

DANH MỤC TỪ VIẾT TẮT

Danh mục từ viết tắt tiếng Anh

Từ	Ý nghĩa
WIMP	window, icon, menu, pointing device – cửa sổ, biểu tượng, thực đơn và các thiết bị con trỏ - đây là các thành phần cơ bản của giao diện đồ họa
GUI	Graphics User Interface – Giao diện đồ họa
AWT	Abstract Windows Toolkit – Bộ thư viện lập trình đồ họa
IDE	Integrated Development Environment – môi trường phát triển được tích hợp sẵn, là các công cụ được tích hợp sẵn môi trường để lập trình, biên dịch và chạy chương trình
JFC	Java Foundation Class – Lớp ứng dụng riêng của Java
JEE	Java Enterprise Edition – Phiên bản của Java dành cho xây dựng các ứng dụng doanh nghiệp
J2ME	Java 2 Micro Edition – Phiên bản của Java dành cho xây dựng các ứng dụng cầm tay và di động.
CPU	Central Processing Unit – Bộ xử lý trung tâm của máy tính
API	Giao diện lập trình ứng dụng (Application Programming Interface)

DANH MỤC CÁC HÌNH VẼ

Hình 1.1. Giao diện đồ họa người dùng (GUI)	7
Hình 1.2. Mô hình WIMP	10
Hình 1.3. Các thành tố điều khiển thường gặp.....	15
Hình 1.4. Mô hình sự kiện.....	17
Hình 1.4. So sánh AWT và SWING	22
Hình 1.5. AWT, SWING và JDK	23
Hình 2.1. Sơ đồ phân cấp các lớp trong gói AWT.....	25
Hình 2.2. Cây thừa kế mô tả lớp Container và các lớp con của lớp đó	25
Hình 2.3. Kết quả thu được với bố cục FlowLayout.....	46
Hình 2.4. Kết quả thu được với bố cục BorderLayout.....	49
Hình 2.5: Kết quả minh họa bố cục GridLayout.....	51
Hình 2.6. Những lớp sự kiện của gói java.awt.events.	53
Hình 2.7. Các giao diện lắng nghe sự kiện của gói <i>java.util.event</i>	54
Hình 2.8. Các sự kiện	63
Hình 2.9. Các giao diện thực thi sự kiện.....	64
Hình 2.10. Các phương thức trong vòng đời của Applet.....	68
Hình 2.11. Vòng đời của applet.	71
Hình 2.12. Applet bị giới hạn tại các phiên bản mới của một số trình duyệt	78
Hình 2.13. Giao diện chương trình máy tính bỏ túi	79
Hình 2.14. Đồng hồ điện tử theo giờ hệ thống.....	88
Hình 3.1. Các lớp cha phổ biến của các thành phần SWING.	92

Hình 3.2. Các thành phần cơ bản của gói SWING.	93
Hình 3.3. Sơ đồ thể hiện sự kế thừa của các button trong JComponent.	103
Hình 3.4. Giao diện Text Sample Demo	116
Hình 3.5. Các giao diện chính của chương trình.....	127
Hình 3.6. Nhân vật Pacman.....	136
Hình 3.7. Pacman di chuyển và ăn các điểm vàng.....	142
Hình 3.8. Giao diện chương trình quản lý sách	144
Hình 3.9. Tìm kiếm theo mã sách	145
Hình 3.10. Giao diện chương trình quản lý danh sách sinh viên.....	163
Hình 3.11. Bố cục trong Project.....	164
Hình 3.12. Thêm các thư viện cần thiết	166
Hình 3.13. Màn hình đăng nhập.....	166

Chương 1. GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN VÀ NGÔN NGỮ JAVA

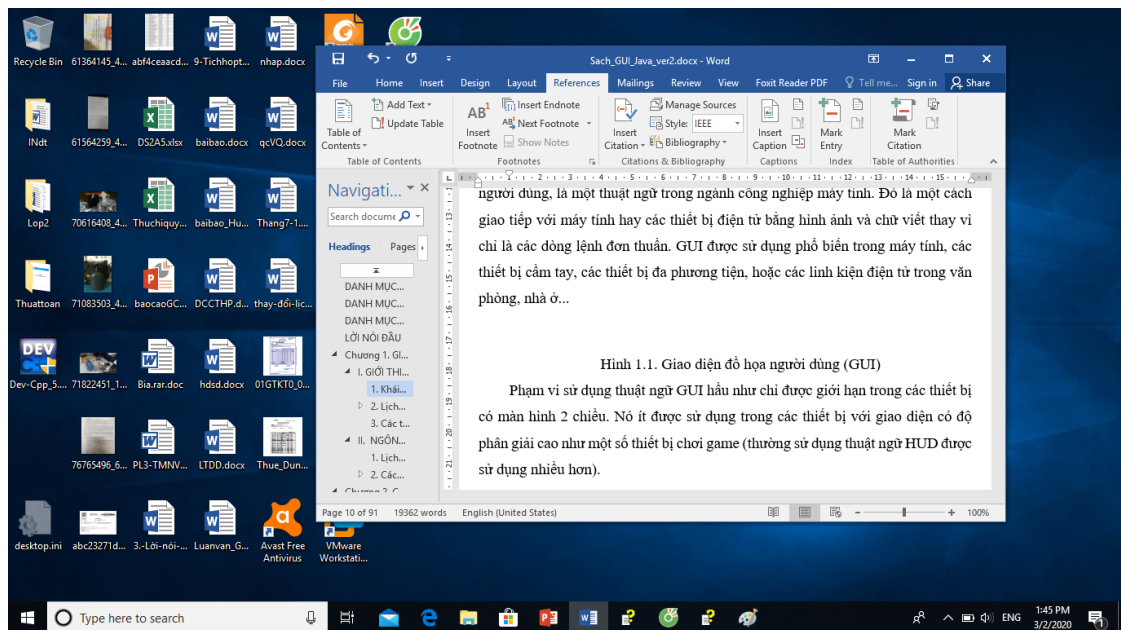
I. GIAO DIỆN ĐỒ HỌA

1. Khái niệm

Phát triển hệ điều hành là một động lực thúc đẩy sự ra đời của Giao diện đồ họa GUI. Hệ điều hành cổ xưa như MS-DOS chỉ thực hiện với giao diện dòng lệnh mang lại nhiều bất tiện như không thân thiện, người sử dụng khó nhớ cú pháp của tập lệnh, do đó không thể tiếp cận đến nhiều người dùng. Sự ra đời và phát triển của Giao diện đồ họa đã góp phần làm cho công nghiệp phần mềm phát triển bùng nổ và xuất hiện trên mọi lĩnh vực như hiện nay. Chương mở đầu này sẽ giới thiệu về giao diện đồ họa, lập trình giao diện đồ họa và cụ thể trong ngôn ngữ lập trình Java. Chương này đã tham khảo các nguồn tài liệu [1], [3] và [6].

Trong từ điển Cambridge, GUI được định nghĩa là *"a way of arranging information on a computer screen that is easy to understand and use because it uses icons (= pictures), menus, and a mouse rather than only text"* tạm dịch là *"một cách sắp xếp thông tin trên màn hình máy tính một cách dễ hiểu bằng cách sử dụng các biểu tượng (= các ảnh), thực đơn và sử dụng con chuột thay vì chỉ dùng chữ"*. Trong đó, GUI là viết tắt của Graphical User Interface, có nghĩa là Giao diện đồ họa người dùng, và đây đã trở thành một thuật ngữ trong ngành công nghiệp máy tính. Đó là một cách giao tiếp với máy tính hay các thiết bị điện tử bằng hình ảnh và chữ viết thay vì chỉ là các dòng lệnh đơn thuần. GUI được sử dụng phổ biến trong máy tính, các thiết bị cầm tay, các thiết bị đa phương tiện, hoặc các linh kiện điện tử trong văn phòng, nhà ở...

Ví dụ Hình 1.1 là ảnh ứng dụng của GUI trên nền tảng hệ điều hành Windows với cửa sổ - window, các thành phần trong một cửa sổ như menu, các nút nhấn, các nút cuộn, hình ảnh và chữ viết, các biểu tượng – icon... và người sử dụng dùng bàn phím hoặc con chuột để làm việc với ứng dụng đồ họa. Khái niệm giao diện đồ họa GUI có thể thấy trên nhiều nền tảng như Windows, Mac và iOS và Android...



