *Profile* do menú principal permite xestionar a análise. A opción *Profile->Profile Main Project* do menú principal permite analizar o proxecto principal.



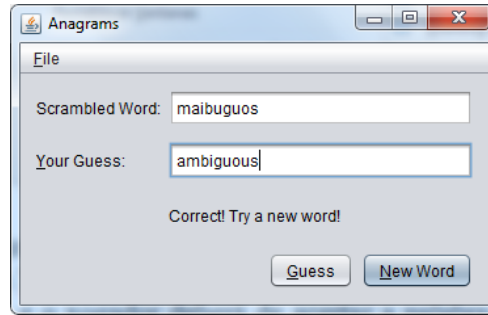
Nalgúns casos, poderanse realizar operación de análise directamente sobre un elemento do código facendo clic dereito sobre o elemento, elixindo *Analizando* e definindo ou seleccionado o elemento a analizar.



Ao longo do proceso de análise e consulta de resultados manéxanse distintas ventás que de non estar visibles poden activarse na opción *Ventana* do menú principal:

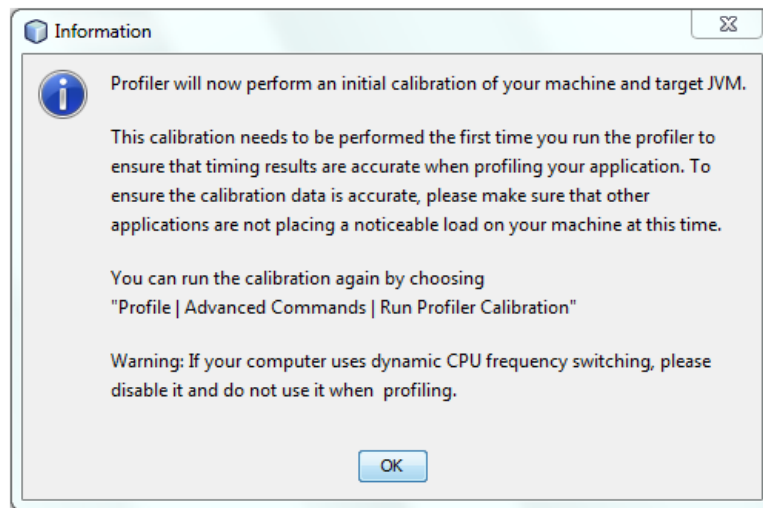


Neste documento utilizarase como proxecto *AnagramGame* de Java que se subministra como exemplo con NetBeans. Cando se executa este proxecto ábrese unha ventá cunhas letras descolocadas e o xogador deberá de acertar a palabra correcta formada con esas letras. Por exemplo, na seguinte execución acertaríase coa palabra *ambiguous*:

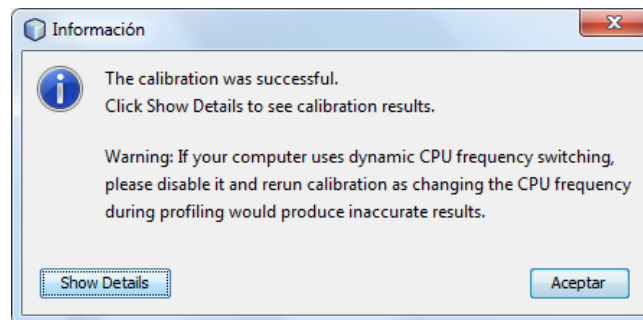


Calibrado inicial do equipo

A primeira vez que se fai unha análise de rendementos, o IDE avisa da necesidade de realizar a calibración inicial do equipo e a plataforma Java para que os resultados sexan precisos.



Prémese en OK para empezar a realizar a calibración:



Sobre a frecuencia dinámica da CPU haberá que consultar a información sobre a CPU do equipo. Por exemplo, na páxina de Intel:

<http://www.intel.com/support/sp/processors/sb/CS-029908.htm> aparece:

“¿Qué es frecuencia dinámica y cómo funciona?

