Shri G.S. Institute of Technology and Science

INDORE

Brick Braker Game

Mini Project

Submitted by:Nandini Patel

Enrollment no: 0801CS221095

INTRODUCTION

1. Brick Breaker is a classic arcade game that has entertained players for decades. It's also known by other names like Breakout or Arkanoid in various versions. In Brick Breaker, the player controls a paddle or platform at the bottom of the screen, and their goal is to break a wall of bricks located at the top of the screen using a ball. The game is known for its simple yet addictive gameplay and has been a staple of the gaming world since the late 1970s.

2. Brick Breaker is a timeless and enjoyable game that challenges players' reflexes and hand-eye coordination. Its simplicity and addictive nature have made it a popular choice for both casual and serious gamers. Over the years, many variations and adaptations of the game have been developed, each adding unique twists and features while staying true to the core gameplay concept.

Introduction to how Brick Braker is typically played :

- **Objective :** The main objective of Brick Breaker is to clear all the bricks from the screen.
- **Paddle Control:** Players control a paddle that moves horizontally along the bottom of the screen. The paddle is used to bounce the ball back up towards the bricks.
- **Ball and Bricks:** A ball is launched from the paddle, and it bounces off the walls and bricks. When the ball hits a brick, the brick disappears.
- **Game Over :**The game is typically over if the ball falls below the paddle and off the screen.
- **Scoring:** Players earn points for breaking bricks.

PROJECT OVERVIEW

1. **Language used:** Java
2. **Libraries used:**
   (a) For GUI Framework - Swing and AWT(Abstract window toolkit)
   (b) For Events - awt.Event

3. **Classes:**
   - **Jframe :** It provides a window in which you can add various GUI components and controls.The methods of Jframe class :
     1 setBounds(): It is used to set the position and dimensions of a JFrame. location (x, y) and size (width, height) of the frame
     2 setTitle() : Sets the title of the frame.
     3 setResizable(): Determines whether the frame can be resized by the user.
     4 setVisible(): Sets the visibility of the frame. Setting it to true makes the frame visible, and setting it to false hides the frame.
     5 setDefaultCloseOperation(): Defines what happens when the user closes the frame.
     6 add(): Adds a component (e.g., JPanel, JButton, JLabel) to the frame.
   - **Color** :The Color class in the Java Swing library is used to represent and manipulate colors in graphical user interfaces .Methods are as follows:
     1 setColor(): it sets of color of background or of any shape.

- **Font** :is used to specify the font family, style, and size for rendering text in graphical user interfaces. Methods are as follows.

  1 setFont() : to set the font of text.

- **Graphics** : The Graphics class in the Java AWT library is used for drawing and rendering graphics on various graphical components and surfaces in a Java application.

  1 fillRect(): Fills a rectangle.

  2 fillOval() : Fills an oval.

  3 drawString() :Renders text at the specified position.

  4 dispose() : Releases any system resources used by the Graphics object when finished with it.

  5 paint(Graphics g) : You can use the Graphics object provided as a parameter to draw shapes, text, and images on the component.

- **Graphics2D** :Graphics2D extends the graphics class which focuses on geometry and color managment.

- **Rectangle** : It is a fundamental class for defining and manipulating rectangles in graphical applications.

  1 Rectangle(int x, int y, int width, int height): Creates a Rectangle object with the specified position (x, y) and dimensions (width and height).