Hình 1.1. Giao diện đồ họa người dùng (GUI)

Phạm vi sử dụng thuật ngữ GUI hầu như chỉ được giới hạn trong các thiết bị có màn hình 2 chiều. Nó ít được sử dụng trong các thiết bị với giao diện có độ phân giải cao như một số thiết bị chơi game (thường thuật ngữ HUD được sử dụng nhiều hơn).

GUI đầu tiên được các nhà nghiên cứu tại Xerox PARC phát triển trong thập niên 1970. Ngày nay hầu hết các hệ điều hành máy tính nhiều người dùng đều sử dụng giao diện này.

2. Lịch sử của GUI

Douglas Englebar (1925-2013) hiện được xem là cha đẻ của giao diện người dùng đồ họa. Sau khi tốt nghiệp ngành kỹ thuật điện vào năm 1948, Douglas làm việc tại Viện NACA (NACA Institute, tiền thân của NASA hiện giờ). Sau đó, với mong muốn được làm công việc có thể mang lại lợi ích cho toàn nhân loại, ông đã nghĩ đến việc xây dựng một chiếc máy có thể làm tăng trí tuệ con người.

Trong giai đoạn chiến tranh, Douglas làm việc với vai trò là một kiểm soát viên radar để có thể hình dung ra một thiết bị hiển thị được xây dựng xoay quanh các ống tia cathode mà qua đó người dùng có thể xây dựng các mô hình thông tin

một cách trực quan thông qua đồ họa, và họ có thể tương tác với bất cứ những gì họ nhìn thấy. Để có thể thực hiện được ý tưởng này, ông phải tìm nguồn tài trợ. Song, đó không phải là việc dễ dàng mà có thể thực hiện được trong thời gian ngắn.

Năm 1955, Douglas đạt được học vị tiến sĩ và vào làm việc tại Viện Nghiên cứu Stanford (Stanford Research Institute), và tại đó ông đã đạt được nhiều bằng sáng chế về "các thành phần tiểu hóa máy tính" (miniaturizing computer components). 1959, ông đã thuyết phục được Lực Lượng Hàng Không Hoa Kỳ (United States Air Force) hỗ trợ quỹ cho ý tưởng của ông. 1962, ông cho xuất bản luận văn Augmenting Human Intellect với nội dung chính là các ý tưởng mà ông đã ấp ủ. Qua luận văn này, ông khẳng định "máy tính không phải để thay thế trí tuệ con người, mà là để cải thiện nó". Một trong những ví dụ mà ông sử dụng trong luận văn này là một kiến trúc sư sử dụng một công cụ nào đó tương tự các phần mềm CAD ngày nay để thiết kế nên các tòa nhà.

Douglas và các cộng sự của ông tiếp tục làm việc để phát triển các ý tưởng và công nghệ này. Và vào năm 1968, một sự trình diễn công khai được thực hiện trước hơn 1000 các chuyên gia máy tính về những thành quả từ những ý tưởng ban đầu của ông.

Tiền thân của GUI được khai sinh bởi những nhà nghiên cứu tại Stanford Research Institute - được dẫn đầu bởi Douglas Engelbart. Khi đó, họ phát triển việc sử dụng những siêu liên kết (hyperlinks) dựa trên chữ cho hệ thống On-Line (One-Line System), trong đó các liên kết được thao tác với một con chuột. Khái niệm siêu liên kết sau đó đã được các nhà nghiên cứu tại Xerox PARC mở rộng sang phạm vi đồ họa.

Đến năm 1963, Sketchpad, một hệ thống dựa trên con trỏ (pointer) được phát triển bởi Ivan Sutherland. Hệ thống này sử dụng một cây bút phát sáng để thao tác việc tạo và thao tác với các đối tượng trong các bản vẽ kỹ thuật.

- Giao diện người dùng PARC

PARC là một công ty nghiên cứu và phát triển tại Palo Alto, California, nổi tiếng với những đóng góp cho công nghệ thông tin và hệ thống phần cứng. Giao diện người dùng PARC gồm các thành phần đồ họa như cửa sổ (window), thực đơn (menu), nút kiểm tròn (radio button), ô kiểm vuông (check box) và các biểu tượng (icon).

Dựa trên hệ thống tiền thân, lần lượt các hệ thống đồ họa được ra đời. Năm 1981, mô hình điều hành máy tính dựa trên GUI đầu tiên được ra đời là Xerox 8010 Star Information System. Tiếp theo đó là Apple Lisa năm 1983, Macintosh 128K của Apple năm 1984, Atari ST và Commodore Amiga năm 1985. Việc điều khiển đồ họa bằng lệnh (command) được khai sinh khi IBM cho ra đời Common User Access (1987), trong đó các chuỗi lệnh khác nhau sẽ được áp dụng cho các chương trình khác nhau như: phím F3 sẽ kích hoạt chế độ hỗ trợ trong WordPerfect, nhưng nó sẽ đóng chương trình trong các ứng dụng khác của IBM.

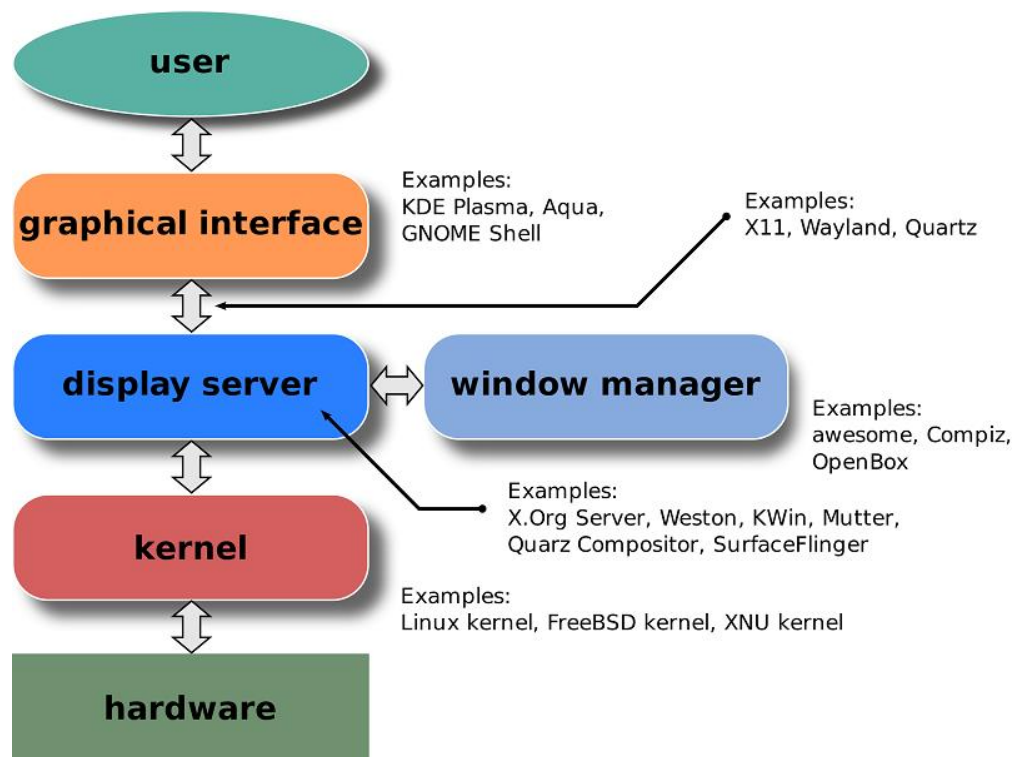
Tiếp nối sự phát triển của các hệ thống GUI là sự ra đời của các keyboard overlays (tạm dịch, bàn phím cho phép điều chỉnh bố cục chức năng). Đó là những mảnh giấy hay plastic được đặt trong những chỗ trống giữa các phím, nó có nhiệm vụ cung cấp cho người dùng chức năng của các phím khác nhau của các ứng dụng đã được xác định. Có thể hình dung những bàn phím như vậy với những keystroke ngày nay như: Control-Alt-Delete sẽ mở Task Manager trong hệ điều hành Windows, còn trong các hệ thống Unix thì sẽ tắt máy tính.

Các hệ thống GUI phổ biến ngày nay là Microsoft Windows, Mac OS X, X Window System trên các PC (Personal Computer), laptop. Ngoài ra còn có sự góp mặt của các thiết bị di động như Symbian, BlackBerry OS, Android, iOS. Các hệ thống này đều được phát triển dựa trên những ý tưởng ban đầu của Xerox, cho nên chúng gần như có các khái niệm tương tự nhau (như button, radio button, menu...)

3. Các thành phần trong GUI

Một hệ thống GUI là sự kết hợp của các công nghệ, thiết bị để cung cấp cho người dùng một nền tảng cho phép người sử dụng có thể tương tác với nó.

