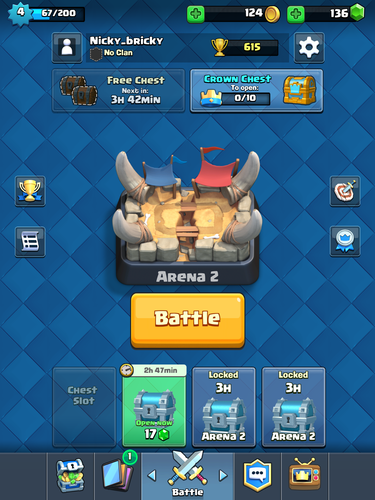
Layouts Description

# Layout 1: Group Layout

Group layout is the most complex and most customizable of Java’s default layouts. The arrangement of components in a group layout is determined using two things: horizontal layout and vertical layout. Components that are arranged on the same baseline along the layout dimension (horizontal line or vertical line) are in sequential arrangement. Components that are stacked on top of each other along the layout dimension are in parallel arrangement. A sequential arrangement can be part of a vertical arrangement and vice versa, which means that arrangements can be nested. By defining the sequential arrangement of components in both the horizontal and vertical layouts as well as any parallel subgroups, an organized arrangement of components can be created.

Gaps can be added to adjust the distance between one component and the next (using addPreferredGap(int pixels)) and between one component and the container border (using addContainerGap(int pixels)). Group layout can automatically create appropriate gaps if when set up the commands setAutoCreateGaps(true) and setAutoCreateContainerGaps(true) are used.

Group layout is used to design user interfaces that must be carefully aligned to look clean and appealing to the user. For example, group layout could be used to design menus for a game. An example of how group layout can be used to make such a menu look nice is the menu from the mobile game Clash Royale.

It is obvious that many of the buttons and text fields in the menu are aligned by row. However, the alignment of the central area (profile icon, profile info, settings button, free chest and crown chest buttons, and bottom row of chest slots) is extremely intricate and is best accomplished by using the arrangement and spacing options of GridLayout. The profile icon, free chest button, and first chest slot are all aligned along one vertical baseline on the left. The settings icon, crown chest, and fourth chest slot are all aligned along one vertical baseline on the right. Ignoring the elements that have not been mentioned so far, the horizontal layout would be a sequential arrangement consisting of three elements: a parallel layout including the profile icon, free chest button, and first chest slot aligned to the left, the profile info bar, and a parallel layout including the settings icon, the crown chest button, and the fourth chest slot aligned to the right. The positions of the second and third chest slots could be specified in the vertical layout as part of a parallel layout with the first and fourth chest slots, and the preferred spacing between each chest slot component could be adjusted so the spacing is uniform.

The main caution/problem with Group Layout is that it is very difficult to use. If the horizontal and vertical layouts conflict it is possible to have overlapping components. If there is a mistake in arranging a group layout it can be very difficult to determine exactly where the mistake was made. Review the method of using the layout on The Java Tutorials before using it. GUI builders automatically generate grid layouts and are included with many IDEs (including NetBeans), which can make the process much easier.

# Layout 2: Flow Layout

Flow layout arranges the components in a row, with each component taking up the space that it needs to in the layout. If there are too many components to put in one row, components will be added to additional rows arranged in the same manner. There is constant horizontal and vertical spacing between components, which can be changed using the setHgap(int) and setVgap(int) methods. By default, the components will be center-aligned.

Flow layouts are best used within another layout to format single rows of similar components (e.g. a row of buttons) or if there is a group of one component. Professional UI designs shouldn’t rely on flow layouts for the main parts of their design and should use GroupLayout instead.

The main issues with FlowLayout are that it is not very customizable and that it changes when components are resized.