Hanoi University Of Science And Technology

Sorting Visualizer

Group 12

Members and Assignments

1. Bùi Nguyễn Minh 20226055
   - Managing project and other member's assignments
   - Designing class diagram
   - Designing main Menu class and bubbleSort visualizer
   - Some update for visualizer screen
2. Nguyễn Công Minh -20226056
   - Designing use case diagram for project
   - Designing visualizer screen and some buttons with effects
   - Creating abstract class Sorting and InsertionSort Visualizer
   - Making presentation slide
3. Phạm Duy Hoàng - 20226042
   - Designing SelectionSort Visualizer
   - Video demo
4. Lê Văn Hậu - 20226038
   - Designing MergeSort and MergeSort Visualizer
   - Writing report for project

Topic of Project: Demonstration of sorting algorithms on an array

Problem: Visualize 4 sorting algorithms: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort

Request:

- Visualize sorting algorithms easily to understand and provide various ways to create array as manual creation, creating some basic kinds of array as nearly sorted or reverse array.
- **Must** use pure array, do not use Arrays that provided by JAVA itself.

Use Case Diagram

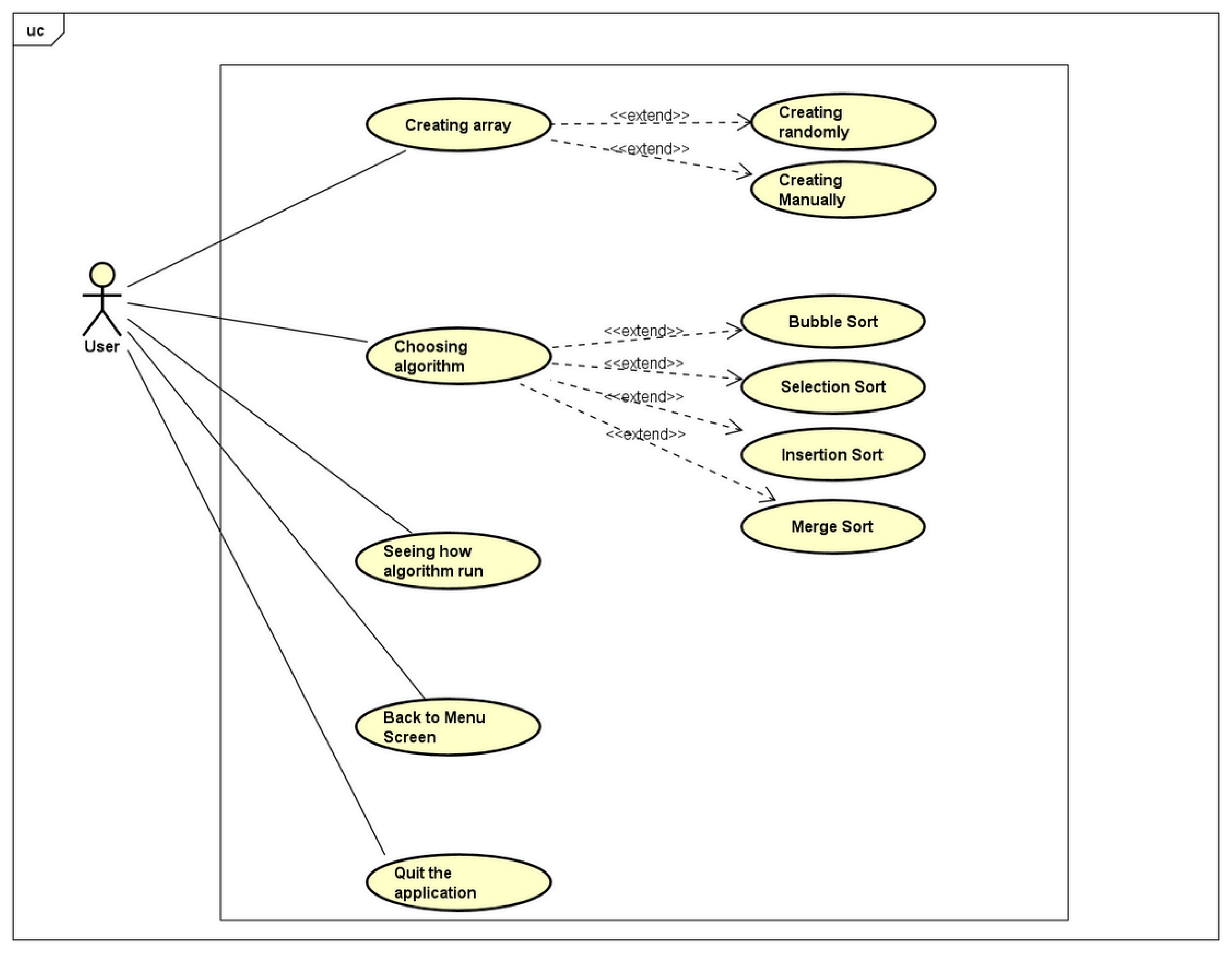
- Creating array: Allow user to choose to create array randomly or manually