  2 Rectangle.intersects : whether two rectangles will intersect or not.

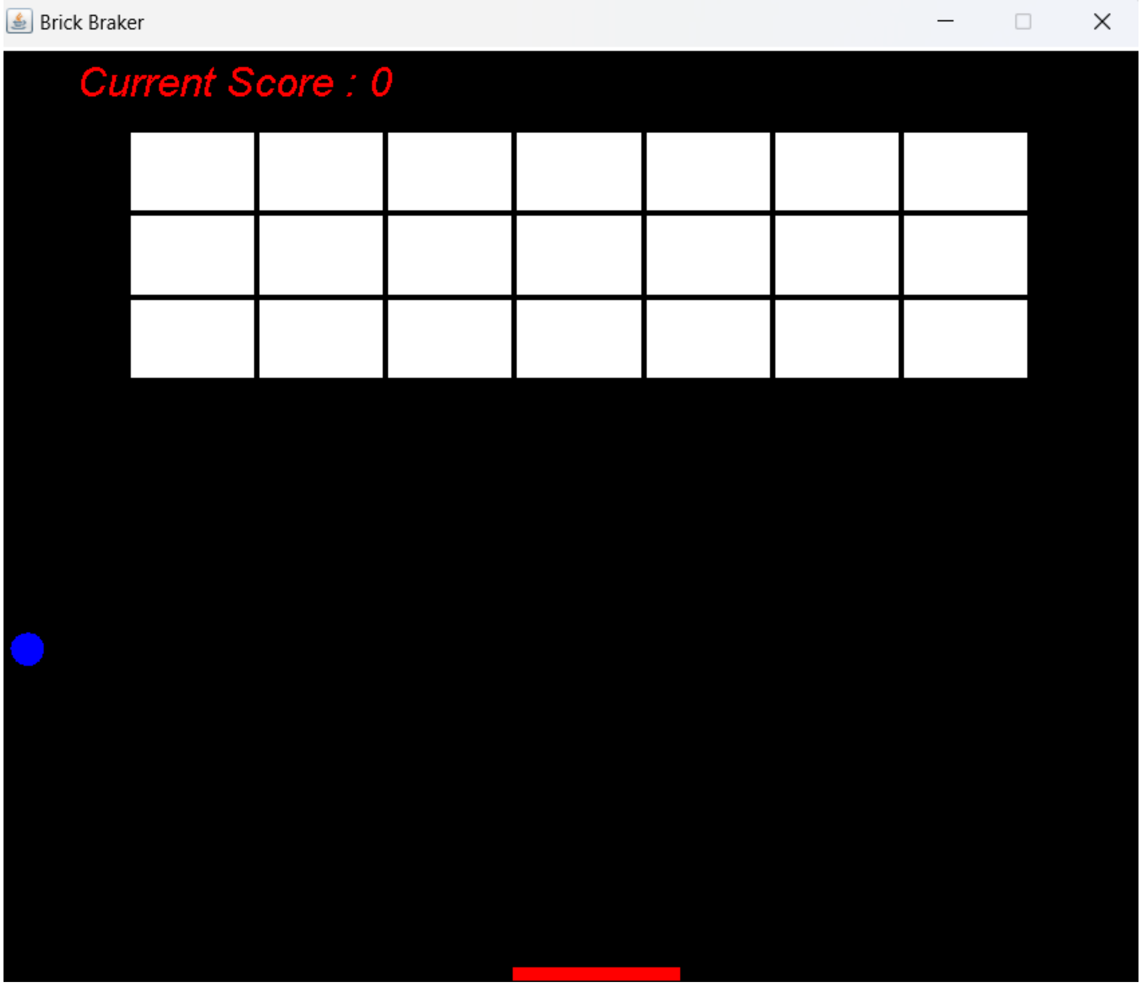
- **Action Event** : The ActionEvent class in Java is part of the java.awt.event package.The ActionEvent class is primarily used to respond to user actions and invoke appropriate event handlers.

- **Action listener interfaces**:These interfaces provide a mechanism for responding to user actions.
  1 actionPerformed() : action performed method is called to perform an action.
- **Key Event** :The KeyEvent's object contains information about the event, such as the key character and modifier keys.

1 getKeyCode() : Returns the integer keyCode associated with the key in this event.