Một chuỗi các thành phần của GUI tuân theo một ngôn ngữ trực quan (visual language) để biểu diễn thông tin được lưu trữ trong các máy tính. Thông dụng nhất khi kể đến sự kết hợp các thành phần như vậy là mô hình WIMP (window, icon, menu, pointing device lần lượt là cửa sổ, biểu tượng, thực đơn và thiết bị trỏ) trong các máy tính cá nhân.



Hình 1.2. Mô hình WIMP

Với mô hình trong Hình 1.2, giao diện đồ họa của phần mềm xuất hiện trước mắt người dùng. Người dùng sẽ làm việc với giao diện này. Giao diện đồ họa làm nhiệm vụ trình bày các đối tượng đồ họa, tiếp nhận các thao tác đầu vào của người dùng (gõ chữ, click chuột). Phần display server nằm dưới có nhiệm vụ đưa kết quả hiện lên màn hình, tiếp nhận yêu cầu của người dùng đẩy cho phần kernel nằm dưới xử lý.

- Thành tố điều khiển hay còn gọi là khiên tố - các thuật ngữ dùng cho nó là graphical widget hoặc widget hoặc control

Thành tố điều khiển (còn gọi là ô điều khiển, khiên tố, thành phần điều khiển,...) là một thành phần của GUI mà thông qua chúng, một người sử dụng có thể tương tác với máy tính, thực hiện các thao tác bằng chuột, bàn phím hay thiết bị ngoại vi khác. Ví dụ về ô điều khiển gồm các hộp văn bản hay các cửa sổ.

Có nhiều loại thành tố điều khiển chẳng hạn như:

1. Chỉ nhận lệnh và thi hành như các nút điều khiển
2. Nhận thông tin như các hộp văn bản
3. Chỉ thi hành thao tác khi được lệnh của hệ thống như đồng hồ hẹn giờ (timer)
4. Tương tác và thông báo như các hộp thông báo (message box), hộp xác nhận (confirm box), và các gợi ý (tool tip)


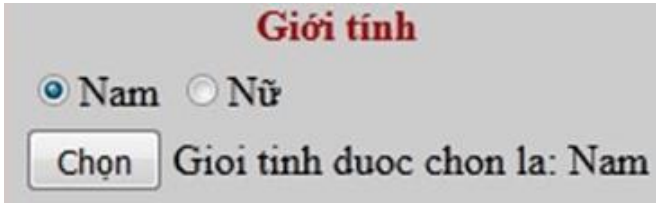
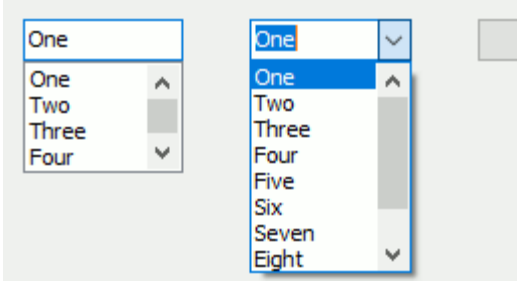
Các khiên tố thực ra phải hiểu với đặc tính ảo nghĩa là chúng phân biệt với các bộ phận vật lý thông thường khác. Thí dụ: các nút điều khiển ảo chỉ có thể được nhấn bởi chuột, bàn phím hay bằng ngón tay trên màn hình cảm biến (touch screen). Hình ảnh các thành phần điều khiển chỉ để so sánh như các nút bấm ở thế giới thực bên ngoài




Một số khiên tố có thể không thấy được trên giao diện đồ họa nhưng vẫn có tương tác với người dùng qua một hay vài chức năng xác định nào đó. Thí dụ với các phím nóng (hot key control) người ta không thể thấy hình dạng của các khiên tố này nhưng nó được cài đặt để giúp người dùng máy sử dụng bàn phím (thay vì phải nhấp các nút của chuột). Một thí dụ khác về ô điều khiển không thấy được là đồng hồ hẹn giờ (timer) rất thông dụng trong ngôn ngữ VB. Các đồng hồ hẹn giờ này có thể được cài đặt trong chương trình để tự động làm các thao tác theo đúng một khoảng thời gian mà người lập trình muốn.

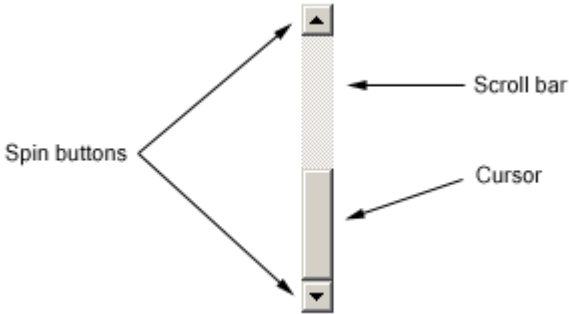



Tùy theo hệ điều hành và chuẩn GUI mà các ô điều khiển và đặc điểm chức năng của chúng có thể khác nhau. Ngoài ra, các ngôn ngữ lập trình hỗ trợ cho việc

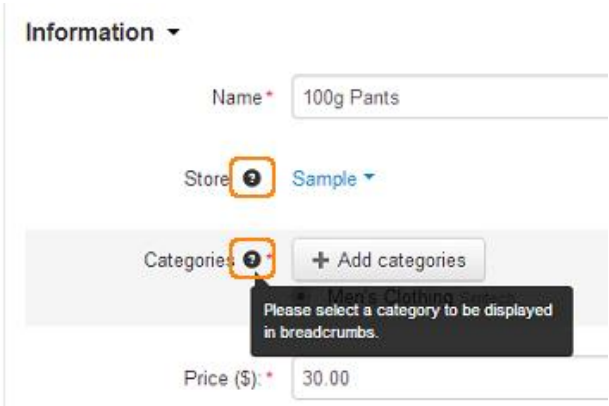
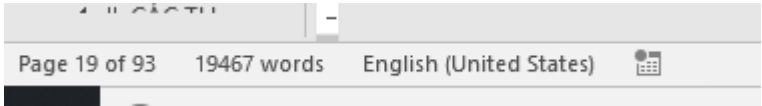
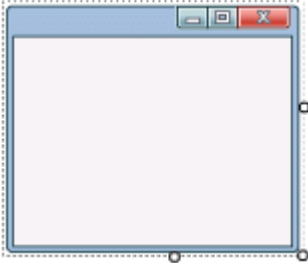
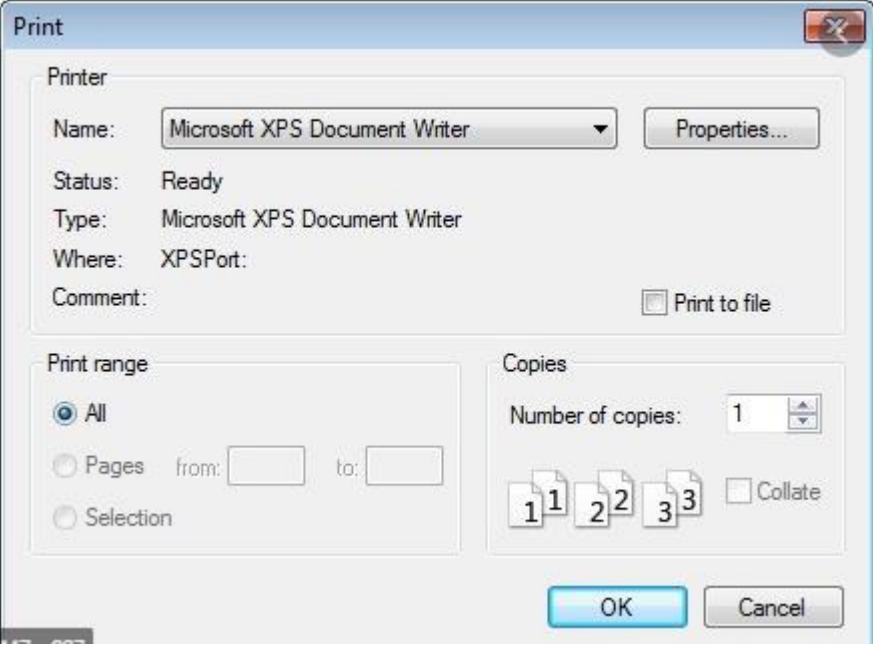
tạo ra các chương trình có giao diện đồ họa cũng có thể cung cấp thêm các ô điều khiển riêng biệt. Trong hệ thống X Window thì các ô điều khiển thường được hỗ trợ chung trong một tập hợp qua các bộ công cụ lập trình. Trong khi đó, các ngôn ngữ lập trình cho Windows thường cung cấp thêm các ô điều khiển bên cạnh những ô điều khiển sẵn có của hệ điều hành.

