The HSA2 Graphics Console

Introduction

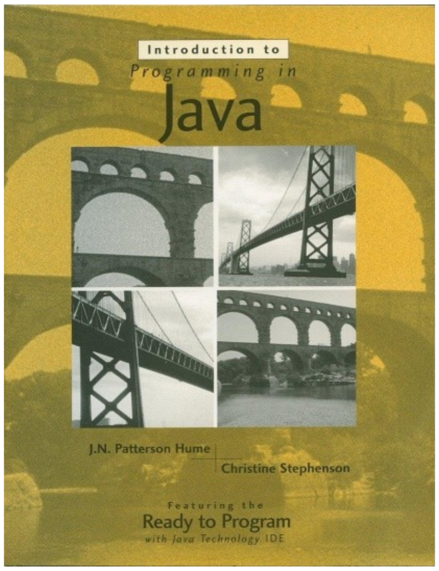
This java package allows users to program graphics in Java without having to learn all of the setup needed to use Swing graphics.

- The syntax is designed to be as close as possible to the standard Java and Swing syntax.
- You can do all sorts of drawing and graphics and animations and load images
- However, other Swing components cannot be used (JLabels, JButtons) and the event handlers are hidden so that event listeners cannot call any user methods.
  Instead you have to poll the listeners to check for keypresses or mouse events. See the example later on.

History (you can skip this)

This software package was originally written by Holt Software Associates (HSA) to teach Java to students. It was called “Ready to Program” and came with a textbook. The original software was written by Tom West.

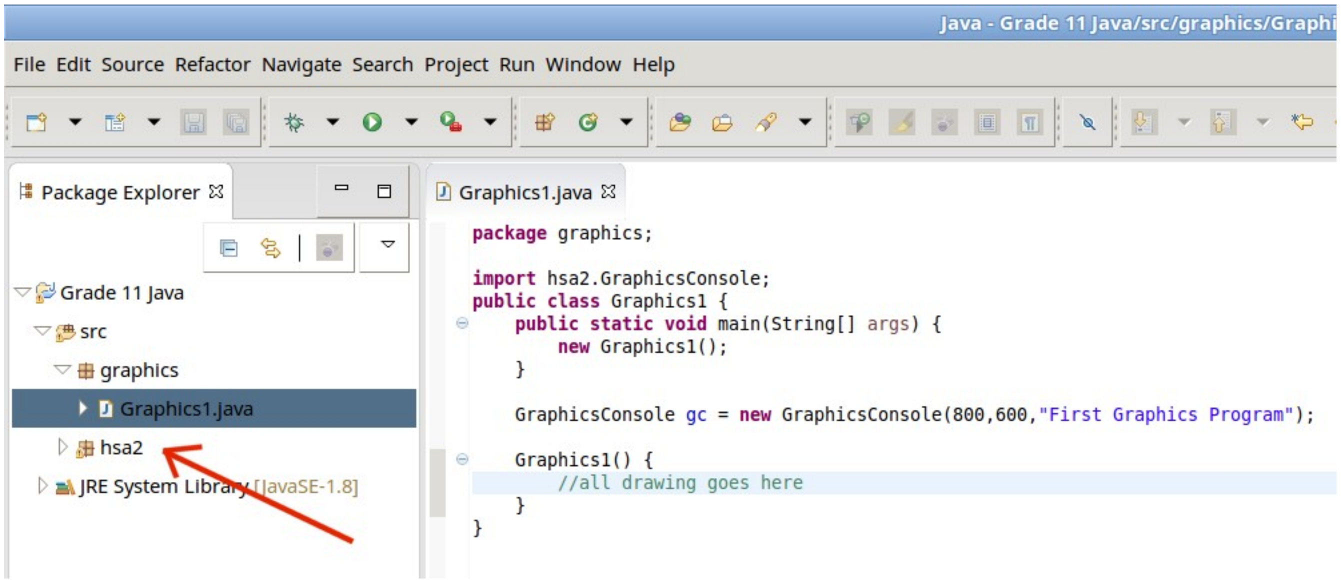
However … the software was not upgradeable and it was stuck on Java 2. The company went bankrupt and the sourcecode was not available.