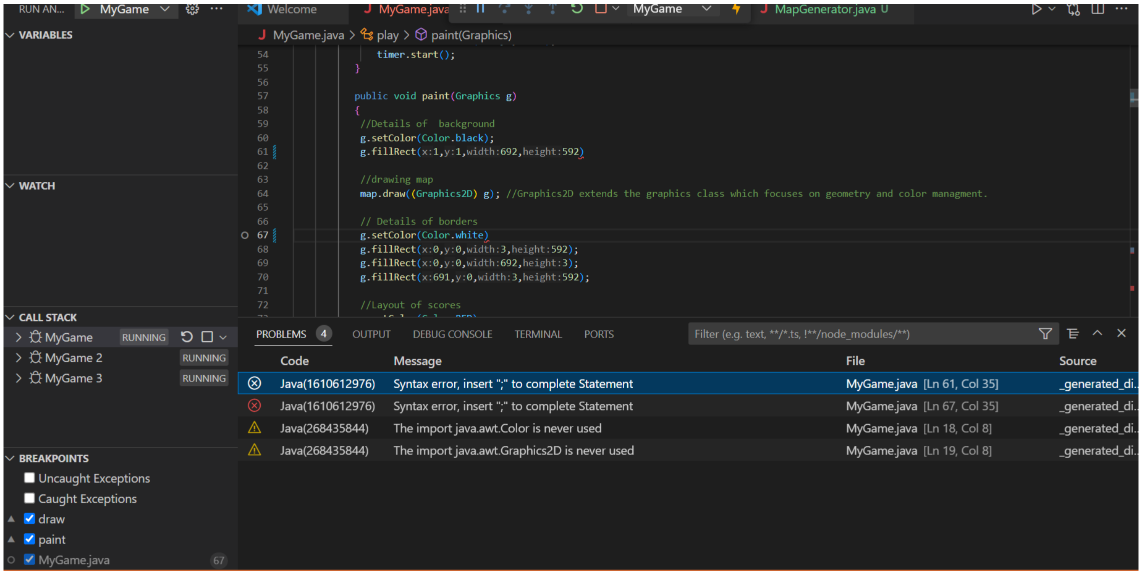
- **Key Listener Interfaces** : It is commonly used to capture and respond to key presses and key releases.
  1 keyPressed(KeyEvent e):This method is called when a key is pressed down. It is triggered as soon as the key is pressed and held.
  - **Timer** : generate periodic events or actions.
    1 start() :Starts the Timer, causing it to start sending action events to its listeners.
  - **JPanel** : JPanel provides a blank canvas, or a region within a GUI, where you can add other Swing components, draw custom graphics, or organize and group other components.
  - **BasicStrokes**: is used to define the basic characteristics of the strokes used for drawing lines and shapes.
  - **MapGenerator** :It generates the layout of bricks.
  - **MyGame** : IT is the main class of the game which implements the features of Jframe class and load all the classes.

OUTPUT



DEBUGGING

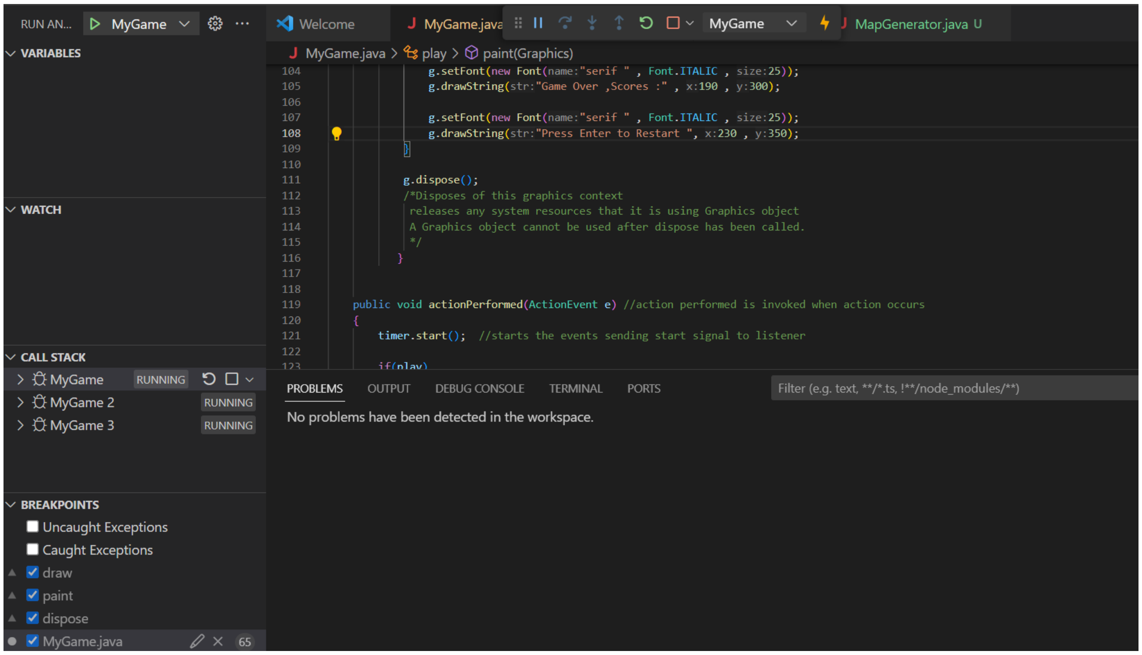
Before Debugging:



```
            timer.start();
        }

        public void paint(Graphics g)
        {
         //Details of  background
         g.setColor(Color.black);
         g.fillRect(x:1,y:1,width:692,height:592)

         //drawing map
         map.draw((Graphics2D) g); //Graphics2D extends the graphics class which focuses on geometry and color managment.

         // Details of borders
         g.setColor(Color.white)
         g.fillRect(x:0,y:0,width:3,height:592);
         g.fillRect(x:0,y:0,width:692,height:3);
         g.fillRect(x:691,y:0,width:3,height:592);

         //Layout of scores
```