- Choosing algorithm: There are 4 algorithms to choose: Bubble, Selection, Insertion, Merge

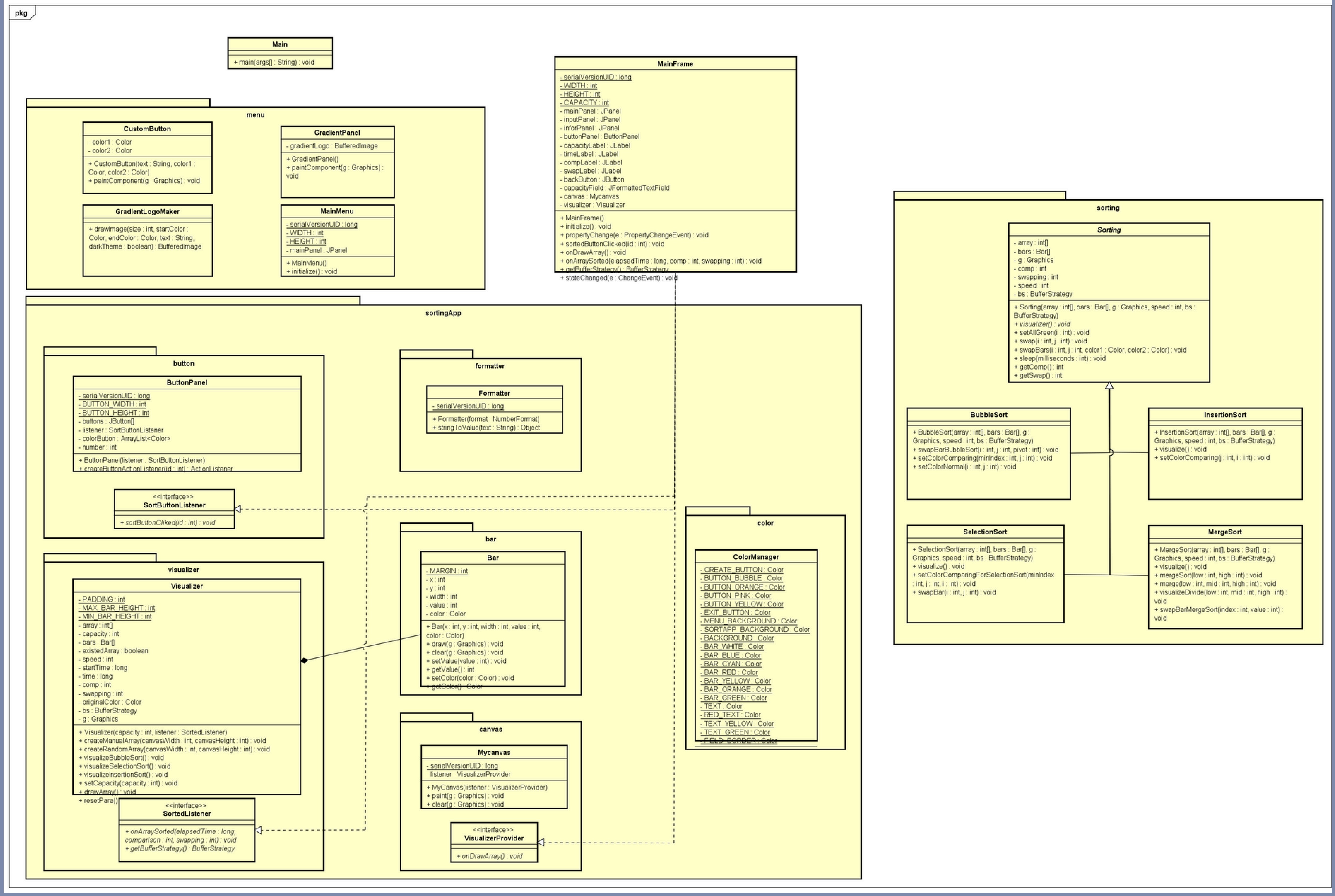
- Seeing how algorithm runs: Understanding how algorithm works by seeing it performed by sorting bars.