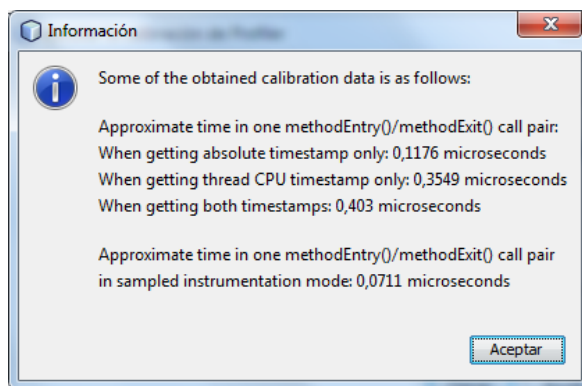
Frecuencia dinámica funciona muy similar al de la tecnología Intel Turbo Boost en forma dinámica que proporciona un impulso al desempeño de gráficos cuando se están realizando las tareas con uso intensivo de gráficos. Como la tecnología Intel Turbo Boost, debe haber margen de ampliación de energía y temperatura disponible en orden para que funcione.

¿Cómo puedo activar frecuencia dinámica?

Frecuencia dinámica se activa automáticamente en la mayoría de los sistemas, por lo

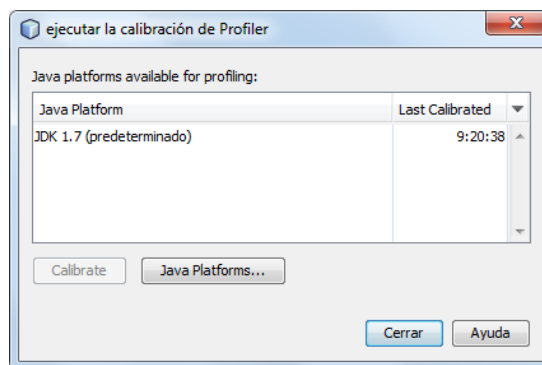
que no se requiere la intervención del usuario. ”

Pódense ver os resultados da calibración premendo en *Show Details*:



Normalmente a calibración só debe de realizarse unha vez. Sen embargo, deberá de volver a realizarse se se fan cambios substanciais na configuración da máquina que afecten ao rendemento da mesma. Para volver a realizar a calibración elíxese no menú principal:

Profile->Comandos avanzados->ejecutar la calibración de Profiler




Selecciónase a plataforma Java a calibrar e prémese en *Calibrate*.

Sesión de análise

Para realizar unha análise dinámica de código Java con NetBeans hai que seguir os seguintes pasos:

- Iniciar sesión de análise elixindo a opción *Profile->Profile Main Project* do menú principal ou facendo clic dereito sobre o nome do proxecto na ventá de proxectos e elixindo *Profile*.
- Elixir a tarefa de análise a realizar. NetBeans subministra unha serie de tarefas predefinidas que normalmente son suficientes e son: supervisión da aplicación (*Monitor*), análise do rendemento (*CPU*) e análise da memoria (*Memory*) que poderán ser axustados para diminuír a sobrecarga na análise ou facer máis pequena a información que emite como resultado.
- Configurar a tarefa, é dicir, indicar tódalas condicións adicionais que se necesiten. Sobre estas tarefas pódense crear e gardar tarefas novas personalizadas e baseadas nelas (*Create Custom...*) que estarán sempre accesibles na propia caixa de diálogo das tarefas, riba da opción *Create Custom...*, e que por tanto poderán ser facilmente executadas máis tarde. Tamén se poden configurar as tarefas coas opcións básicas ou coas avanzadas; para pasar dunhas á outras faise clic en *Advanced*.
- Executar a tarefa de análise. Isto ten como consecuencia a execución do código Java, a apertura da ventá *Analizador* e a xeración de resultados como por exemplo informe das