Có nhiều loại khiển tố, tùy theo GUI và tùy theo các phần mềm thiết kế sẵn cho các chúng. Danh sách sau đây chỉ liệt kê một số thường dùng.

Tên thành tố điều khiển	Minh họa				
<i>Lựa chọn và hiển thị</i>					
Nút nhấn – button					
Ô đánh dấu (check box)	<table border="1"> <tr> <td>Đúng</td><td>Sai</td></tr> <tr> <td><input checked="" type="checkbox"/></td><td><input type="checkbox"/></td></tr> </table>	Đúng	Sai	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Đúng	Sai				
<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Nút chọn (radio button)					
Hộp đa hợp (combo box)					

Biểu tượng icon	
Thanh công cụ toolbar	
Trình đơn menu	
Đệ quy	
Hiển thị dạng cây	<div> <input type="checkbox"/> > Flavors </div> <div> <input checked="" type="checkbox"/> ∨ Toppings </div> <div> <input type="checkbox"/> > Candy </div> <div> <input checked="" type="checkbox"/> ∨ Fruits </div> <div> <input type="checkbox"/> Mango </div> <div> <input checked="" type="checkbox"/> Peach </div> <div> <input type="checkbox"/> Kiwi </div> <div> <input checked="" type="checkbox"/> ∨ Berries </div> <div> <input checked="" type="checkbox"/> Strawberry </div> <div> <input checked="" type="checkbox"/> Blueberry </div> <div> <input checked="" type="checkbox"/> Blackberry </div>

<i>Thanh di chuyển/tiến triển</i>	
Thanh cuộn (Scrollbar)	 <p>The diagram shows a vertical scrollbar. On the left, two arrows point to the up and down arrow buttons, labeled 'Spin buttons'. On the right, a horizontal line represents the 'Scroll bar', and a small rectangular slider is labeled 'Cursor'.</p>
Thanh tiến trình (Progress bar)	 <p>The screenshot shows a dialog box titled 'Title of progress dialog.' with the text 'Loading.....' above a progress bar. The bar is filled with a teal color up to 82%, with '82%' displayed on the left and '82/100' on the right.</p>
Thanh điều chỉnh (trackbar)	 <p>The screenshot shows a horizontal trackbar with a blue track and a white slider knob in the center. There are small tick marks along the track.</p>
<i>Văn bản</i>	
Nhập: Hộp văn bản (text box)	 <p>The screenshot shows a light blue form with two text input fields. The first field is labeled 'Enter First Name' and the second is labeled 'Enter Last Name'.</p>

<p>Xuất: Gợi ý sử dụng(tooltip)</p>	
<p>Xuất: Thanh trạng thái (status bar)</p>	
<p>Cửa sổ Window</p>	
<p>Hộp thoại (dialog box)</p>	

Hình 1.3. Các thành tố điều khiển thường gặp

Ngoài ra còn có một số thành tố điều khiển khác như: thẻ tab, đồng hồ hẹn giờ timer, danh mục hình image list, lên xuống up/down và các phím nóng – hot key như Ctrl+S, Ctrl+N v.v.

- Sự kiện và xử lý sự kiện (Event và Event handle)

Khi lập trình GUI, khái niệm sự kiện Event là một khái niệm thường gặp nhưng những người mới lập trình thường cảm thấy khó hiểu, khó lập trình với nó. Giả sử khi chúng ta click vào một nút nhấn Button, trong lập trình sẽ có 1 phương thức để xử lý và truyền các tham số cần thiết để cho ra một kết quả nào đó, chẳng hạn như nhấn vào nút Next thì một cửa sổ tiếp theo hiện ra, bấm vào nút Close trên cửa sổ thì cửa sổ bị mất v.v.. Điều này liên quan đến sự kiện - Event và xử lý sự kiện – Event handle.

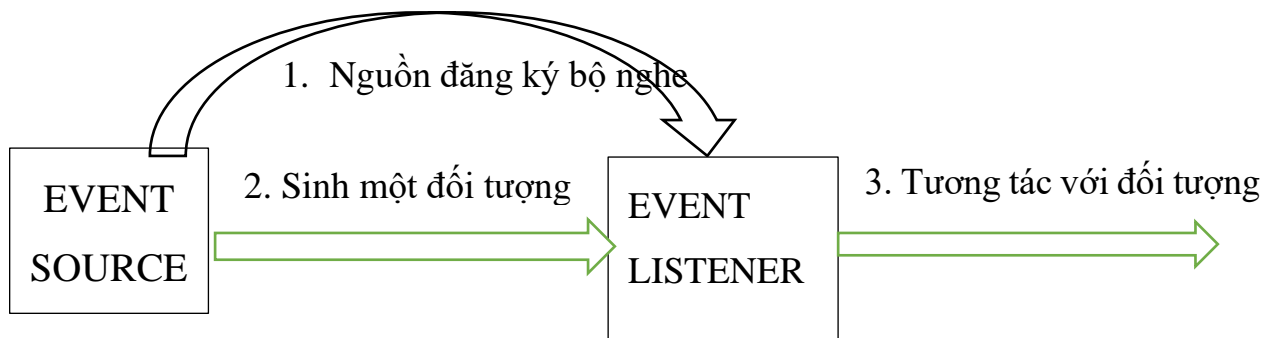
Event được hiểu là một khi hành động được xảy ra thì sẽ phát sinh, hay thông báo cho người dùng sẽ phản hồi với hành động như thế nào. Ví dụ khi click vào nút nhấn thì người lập trình sẽ xử lý những đoạn mã của trong phương thức onClick của interface OnClickListener. Mô hình sự kiện được mô tả thành các phần như sau:

+ Nguồn sự kiện – Event source: Đó là các thành tố điều khiển như: Cửa sổ, con chuột, bàn phím, nút nhấn v.v.

+ Bộ lắng nghe và xử lý – Event Listener/ Handler: Đây là các bộ nhận những sự kiện và xử lý tương tác với người dùng. Như: hiển thị thông tin cho người dùng, thực hiện tác vụ tính toán v.v.. Ví dụ các Bộ nghe ActionListener, MouseListener, WindowListener lần lượt xử lý các sự kiện hoạt động nào đó, sự kiện chuột, sự kiện liên quan đến cửa sổ..

+ Đối tượng sự kiện – Event Object: Đối tượng được tạo ra khi sự kiện xảy ra, chứa tất cả các thông tin về sự kiện mà nó xảy ra như loại sự kiện, nguồn của sự kiện.

Mô hình sự kiện hoạt động như sau: Một bộ Lắng nghe Listener được đăng ký với một Nguồn Source. Khi bộ Lắng nghe được đăng ký, nó sẽ chờ tới khi sự kiện xảy ra. Khi sự kiện xảy ra, một Đối tượng Object sự kiện được sinh ra, đối tượng sự kiện này được kích hoạt bằng nguồn đã đăng ký bộ Lắng nghe. Mỗi lần bộ Lắng nghe nhận được Đối tượng từ sự kiện Nguồn nó sẽ giải mã những thông điệp và xử lý sự kiện mà nó xuất hiện. (Hình 1.4)



Hình 1.4. Mô hình sự kiện

Đối với lập trình GUI, bên cạnh các thành tố widget thì các sự kiện và xử lý các sự kiện là không thể thiếu vì các ứng dụng có giao diện GUI, việc thao tác click chuột, sử dụng bàn phím là thường xuyên xảy ra.

II. LẬP TRÌNH GIAO DIỆN ĐỒ HỌA VỚI NGÔN NGỮ JAVA

1. Ngôn ngữ lập trình Java

a. Lịch sử, đặc trưng của ngôn ngữ Java

Java được đề xuất bởi James Gosling (sinh năm 1955) và các đồng nghiệp ở Sun Microsystems năm 1991. Từ ý tưởng muốn lập trình để điều khiển mà không phụ thuộc vào loại CPU cho các thiết bị điện tử như tivi, máy giặt, lò nướng... họ đã xây dựng một ngôn ngữ chạy nhanh, gọn, hiệu quả, độc lập thiết bị gọi là ngôn ngữ “Oak”, sau này được đổi tên thành Java vào năm 1995.

- Đặc trưng của ngôn ngữ Java gồm có:

+ Ngôn ngữ lập trình hướng đối tượng; độc lập với mọi nền tảng phần cứng, mọi hệ điều hành; hỗ trợ phát triển ứng dụng chạy trên mạng.