	Code	Message	File	Source
⊗	Java(1610612976)	Syntax error, insert ";" to complete Statement	MyGame.java [Ln 61, Col 35]	_generated_di..
⊗	Java(1610612976)	Syntax error, insert ";" to complete Statement	MyGame.java [Ln 67, Col 35]	_generated_di..
⚠	Java(268435844)	The import java.awt.Color is never used	MyGame.java [Ln 18, Col 8]	_generated_di..
⚠	Java(268435844)	The import java.awt.Graphics2D is never used	MyGame.java [Ln 19, Col 8]	_generated_di..

After Debugging:

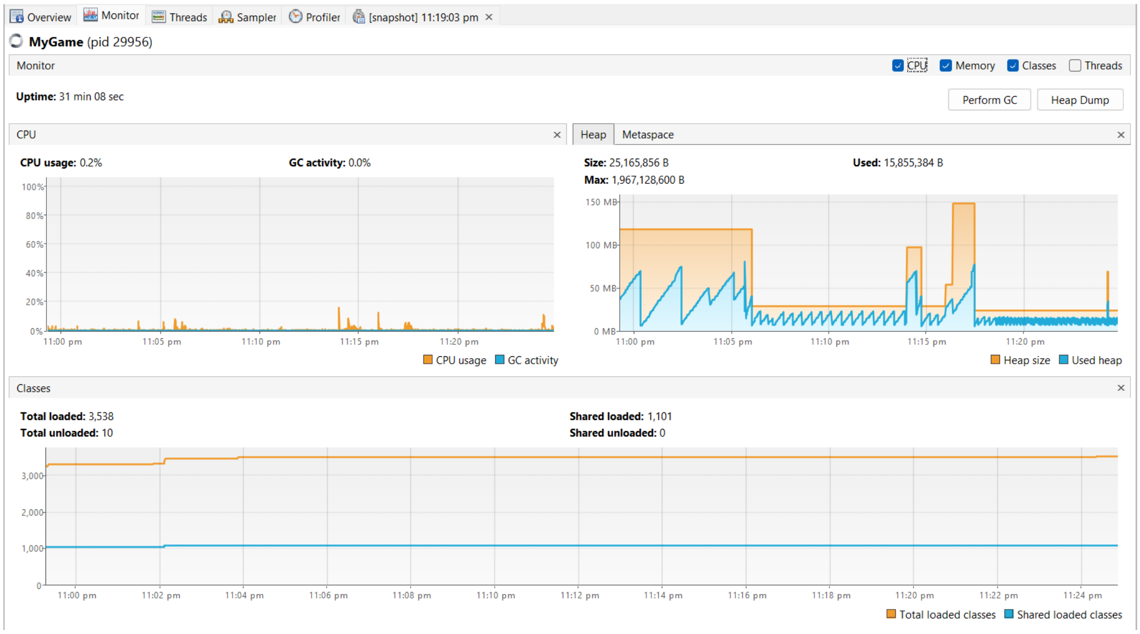


```
                g.setFont(new Font(name:"serif " , Font.ITALIC , size:25));
                g.drawString(str:"Game Over ,Scores :" , x:190 , y:300);

                g.setFont(new Font(name:"serif " , Font.ITALIC , size:25));
                g.drawString(str:"Press Enter to Restart ", x:230 , y:350);
            }

            g.dispose();
            /*Disposes of this graphics context
             releases any system resources that it is using Graphics object
             A Graphics object cannot be used after dispose has been called.
             */
           }


    public void actionPerformed(ActionEvent e) //action performed is invoked when action occurs
    {
        timer.start();  //starts the events sending start signal to listener

        if(play)
```

No problems have been detected in the workspace.

PROFILLING

VisualVm tool: : It is used as a profiller tool for java which is already installed in jdk.