At some point it was totally rewritten, reverse engineered, by Sam Scott at Sheridan College who based the graphics on Swing JFrame and JPanels. He called the software HSA_UFA (as it was written for the UFA school). After this Josh Gray from TVDSB added mouse routines. Then Michael Harwood from TVDSB added some other features, renamed it to HSA2, and made a GitHub repository for the project at https://github.com/salamander2/HSA2

How to use:

You need to download the package from the a GitHub repository at
https://github.com/salamander2/HSA2 *(click on Clone/Download and save as zip)*
Inside this there is a subfolder called “hsa2” which needs to be copied to your source code folder for your IDE. This is how to do it for Eclipse (Mars):


Java - Grade 11 Java/src/graphics/Graphi
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Grade 11 Java
src
graphics
Graphics1.java
hsa2
JRE System Library [JavaSE-1.8]
Graphics1.java

```
package graphics;

import hsa2.GraphicsConsole;
public class Graphics1 {
    public static void main(String[] args) {
        new Graphics1();
    }

    GraphicsConsole gc = new GraphicsConsole(800,600,"First Graphics Program");

    Graphics1() {
        //all drawing goes here
    }
}
```

1. First you have to make a Java Project (e.g. “Grade 11 Java”)
2. Then make a package (e.g. “graphics”). You’ll now see the “src” file appear.
3. Using Windows Explorer, drag “hsa2” into the “src” folder and choose the “copy files” option.
4. It should now look like this (above) and you can start to use the HSA2 software.

Creating the Console Window

- 🟥 The main part of the program is called a “Graphics Console” because allows normal text console I/O as well as graphics.

The GraphicsConsole window is the window that pops up when you run a program that uses the hsa2 console. The console is a Java **object**.

★ Any time that you use the word **new** you create an object. The object is created by something in the class file called a constructor. There are different constructors which can take different parameters, depending on what sort of GraphicsConsole you want to create.

When you are creating a console, you can specify its size, the size of the font it will use, and the title that should appear in the window. The most common ways of doing this are at the top of the next page:

GraphicsConsole gc = new GraphicsConsole (*width*, *height*);

Creates a console named “gc” of the size specified for *width* and *height* (values are in pixels). *You could name it something other than gc if you wish.*

GraphicsConsole gc = new GraphicsConsole (*width*, *height*, "*title*")

As above, but also specifies a title for the window.

& *You can also do any of the following:* &

GraphicsConsole gc = new GraphicsConsole ();

Creates a console window named “gc” with the default size (650 pixels wide, 500 pixels high) using the default font size (set in File→Preferences→Run Window), and no title.

GraphicsConsole gc = new GraphicsConsole ("*title*")

Creates a console window named “gc” with default size, but with your title for the window.

GraphicsConsole gc = new GraphicsConsole (*width*, *height*);

Creates a console named “gc” of whatever size you like, depending on what you put for *width* and *height* (values are in pixels).

GraphicsConsole gc = new GraphicsConsole (*width*, *height*, *fontSize*)

As above, but also specifies the size of the font to be used in the output window.

GraphicsConsole gc = new GraphicsConsole (*width*, *height*, *fontSize*, "*title*")

As above, but also specifies a title for the window.

Note: You can also open two consoles at once. Can you figure out how?

You can change these properties later on too:

- To set the title to a new title do this: **gc.setTitle("new title")**
- Changing the font size can be done with **gc.setFont(…)**
  See the pdf on fonts for how to do this.
- Changing the screen size after the graphics console is created is more tricky. By far, the easiest way is to set the desired size when you make the GraphicsConsole. You might be able to do it with gc.setPreferredSize()
  gc.setResizable(true) allows the window to be resized with the mouse.

How to make the graphics console fill up the whole screen:

```
/***** Global (instance) Variables ******/
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
int SCRWIDTH = (int) screenSize.getWidth();
int SCRHEIGHT = (int) screenSize.getHeight();
GraphicsConsole gc = new GraphicsConsole (SCRWIDTH, SCRHEIGHT, "Title
of screen");
```

Warnings for common mistakes

1. After the line that says what your package is in your program, you'll have to include the following: import hsa2.GraphicsConsole;
   See ProgramTemplate.java in the pdf folder.

2. Make sure that you use straight quotes. " " not " ". This is often a problem when you cut and paste from internet documents.

3. The spelling of "colour" in programming is the American spelling.
   So you can create a variable: String textColour = "red"; since variables can be any spelling,
   but if you use any built in objects or functions, you have to write
   Color myBackground = Color.RED;
   gc.setColor(new Color(100,200,255));

4. If you ever use println(), the word is lowercase for PRINTLN It's an L (as in **print li**ne). It is not a 1 nor an i .

5. **setBackgroundColor (*Color*)**

   a. There are two setBackground methods.
   The Frame class has **setBackground(*color*)** This <u>will not work</u> with HSA2 graphics.
   You have to use **setBackgroundColor (*color*)**
   See the pdf on how to use colours in order to specify desired colour.

   b. The background colour is only visible after you clear the screen again, because clearing the screen actually redraws everything on it, including the background.
   So, always type: **gc.setBackgroundColor (*Color*);** then **gc.clear();**