+ Ngôn ngữ lập trình mạnh: tất cả dữ liệu đều phải được khai báo một cách tường minh; việc kiểm tra mã lệnh (code) được thực hiện tại thời điểm biên dịch và thông dịch; hạn chế các lỗi của chương trình.

+ Ngôn ngữ lập trình có tính bảo mật cao: xây dựng môi trường bảo mật cho việc thực thi chương trình; có nhiều mức khác nhau cho việc kiểm soát bảo mật.

+ Ngôn ngữ lập trình đa tuyến: cho phép thực hiện nhiều nhiệm vụ đồng thời trong một ứng dụng.

b. Các ứng dụng của Java

Có 4 kiểu ứng dụng chính của Java:

- Ứng dụng độc lập - Standalone Application: Còn được biết đến là ứng dụng Desktop (Desktop Application) hoặc ứng dụng trên nền tảng hệ điều hành Windows (Window based Application). Để tạo ra ứng dụng kiểu này ta thường sử dụng AWT, SWING hoặc JavaFX framework.

- Ứng dụng web - Web Application: Là ứng dụng chạy trên server và tạo được các trang động. Hiện nay, servlet, jsp, struts, jsf, spring... là những công nghệ được sử dụng để tạo Web Application trong java.

- Ứng dụng doanh nghiệp - Enterprise Application: Là ứng dụng dành cho doanh nghiệp, ví dụ như các ứng dụng cho ngân hàng (Banking Application) với lợi thế là tính bảo mật cao, cân bằng tải (load balancing) và phân cụm (clustering). Trong java, EJB được sử dụng để tạo các Enterprise Application.

- Ứng dụng di động - Mobile Application: Là ứng dụng được tạo ra cho các thiết bị di động. Hiện nay 2 phiên bản Android và J2ME được sử dụng để tạo các ứng dụng này.

Chương 4 của cuốn sách này sẽ tìm hiểu một số ứng dụng cụ thể xây dựng trên ngôn ngữ Java, trong đó chúng tôi tập trung vào 2 loại ứng dụng đầu tiên được đề cập ở trên.

c. Các công cụ lập trình với Java

Là môi trường phát triển tích hợp IDE (Integrated Development Environment) hay là một chương trình để viết các mã (code). Chương trình này hỗ trợ nhiều tính năng tự động hóa cho người phát triển, như gợi ý khi lập trình, tự hoàn thiện mã,... Một số IDE thông dụng hiện nay cho Java gồm:

- NetBean: Được đánh giá là môi trường phát triển tích hợp mã nguồn mở miễn phí mạnh nhất cho Java. Nó được các nhà phát triển phần mềm chuyên nghiệp sử dụng để xây dựng các ứng dụng doanh nghiệp, web, di động và desktop. NetBeans là đa nền tảng IDE và được sử dụng để hỗ trợ trên Linux, Windows, Mac cũng như trên Oracle Solaris. Ngoài ra, NetBeans là một môi trường được phát triển nhằm mục đích hỗ trợ phân tích, thiết kế, mã hoá, lập hồ sơ, thử nghiệm, biên soạn, gỡ lỗi, chạy và triển khai các ứng dụng.

Trang web tải NetBean: <https://netbeans.org/downloads/8.2/rc/>

- IntelliJ IDEA: Được đánh giá là một IDE có đầy đủ tất cả các tính năng cho các nhà phát triển Java EE. IntelliJ IDEA có hai phiên bản miễn phí và bản “Ultimate”. Phiên bản miễn phí tích hợp nhiều tính năng để xây dựng ứng dụng Android cũng như các ứng dụng JVM. Phiên bản Ultimate có các tính năng bổ sung như hỗ trợ cho các ngôn ngữ phát triển web (JavaScript, Coffeescript,...), hỗ trợ các framework (Node.js, Angular và React), hỗ trợ Java EE (JSF, JAX-RS, CDI, JPA,...), kiểm soát phiên bản với Team Foundation Server, Perforce, Clearcase và Visual SourceSafe.

Trang web tải IntelliJ IDEA: <https://www.jetbrains.com/idea/download/>.

- Eclipse IDE: Có hệ sinh thái riêng với một cộng đồng lớn của các nhà phát triển, với nhiều tài liệu hay và nhiều plugin để phát triển Java một cách tốt nhất. Các lập trình viên sử dụng Eclipse để phát triển ứng dụng Java cho di động, máy tính để bàn, web, doanh nghiệp cũng như các ứng dụng hệ thống nhúng.

Trang web tải Eclipse:

<https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/keplersr1>

Ngoài các IDE nêu trên, còn nhiều công cụ hỗ trợ lập trình Java khác như: Java Studio Enterprise, Sun Java Studio Creator, Borland JBuilder, JDeveloper, và JCreator v.v..

Các chương trình được viết trong cuốn sách này được triển khai trên công cụ Netbeans 8.1 và JDK 1.8. Vì vậy để thực hiện theo các chương trình được viết trong sách, độc giả cũng cần tải về và cài đặt các phiên bản Netbeans và JDK tương đương hoặc mới hơn.

2. Lập trình giao diện đồ họa với Java

a. Giới thiệu về JFC (Java Foundation Class)

Để hỗ trợ lập trình GUI, Java đã xây dựng một dự án riêng là JFC – Java Foundation Class. JFC là tập hợp các tính năng để xây dựng giao diện người dùng đồ họa - GUI, để tạo ra các chức năng đồ họa phong phú và tương tác với các ứng dụng Java. JFC là một dự án phối hợp giữa Netscape's Internet Foundation Classes (IFC) và IBM's Taligent division and Lighthouse Design. JFC bao gồm có 5 thư viện:

- SWING GUI Components: gói SWING

- AWT: gói AWT

- Pluggable Look and Feel: Pluggable Look-and-Feel là một trong số những tính năng của SWING. Tính năng này cho phép ứng dụng có thể thay đổi toàn bộ giao diện chỉ với một thao tác lệnh. Giống như chúng ta chọn giao diện của chiếc điện thoại thông minh.

- Accessibility API (Accessibility Application Programming Interface): Thành phần hỗ trợ truy nhập bằng cách làm cho ứng dụng tương tác với các phương tiện khác như bộ đọc chữ nổi Braille, bộ đọc chữ từ màn hình Screen Reader...

- Java 2D API: Hỗ trợ đồ họa 2D và hỗ trợ kéo và thả Drag and Drop giúp tương tác kéo – thả giữa người và máy trên mọi hệ điều hành, qua đó có thể trao đổi dữ liệu giữa chương trình Java và chương trình không phải Java.

b. Hai gói cơ bản AWT và SWING

AWT (Abstract Windows Toolkit) là thư viện API cung cấp các đối tượng GUI. Nó dùng để tạo các liên kết giao diện giữa ứng dụng Java và hệ điều hành. AWT sẽ chiếm nhiều tài nguyên của hệ thống. Trong gói này sẽ gồm nhiều lớp định nghĩa nhiều thành phần đối tượng khác nhau để tạo một giao diện GUI bao gồm có tạo các thành phần đồ họa như nút nhấn, hộp chữ...hay có cả các lớp quản lý việc bố trí các phần tử. AWT hoạt động theo mô hình ứng dụng hướng sự kiện event-oriented application để xử lý các sự kiện chuột và bàn phím của người dùng. Ngoài ra, AWT cũng chứa các công cụ xử lý đồ họa và hình ảnh, các lớp sử dụng tác vụ với clipboard – vùng nhớ đệm như thao tác cut, paste.

AWT nằm trong *package java.awt*.

SWING không phải là một từ viết tắt. SWING là một sản phẩm lớn của Java, một thành viên quan trọng của JFC. SWING bắt đầu được phát triển từ bản Beta của JDK 1.1. Đến năm 1998 thì SWING được công bố rộng rãi.

Khi phát hành SWING 1.0 chứa 250 lớp class và giao diện interface, đến nay SWING 1.4 chứa 451 class và 85 interface. Mặc dù SWING được phát triển riêng rẽ với phần lõi của Java Development Kit, nó đòi hỏi ít nhất JDK 1.1.5 để chạy.

SWING là sự mở rộng của AWT với các cải tiến như sau:

- Tạo giao diện nhất quán độc lập với môi trường (cross-platform GUI);
- Đa luồng và nhẹ với nhiều đặc điểm nâng cấp;
- Có khả năng tùy biến tại thời gian chạy;

- Không sử dụng trộn lẫn AWT và SWING trong cùng một giao diện được. Hiểu theo nghĩa không thể tạo ra một thành phần của AWT và của SWING trong

cùng một cửa sổ được. Khi đó, thành phần của SWING thường sẽ che mất thành phần của AWT.