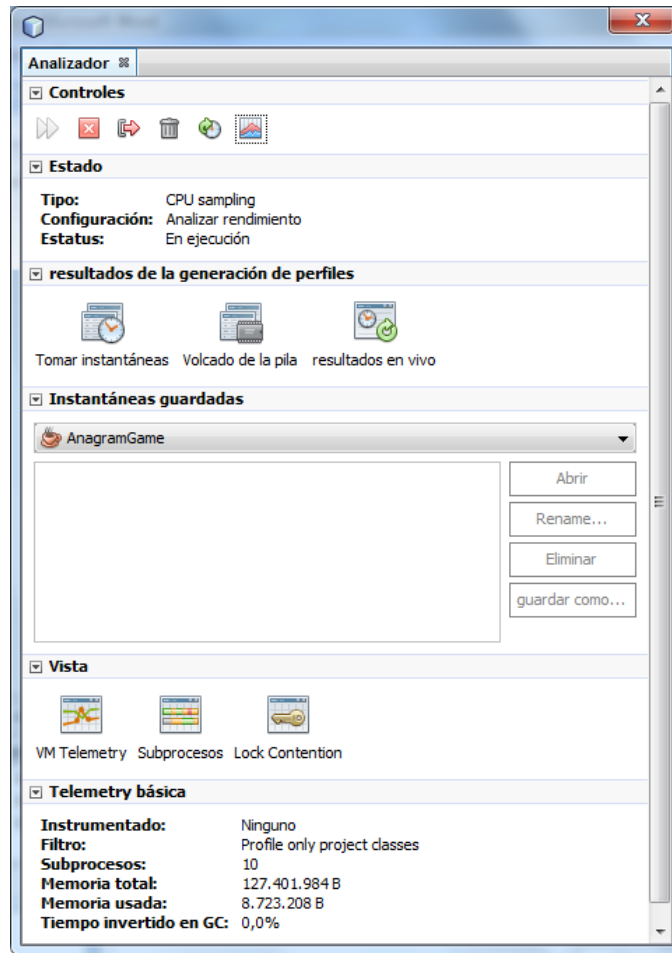
partes do código que están consumindo máis recursos do sistema durante a execución, é dicir, informe sobre a sobrecarga (*overhead*) na execución do código.

- Finalizar a sesión de análise porque finaliza a execución do código Java ou forzar a finalización da análise premendo na icona .

Ventá Analizador

Contén as seguintes partes:

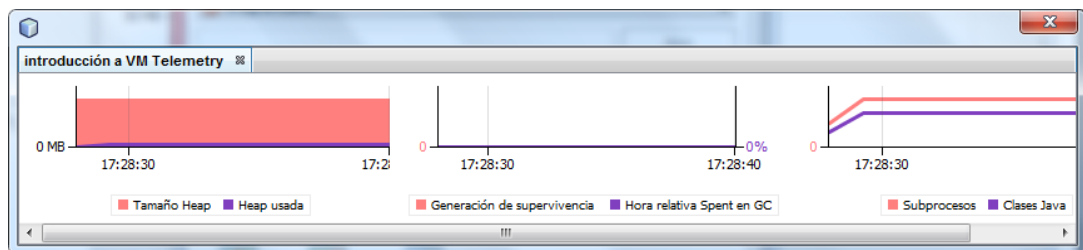
- Controles: operacións a realizar sobre a sesión de análise. As 5 primeiras iconas de control tamén están dispoñibles no menú principal na opción *Profile* e utilízanse respectivamente para: volver a executar a última sesión de análise, deter a sesión actual, *Reset results* (descartar os resultados acumulados na memoria intermedia), executar o recolector de lixo e modificar a configuración da sesión actual. A 6ª icona permite acceder á ventá VM Telemetry.
- Estado: información sobre o estado da sesión.
- Resultados de la generación de perfiles: permite realizar certas operacións durante a sesión como ver resultados en vivo e restablecer os resultados obtidos (activar a supresión dos datos acumulados no búfer. Neste caso, os resultados en vivo desaparecen ata a próxima actualización de resultados automática ou manual). Na icona *Tomar instantáneas* pódese tomar instantánea dos resultados obtidos ou tomar instantánea de memoria dinámica.
- Instantáneas gardadas: para xestionar as instantáneas que previamente foron gardadas na localización por defecto ou cargar instantáneas que non están gardadas na localización por defecto.
- Vista: para acceder a certos informes ou vistas da sesión cando estean permitidas dentro desa sesión.
- Basic Telemetry: información sobre telemetría da sesión.



