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.

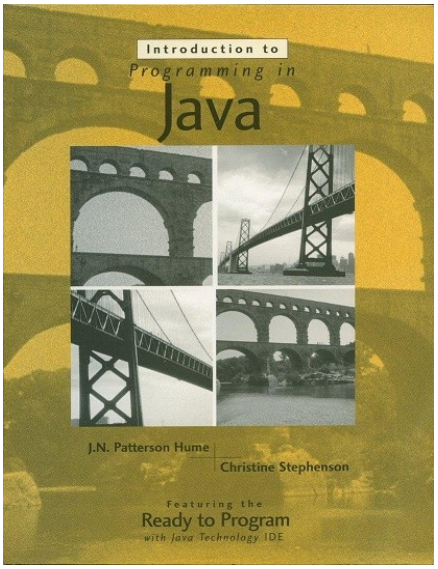
- The syntax is designed to be as close as possible to the standard Java and Swing syntax.
- You can do all sorts of drawing and graphics and animations and load images
- However, other Swing components cannot be used (JLabels, JButtons) and the event handlers are hidden so that event listeners cannot 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 more drawing commands and other features, renamed it to HSA2, and made a GitHub repository for the project at https://github.com/salamander2/HSA2 .



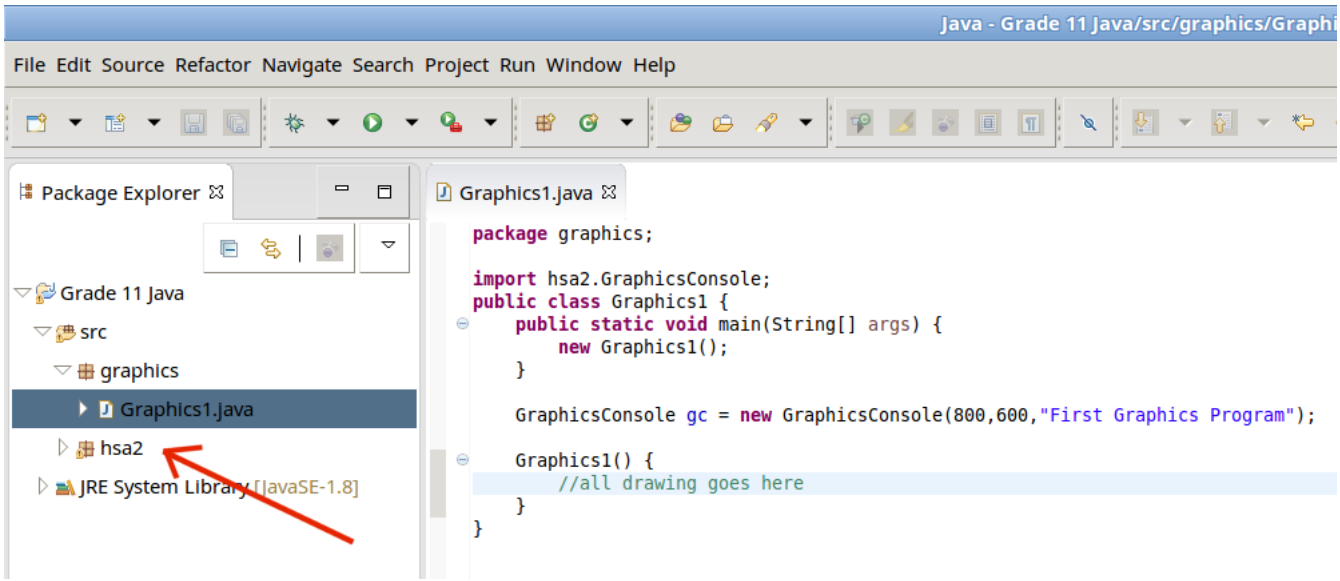
How to use:

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)

Unzip this to somewhere on your desktop (you won't be able to do step 3 below if you don't unzip it).
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 IDE:

1. First you have to make a Java Project (e.g. "Grade 11 Java" in the screen shot below)
2. Then make a package (e.g. "graphics" in screen shot). 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 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**. In all of the examples and code that follows, the GraphicsConsole object is called **gc** and thus all of its methods are accessed by typing **gc.methodName();**
There are some situations where it wouldn't be called "gc", namely when we're creating a second window.

★ Any time that you see the Java keyword **new**, it means that an object is being created. 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.

GraphicsConsole Constructors

This is the most common way of creating a GraphicsConsole. You probably will never need anything else:

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

This creates a window named "gc" of the specified *width* and *height* (values are in pixels).
The default title is "HSA2 Graphics Console", which we change later on in the program.

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

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

GraphicsConsole gc = new GraphicsConsole ();

Creates a console window named "gc" with the default size (650 pixels wide, 500 pixels high)

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

Creates a console window named "gc" with default size, but with a custom title for the window.

The following two constructors are seldom used and remain for legacy programs, for backwards compatibility:

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 the size of the font to be used in the output window.

You can change these properties later on too:

- To set the title to a new title do this: **gc.setTitle("new title");**
  This can be very useful for debugging.

- Changing the screen size after the graphics console is created is more tricky.
  It's always best to set the desired size when you create 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 have to include the following **import**:
   import hsa2.GraphicsConsole;
   See ProgramTemplate.java in the pdf folder.
   (You'll also have to add imports for Font and Color if you use them.)

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. **gc.setBackgroundColor (*Color*)**

   a. There are two setBackground methods.
   The Frame class has **setBackground(*color*)** This will not work 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();**