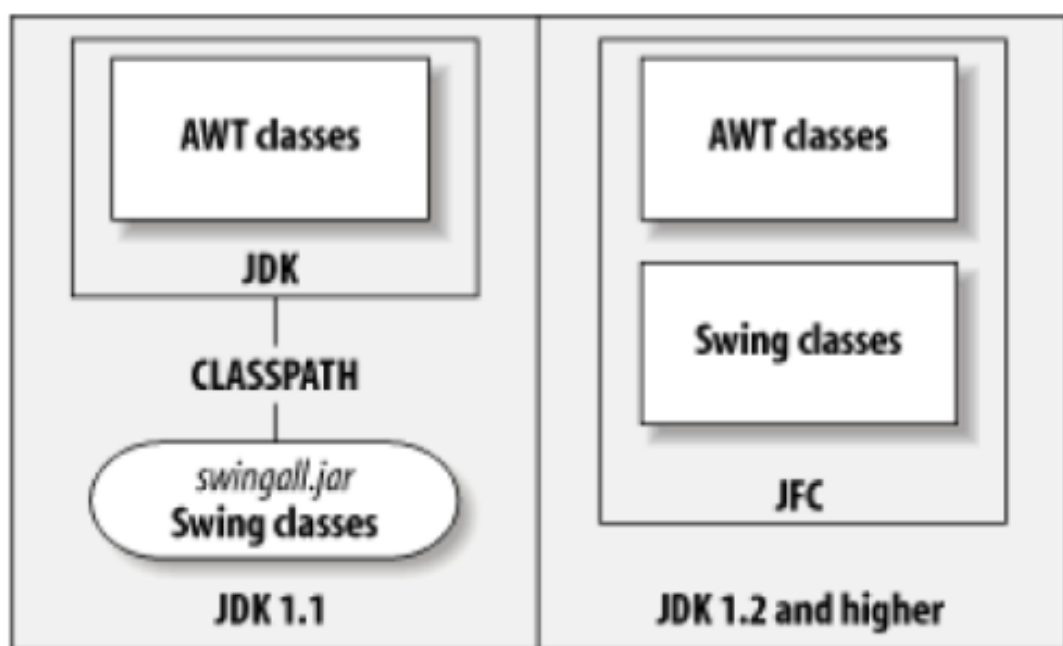
SWING nằm trong package *java.swing*. SWING giúp khắc phục một số khó khăn của AWT khi cần bổ sung các widget mới. Đặc biệt, SWING giúp cho các giao diện trở nên đẹp hơn như MFC của Microsoft. Dưới đây là so sánh giữa AWT và SWING:

AWT	SWING
<ul style="list-style-type: none"> - Xây dựng bằng mã nguồn thường trú - Các thành phần của awt bị lệ thuộc vào hệ điều hành. Các thành phần đồ họa trong awt được xây dựng từ các thành phần đồ họa của hệ điều hành tương ứng, do đó GUI xây dựng bằng awt thì có cảm quan look-and-feel ở mỗi hệ điều hành là khác nhau. - Khó phát triển thêm các thành tố điều khiển widget mới do phụ thuộc vào hệ điều hành. Có thể có widget hoạt động trên hệ điều hành này nhưng bị vô hiệu hóa trên hệ điều hành khác hoặc cùng 1 widget trên 2 hệ điều hành khác nhau lại cho 2 kết quả hoạt động khác nhau. 	<ul style="list-style-type: none"> - Xây dựng hoàn toàn bằng Java API - Các thành phần đồ họa trong được xây dựng từ ngôn ngữ Java. Do đó phần look-and-feel mềm dẻo hơn, có thể đồng nhất trên tất cả các hệ điều hành hoặc tùy biến theo ý muốn của người dùng. - Dễ phát triển các widget, có thể thay đổi diện mạo của widget lúc chạy chương trình - Có thể bổ sung biểu tượng bên cạnh chữ cho Label, Button, Menu item; Tạo đường viền và ghi tựa đề cho các widget; Có thể chọn một thành phần bằng bàn phím thay con chuột; Hỗ trợ Unicode nên có thể đưa tiếng Việt vào các thành phần của swing. - Hoạt động theo mô hình MVC (Model-View-Controller)

Hình 1.5. So sánh AWT và SWING

Trong bảng so sánh trên đã tóm tắt một số điểm chính mô tả sự khác nhau của AWT và SWING. Trong đó có nhắc đến mô hình MVC. Đây là một dạng kiến trúc phần mềm phổ biến hiện nay để chia phần mềm thành 3 phần tương tác được với nhau: Mô hình - Model, Khung nhìn - View và Điều khiển - Controller với các chữ cái đầu ghép lại tạo thành tên kiến trúc MVC. Mô hình quản lý dữ liệu của ứng dụng, hồi đáp các yêu cầu lấy thông tin. Khung nhìn cung cấp giao diện cho người dùng sử dụng ứng dụng. Điều khiển để nhận các đầu vào từ người dùng, gọi các đối tượng của Mô hình và của Khung nhìn. SWING được phát triển sau và dựa vào AWT, nên hoạt động của SWING đã được thiết kế để tuân theo kiến trúc MVC như đã nói.

Như vậy SWING có thay thế cho AWT không? Câu trả lời là Không thể bởi vì thực tế SWING được tạo nên từ phần lõi của AWT. Bởi vì SWING không chứa bất kì mã thường trú dành cho nền tảng hệ điều hành nào (native code). Có thể thấy các xử lý sự kiện vẫn phải sử dụng AWT vì nó tương tác với các phần cứng như màn hình, cửa sổ, con chuột. Hình dưới đây mô tả mối quan hệ của AWT, SWING và JDK như sau:



Hình 1.6. AWT, SWING và JDK

Chương 2. CÁC THÀNH PHẦN ĐỒ HỌA GIAO DIỆN VỚI AWT

I. GIỚI THIỆU

Để lập trình giao diện đồ họa, Java sớm đã phát triển gói AWT. AWT là viết tắt của Abstract Windowing Toolkit, nó là một tập các lớp Java cho phép xây dựng giao diện đồ họa. Các thành phần đồ họa trong AWT được xây dựng từ các thành phần đồ họa của hệ điều hành tương ứng, do đó GUI được xây dựng từ AWT có giao diện được quan sát look-and-feel ở mỗi hệ điều hành là khác nhau. Chương này sẽ giới thiệu đến người đọc về gói AWT với các thành phần cơ bản và các điều khiển cơ bản. Chương đã được biên soạn dựa trên tham khảo các tài liệu số [1], [2], [3], [4] và [5].

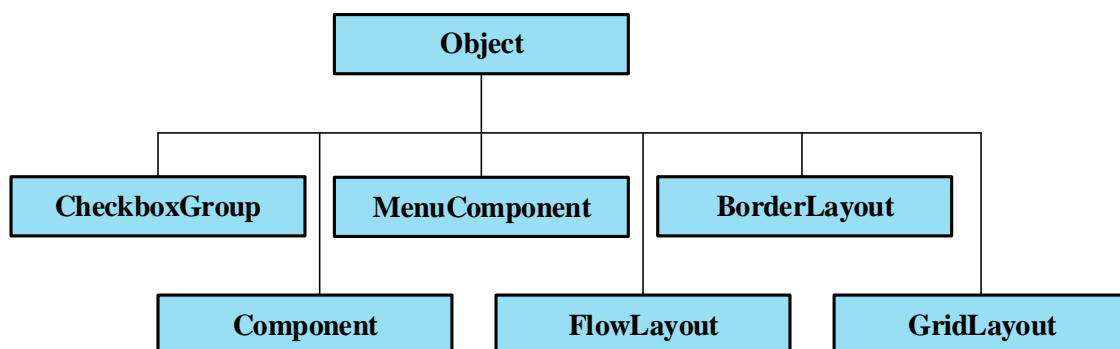
Để gọi gói thư viện này có thể sử dụng câu lệnh: *import java.awt*. AWT là một tập các lớp Java cho phép ta xây dựng giao diện đồ họa. AWT cung cấp các nhóm đối tượng khác nhau để xây dựng giao diện đồ họa bao gồm:

- *Container (khung chứa)*: Là một đối tượng khung chứa dùng để quản lý và nhóm các đối tượng con trong lớp Component;

- *Component (thành phần)*: Là một đối tượng có biểu diễn đồ họa được hiển thị trên màn hình mà người dùng có thể tương tác được. Chẳng hạn như những nút bấm (button), những ô checkbox, những thanh cuộn scrollbar,... Lớp Component là một lớp trừu tượng;

- *Layout manager (quản lý bố cục)*: Khung chứa (Container) nhận các đối tượng từ bên ngoài đưa vào và nó phải biết làm thế nào để tổ chức, sắp xếp vị trí cho các đối tượng đó. Mỗi đối tượng khung chứa đều có một bộ quản lý bố cục;

- *Events (các sự kiện)*: Là các lớp giúp xử lý các sự kiện chuột và bàn phím.