Ventá de VM Telemetry

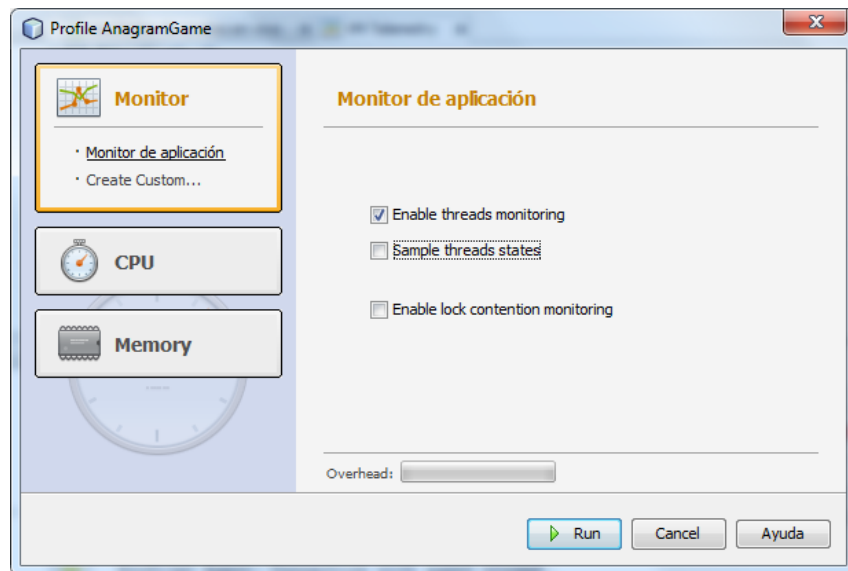
Esta ventá de VM Telemetry pode ser vista en calquera das diferentes tarefas de análise e permite ter unha vista dos datos de monitorización en tempo real: utilización da pila e memoria, xeracións vivas e tempo de uso do GC, clases cargadas e subprocesos.

Pódese colocar o cursor sobre unha das gráficas, para ter máis información en forma de texto e pódese facer dobre clic sobre unha das gráficas, e abrírase na pestana *VM Telemetry* a mesma gráfica pero con máis detalle no tempo.

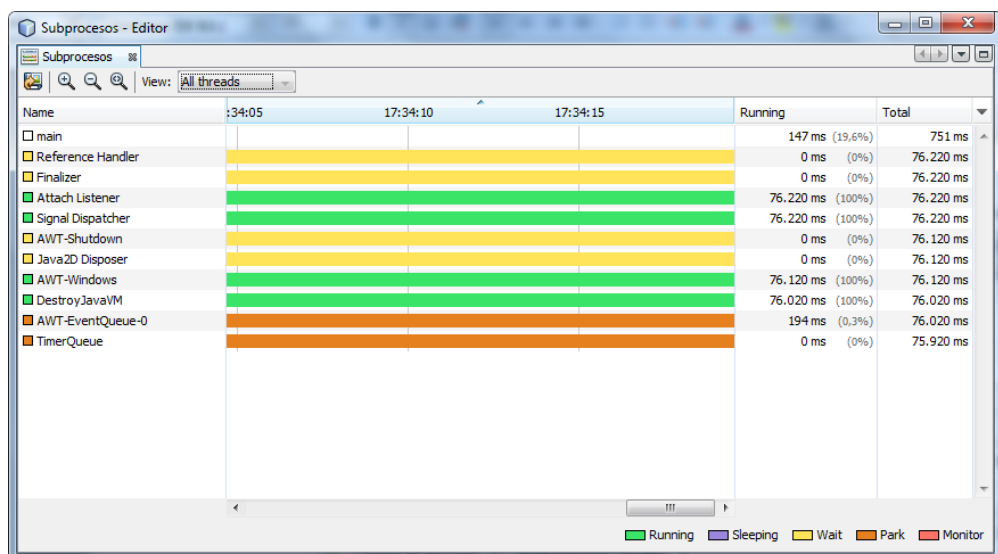


Monitor

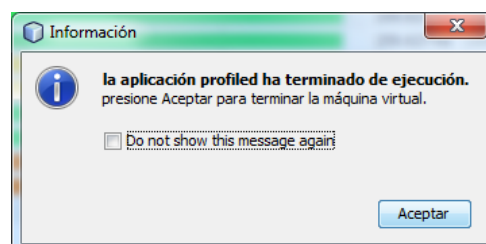
Permite supervisar a aplicación e obter información de alto nivel sobre as propiedades do código executable incluíndo uso de memoria e subprocesos fío se se desexa. Como se ve en *Overhead* esta tarefa sobrecarga pouco.



Cando se preme en Run, lánzase a aplicación, e ábrese a pestana de *Subprocesos* e as ventás de *Analizador* e a de *VM Telemetry*.