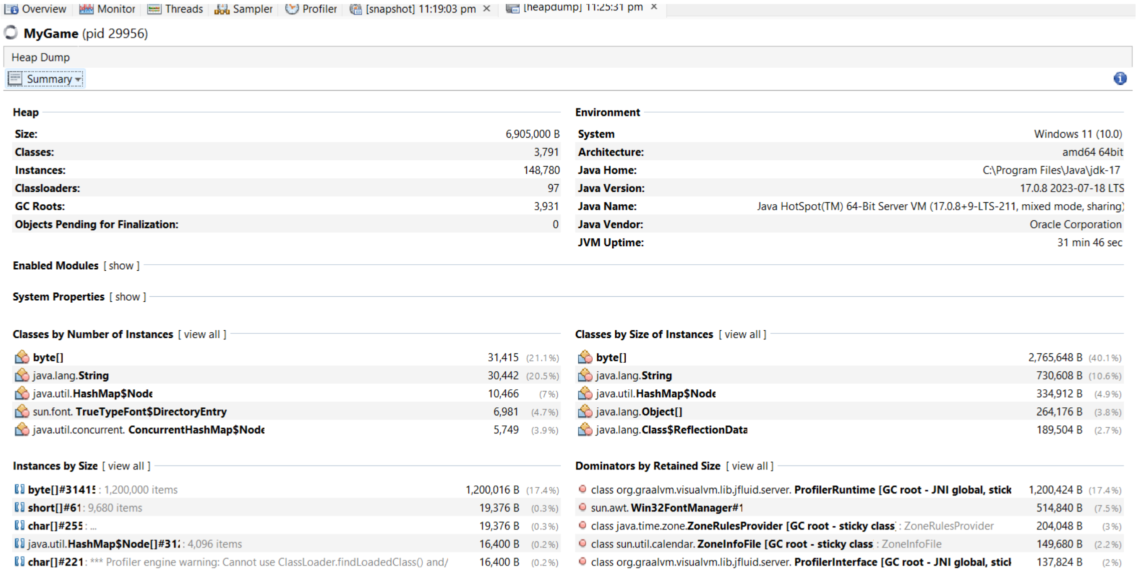
MyGame (pid 29956)

Uptime: 31 min 08 sec

CPU usage: 0.2% **GC activity:** 0.0%

Size: 25,165,856 B **Used:** 15,855,384 B
Max: 1,967,128,600 B

Total loaded: 3,538 **Shared loaded:** 1,101
Total unloaded: 10 **Shared unloaded:** 0



MyGame (pid 29956)

Heap

Size:	6,905,000 B
Classes:	3,791
Instances:	148,780
Classloaders:	97
GC Roots:	3,931
Objects Pending for Finalization:	0

Environment

System	Windows 11 (10.0)
Architecture:	amd64 64bit
Java Home:	C:\Program Files\Java\jdk-17
Java Version:	17.0.8 2023-07-18 LTS
Java Name:	Java HotSpot(TM) 64-Bit Server VM (17.0.8+9-LTS-211, mixed mode, sharing)
Java Vendor:	Oracle Corporation
JVM Uptime:	31 min 46 sec

Enabled Modules [show]

System Properties [show]

Classes by Number of Instances [view all]

byte[]	31,415	(21.1%)
java.lang.**String**	30,442	(20.5%)
java.util.**HashMap$Node**	10,466	(7%)
sun.font. **TrueTypeFont$DirectoryEntry**	6,981	(4.7%)
java.util.concurrent. **ConcurrentHashMap$Node**	5,749	(3.9%)

Classes by Size of Instances [view all]