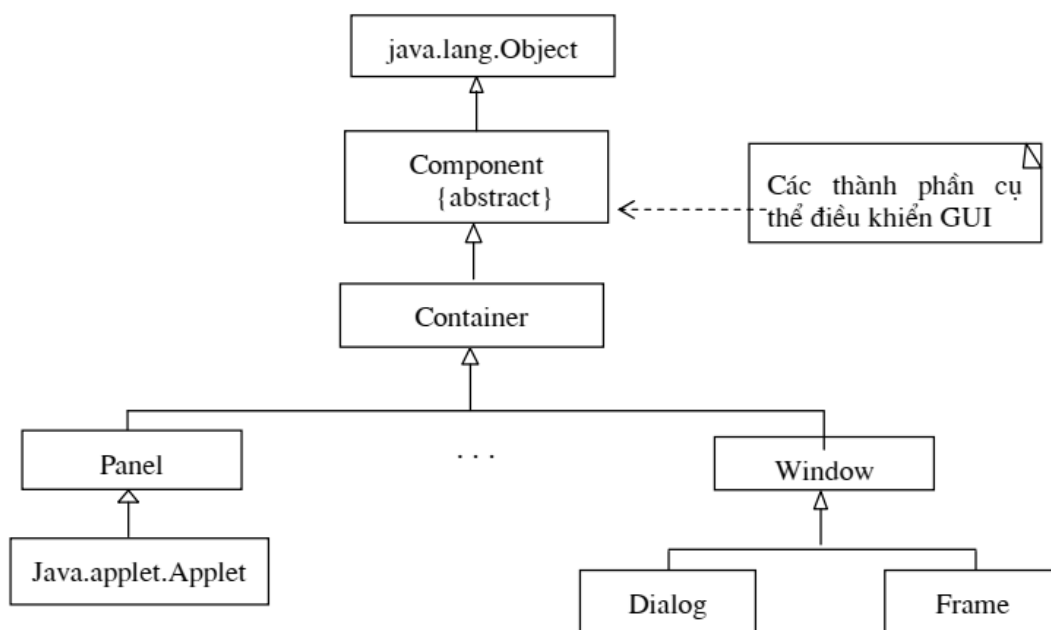
Hình 2.1. Sơ đồ phân cấp các lớp trong gói AWT.

II. CÁC THÀNH PHẦN CỦA AWT

1. Các đối tượng khung chứa (Container)

Container là nơi ta đặt các đối tượng thành phần, nơi các đối tượng thành phần có thể được vẽ, tô màu và hiển thị. Lớp Container này nằm trong *java.awt*. Lớp này trực tiếp hoặc gián tiếp dẫn xuất ra hai đối tượng khung chứa được dùng phổ biến là Frame và Panel.

Hình 2.2 mô tả một phần cấu trúc phân cấp của các lớp theo quan hệ kế thừa, trong đó các lớp con của lớp chứa Container sẽ cung cấp những chức năng chính để phát triển các ứng dụng dựa trên GUI



Hình 2.2. Cây thừa kế mô tả lớp Container và các lớp con của lớp đó

a. Frame

Frame là một cửa sổ độc lập, là lớp con của lớp cửa sổ Window. Nó được hiển thị dưới dạng một cửa sổ độc lập và có biên. Frame có thể chạy độc lập với ứng dụng applet và trình duyệt.

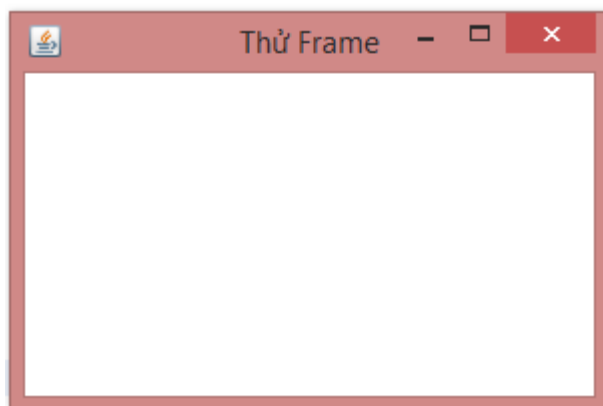
Toán tử khởi tạo:

```
Frame ( )  
Frame (String title)
```

Ví dụ 2.1. Tạo ra một FrameDemo thừa kế từ lớp Frame, khung chứa này sẽ có tên là “Thử Frame”

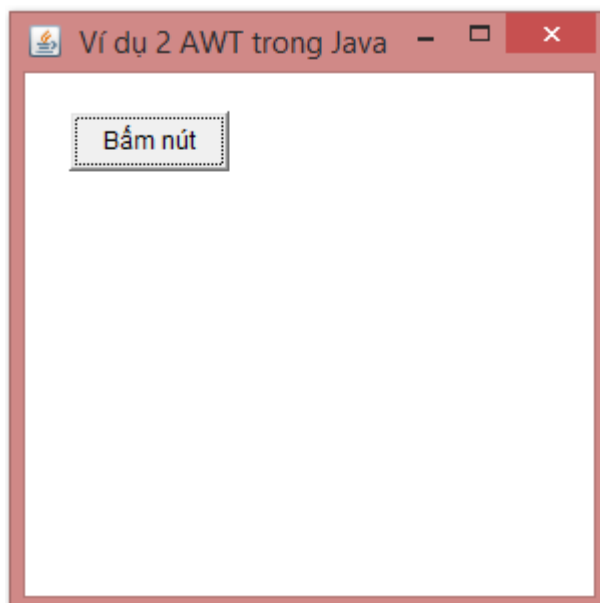
```
import    java.awt.*;  
  
class    FrameDemo    extends Frame {  
    public    FrameDemo (String title){  
        super (title);  
    }  
  
    public    static void main(String arg [ ]) {  
        FrameDemo f= new FrameDemo("Thử Frame");  
        f.setSize (300,200);  
        f.setVisible (true);  
    }  
}
```

Kết quả là một cửa sổ có tên là Thử Frame xuất hiện



Cách thứ 2 chúng ta hoàn toàn có thể tạo ra một đối tượng Frame và khiến cho nó xuất hiện trong hàm main.

```
package GUI.testawt;  
  
import java.awt.Button;  
import java.awt.Frame;  
  
public class Frame2 {  
    Frame2 () {  
        Frame f = new Frame();  
        f.setTitle("Ví dụ 2 AWT trong Java");  
        Button b = new Button("Bấm nút");  
        b.setBounds(30, 50, 80, 30);  
        f.add(b);  
        f.setSize(300, 300);  
        f.setLayout(null);  
        f.setVisible(true);  
    }  
  
    public static void main(String args[]) {  
        new Frame2 ();  
    }  
}
```



b. Panel

Panel là một lớp container đơn giản nhất. Nó cung cấp không gian trong đó một ứng dụng có thể đính kèm bất kỳ thành phần nào khác. Nó kế thừa lớp Container. Panel là vùng chứa nằm trong cửa sổ (có thể là cửa sổ độc lập hay vùng applet do trình duyệt cung cấp) và không có biên. Nó được dùng để nhóm một số các đối tượng thành phần.