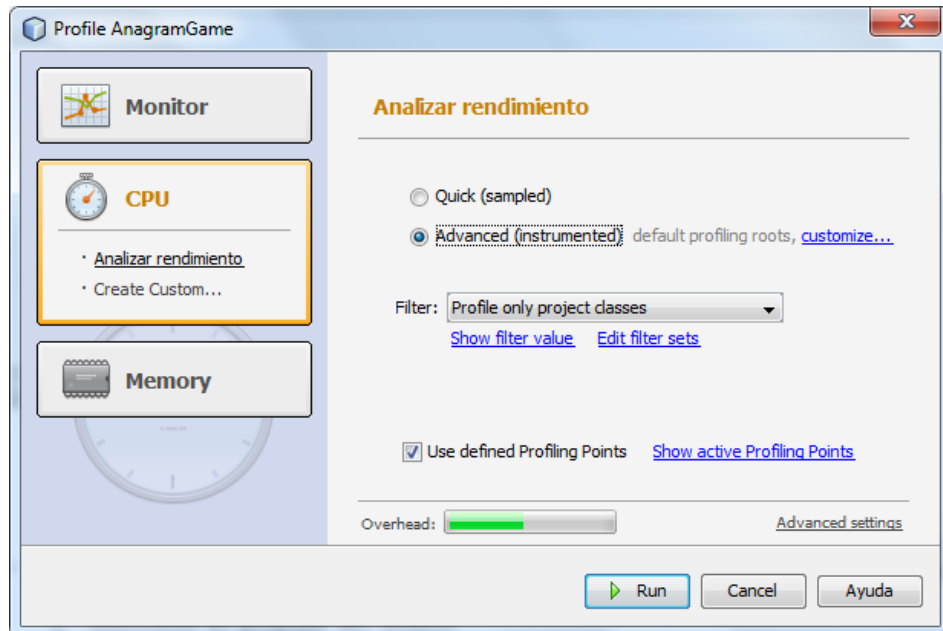


Finaliza a sesión de análise de forma normal cando finaliza a execución da aplicación *AnagramGame*; entón aparece unha ventá de información indicando que o analizador finalizou a execución e solicitando que se termine a sesión coa máquina virtual de Java.



CPU

Permite analizar o rendemento do código sobre a CPU en tempo real e obtén datos detallados incluíndo o tempo de duración da execución dos métodos e o número de veces que son invocados. Pódese elixir entre analizar a aplicación enteira ou parte dela.



De elixir analizar o rendemento de toda a aplicación (*Quick*) analizaranse tódolos métodos da aplicación. O IDE rexistra o instante de entrada nun método e o de saída e por tanto pode calcular o tempo real de execución. Para algunha aplicación analizar o rendemento de toda a aplicación non sería recomendable debido á sobrecarga que se xeraría, chegando incluso a romper a execución.

No caso de elixir analizar o rendemento de parte da aplicación (*Advanced*), débense de definir un ou máis métodos denominados métodos raíz. Un método raíz é un método, clase ou paquete do código fonte sobre o que se centrará a análise, é dicir, os datos de rendemento recolleranse cando un subproceso da aplicación entre ou saia dun método raíz. Para especificar os métodos raíz pódese facer clic en *customize...* na ventá anterior ou facer clic dereito sobre o nome do método no código fonte e elixindo a opción *Analizando >seleccione métodos raíces de profiling*. En ambos casos abrírase a caixa de diálogo *seleccione los métodos raíz*. Elixir métodos raíz reduce a sobrecarga.

Pódense establecer filtros para limitar o código a analizar. Por defecto son posibles os seguintes filtros: Todas as clases, solo clases de proxecto, clases do proxecto e subproxectos, filtro rápido e excluír clases de Java básico.

Máis adiante neste documento verase como establecer puntos de análise definidos.

Pódese observar o nivel de sobrecarga na gráfica verde de Overhead.

Por exemplo, analizar só os métodos da clase `WordLibrary.java` do proxecto `AnagramGame` o proceso a seguir é:

- Seleccionar *Profile* - > *Profile Main Project* do menú principal.
- Seleccionar *CPU*.
- Seleccionar *Advanced*.

Clic en *customize...* para definir un método raíz. Aparece a ventá *seleccione los métodos raíz* na que se premerá para agregar o método raíz dende o proxecto.