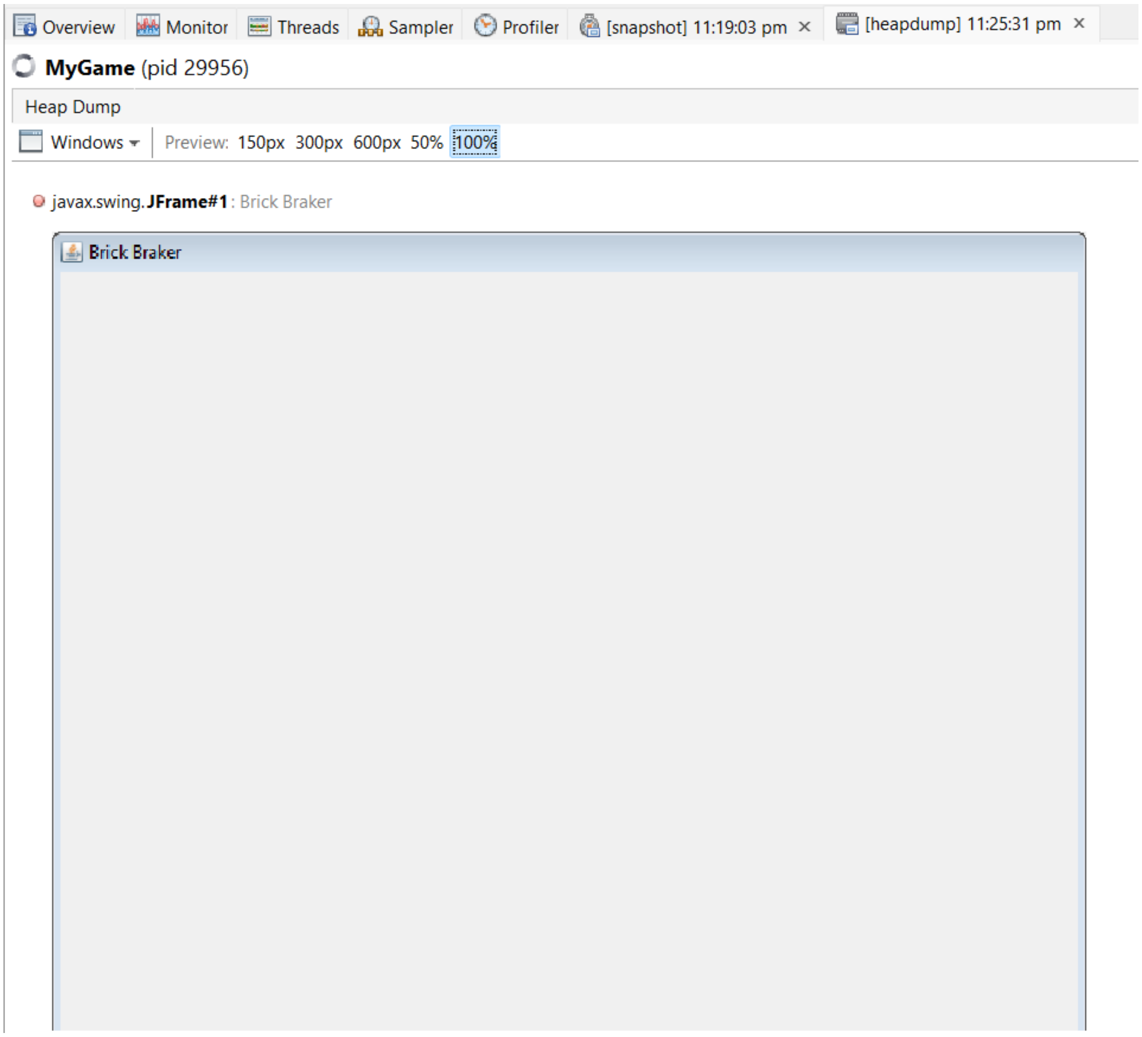
byte[]	2,765,648 B	(40.1%)
java.lang.**String**	730,608 B	(10.6%)
java.util.**HashMap$Node**	334,912 B	(4.9%)
java.lang.**Object[]**	264,176 B	(3.8%)
java.lang.**Class$ReflectionData**	189,504 B	(2.7%)

Instances by Size [view all]

byte[]#3141! : 1,200,000 items	1,200,016 B	(17.4%)
short[]#61 : 9,680 items	19,376 B	(0.3%)
char[]#255 : ...	19,376 B	(0.3%)
java.util.**HashMap$Node[]#31:** : 4,096 items	16,400 B	(0.2%)
char[]#221 : *** Profiler engine warning: Cannot use ClassLoader.findLoadedClass() and/	16,400 B	(0.2%)

Dominators by Retained Size [view all]

class org.graalvm.visualvm.lib.jfluid.server. **ProfilerRuntime [GC root - JNI global, stick**	1,200,424 B	(17.4%)
sun.awt. **Win32FontManager#1**	514,840 B	(7.5%)
class java.time.zone.**ZoneRulesProvider [GC root - sticky class** : ZoneRulesProvider	204,048 B	(3%)
class sun.util.calendar. **ZoneInfoFile [GC root - sticky class** : ZoneInfoFile	149,680 B	(2.2%)
class org.graalvm.visualvm.lib.jfluid.server. **ProfilerInterface [GC root - JNI global, sticl**	137,824 B	(2%)


Overview
Monitor
Threads
Sampler
Profiler
[snapshot] 11:19:03 pm
[heapdump] 11:25:31 pm
MyGame (pid 29956)
Heap Dump
Windows
Preview: 150px 300px 600px 50% 100%
javax.swing.JFrame#1 : Brick Braker
Brick Braker