Cú pháp của Panel:

```
public class Panel extends Container implements Accessible
```

Phương thức khởi tạo

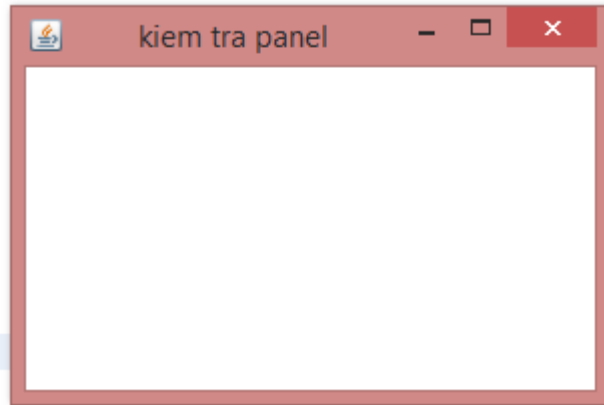
```
Panel ()
```

Ví dụ 2.2. Tạo mới một Panel, sau đó thêm nó vào một Frame

```
import java.awt.*;

class Paneltest extends Panel {
    public static void main (String arg [ ]) {
        Paneltest p= new Paneltest ( );
        Frame f= new frame ("kiem tra panel");
        f.add (p);
        f.setSize (300,200);
        f.setVisible (true);
    }
    public Paneltest ( )
    {
    }
}
```

Kết quả:



Không thể quan sát trực tiếp một Panel, vì thế, ta phải đưa nó vào trong một Frame để hiển thị nó lên cửa sổ.

Ví dụ 2.3. Tạo một Panel và add vào cửa sổ

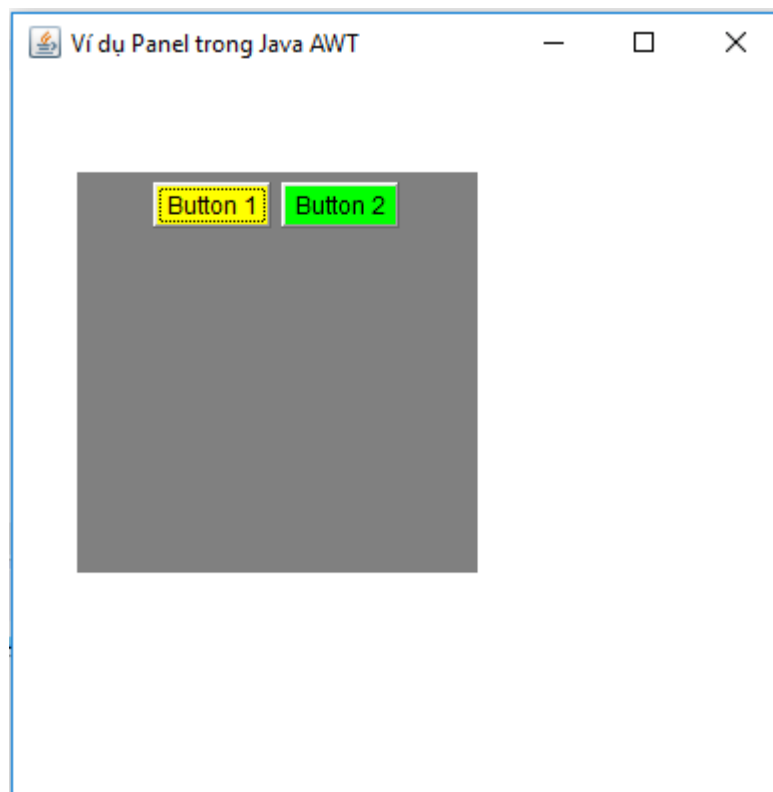
```
package vn.viettuts.awt;

import java.awt.Button;
import java.awt.Color;
import java.awt.Frame;
import java.awt.Panel;

public class PanelExample {
    PanelExample() {
        Frame frame = new Frame("Ví dụ Panel trong Java AWT");
        Panel panel = new Panel();
        panel.setBounds(40, 80, 200, 200);
        panel.setBackground(Color.gray);
        Button button1 = new Button("Button 1");
        button1.setBounds(50, 100, 80, 30);
        button1.setBackground(Color.yellow);
        Button button2 = new Button("Button 2");
        button2.setBounds(100, 100, 80, 30);
        button2.setBackground(Color.green);
        panel.add(button1);
    }
}
```

```
        panel.add(button2);  
        frame.add(panel);  
        frame.setSize(400, 400);  
        frame.setLayout(null);  
        frame.setVisible(true);  
    }  
  
    public static void main(String args[]) {  
        new PanelExample();  
    }  
}
```

Kết quả: Panel chứa 2 đối tượng Button 1 và Button 2 (2 nút nhấn). Panel có kích thước 200x200 trong khi Frame có kích thước 400x400. Như vậy 1 Frame hoàn toàn có thể chứa nhiều Panel khác nhau.



c. Dialog

Lớp Dialog giống như Frame, là một lớp con của Window. Dialog (hộp thoại) đại diện cho một cửa sổ cấp cao nhất với một đường viền và một tiêu đề được sử dụng để tạo hộp thoại lấy thông tin từ người dùng.

Đối tượng Dialog được cấu trúc như ví dụ dưới đây:

Ví dụ 2.4. Tạo một hộp thoại Dialog

```
package GUI.testawt;

import java.awt.Button;
import java.awt.Dialog;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Label;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class DialogExample {
    private static Dialog dialog;

    public DialogExample() {
        Frame frame = new Frame();
        dialog = new Dialog(frame, "Ví dụ Dialog trong Java AWT",
true);

        dialog.setLayout(new FlowLayout());
        Button button = new Button("Close");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                DialogExample.dialog.setVisible(false);
            }
        });
    }
}
```

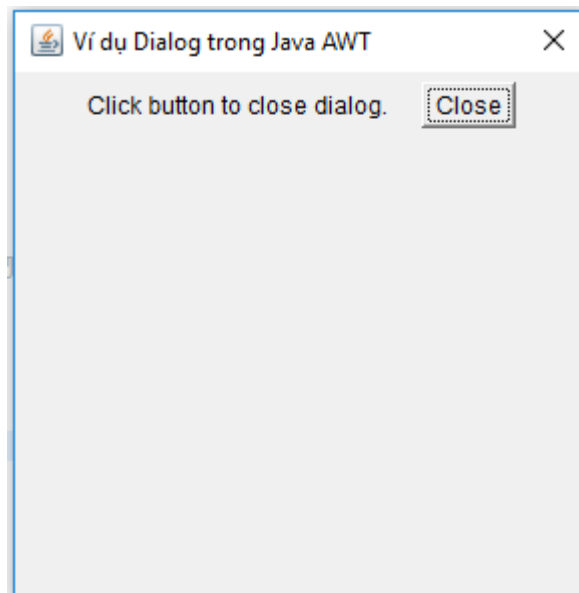
```

        dialog.add(new Label("Click button to close dialog.));
        dialog.add(button);
        dialog.setSize(300, 300);
        dialog.setVisible(true);
    }

    public static void main(String args[]) {
        new DialogExample();
    }
}

```

Kết quả:



d. Các bước chung nhất để tạo ra một cửa sổ

Có thể có nhiều cách lập trình khác nhau để tạo ra một cửa sổ, nhưng có thể sử dụng một kịch bản chung để xây dựng giao diện ứng dụng là:

1. Tạo ra một frame có tên, ví dụ “My Frame” :

```
Frame guiFrame = new Frame("My Frame");
```

2. Xây dựng một cấu trúc phân cấp các thành phần bằng cách sử dụng hàm add() để bổ sung thêm panel hoặc những thành phần điều khiển GUI vào cấu trúc đó:

```
guiFrame.add(new Button("OK")); // Đưa vào một nút (Button) có tên "OK"
```

3. Đặt lại kích thước cho frame sử dụng hàm setSize():

```
guiFrame.setSize(200, 300); // Đặt lại khung frame là 200 x 300
```

4. Làm cho frame đó nhìn thấy được:

```
guiFrame.setVisible(true);
```

2. Các đối tượng thành phần Component

Các ví dụ 2.1, 2.3 và 2.4 đã vẽ các cửa sổ bên trong có chứa nút nhấn và nhãn. Nút nhấn và nhãn chính là hai trong nhiều đối tượng thành phần thường gặp trong GUI. Để tạo ra các đối tượng thành phần này, lớp Component được xây dựng. Các đối tượng thành phần đều là con của lớp Component này.

Các đối tượng thành phần có thể được dùng để tạo giao diện người dùng, và có thể được thay đổi kích cỡ, làm cho ẩn đi hay được hiện lên. Ví dụ: TextField, Label, Checkbox, TextArea, Scrollbar, Scrollpane là các đối tượng thành phần.

Lớp Component cung cấp các hàm tiện ích chung cho các lớp con của nó.

```
Dimension getSize()  
void setSize(int width, int height)  
void setSize(Dimension d)
```

Hàm getSize() cho lại đối tượng thuộc lớp Dimension gồm width (chiều rộng), height (chiều cao) xác định kích thước của một thành phần tính theo pixel.

Hàm setSize() đặt lại kích thước của thành phần.

```
Point getLocation()  
void setLocation(int x, int y)  
void setLocation(Point p)
```