- Back to menu screen: It helps us to back to menu

- Quit the application



General Class Diagram



pkg

Main

+ main(args[] : String) : void

MainFrame

- serialVersionUID : long
- WIDTH : int
- HEIGHT : int
- CAPACITY : int
- mainPanel : JPanel
- inputPanel : JPanel
- inforPanel : JPanel
- buttonPanel : ButtonPanel
- capacityLabel : JLabel
- timeLabel : JLabel
- compLabel : JLabel
- swapLabel : JLabel
- backButton : JButton
- capacityField : JFormattedTextField
- canvas : Mycanvas
- visualizer : Visualizer

+ MainFrame()
+ initialize() : void
+ propertyChange(e : PropertyChangeEvent) : void
+ sortedButtonClicked(id : int) : void
+ onDrawArray() : void
+ onArraySorted(elapsedTime : long, comp : int, swapping : int) : void
+ getBufferStrategy() : BufferStrategy
+ stateChanged(e : ChangeEvent) : void

menu

CustomButton

- color1 : Color
- color2 : Color

+ CustomButton(text : String, color1 : Color, color2 : Color)
+ paintComponent(g : Graphics) : void

GradientPanel

- gradientLogo : BufferedImage

+ GradientPanel()
+ paintComponent(g : Graphics) : void

GradientLogoMaker

+ drawImage(size : int, startColor : Color, endColor : Color, text : String, darkTheme : boolean) : BufferedImage

MainMenu

- serialVersionUID : long
- WIDTH : int
- HEIGHT : int
- mainPanel : JPanel

+ MainMenu()
+ initialize() : void

sortingApp

button

ButtonPanel

- serialVersionUID : long
- BUTTON_WIDTH : int
- BUTTON_HEIGHT : int
- buttons : JButton[]
- listener : SortButtonListener
- colorButton : ArrayList<Color>
- number : int

+ ButtonPanel(listener : SortButtonListener)
+ createButtonActionListener(id : int) : ActionListener

<<interface>>
SortButtonListener

+ *sortButtonCliked(id : int) : void*

formatter

Formatter

- serialVersionUID : long

+ Formatter(format : NumberFormat)
+ stringToValue(text : String) : Object

visualizer

Visualizer

- PADDING : int
- MAX_BAR_HEIGHT : int
- MIN_BAR_HEIGHT : int
- array : int[]
- capacity : int
- bars : Bar[]
- existedArray : boolean
- speed : int
- startTime : long
- time : long
- comp : int
- swapping : int
- originalColor : Color
- bs : BufferStrategy
- g : Graphics

+ Visualizer(capacity : int, listener : SortedListener)
+ createManualArray(canvasWidth : int, canvasHeight : int) : void
+ createRandomArray(canvasWidth : int, canvasHeight : int) : void
+ visualizeBubbleSort() : void
+ visualizeSelectionSort() : void
+ visualizeInsertionSort() : void
+ setCapacity(capacity : int) : void
+ drawArray() : void
+ resetPara()

<<interface>>
SortedListener

+ *onArraySorted(elapsedTime : long, comparison : int, swapping : int) : void*
+ *getBufferStrategy() : BufferStrategy*

bar

Bar

- MARGIN : int
- x : int
- y : int
- width : int
- value : int
- color : Color

+ Bar(x : int, y : int, width : int, value : int, color : Color)
+ draw(g : Graphics) : void
+ clear(g : Graphics) : void
+ setValue(value : int) : void
+ getValue() : int
+ setColor(color : Color) : void
+ getColor() : Color

canvas

Mycanvas

- serialVersionUID : long
- listener : VisualizerProvider

+ MyCanvas(listener : VisualizerProvider)
+ paint(g : Graphics) : void
+ clear(g : Graphics) : void

<<interface>>
VisualizerProvider

+ *onDrawArray() : void*

color

ColorManager

- CREATE_BUTTON : Color
- BUTTON_BUBBLE : Color
- BUTTON_ORANGE : Color
- BUTTON_PINK : Color
- BUTTON_YELLOW : Color
- EXIT_BUTTON : Color
- MENU_BACKGROUND : Color
- SORTAPP_BACKGROUND : Color
- BACKGROUND : Color
- BAR_WHITE : Color
- BAR_BLUE : Color
- BAR_CYAN : Color
- BAR_RED : Color
- BAR_YELLOW : Color
- BAR_ORANGE : Color
- BAR_GREEN : Color
- TEXT : Color
- RED_TEXT : Color
- TEXT_YELLOW : Color
- TEXT_GREEN : Color
- FIELD_BORDER : Color

sorting

Sorting

- array : int[]
- bars : Bar[]
- g : Graphics
- comp : int
- swapping : int
- speed : int
- bs : BufferStrategy

+ Sorting(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ *visualizer() : void*
+ setAllGreen(i : int) : void
+ swap(i : int, j : int) : void
+ swapBars(i : int, j : int, color1 : Color, color2 : Color) : void
+ sleep(milliseconds : int) : void
+ getComp() : int
+ getSwap() : int

BubbleSort

+ BubbleSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ swapBarBubbleSort(i : int, j : int, pivot : int) : void
+ setColorComparing(minIndex : int, j : int) : void
+ setColorNormal(i : int, j : int) : void

InsertionSort

+ InsertionSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ visualize() : void
+ setColorComparing(j : int, i : int) : void

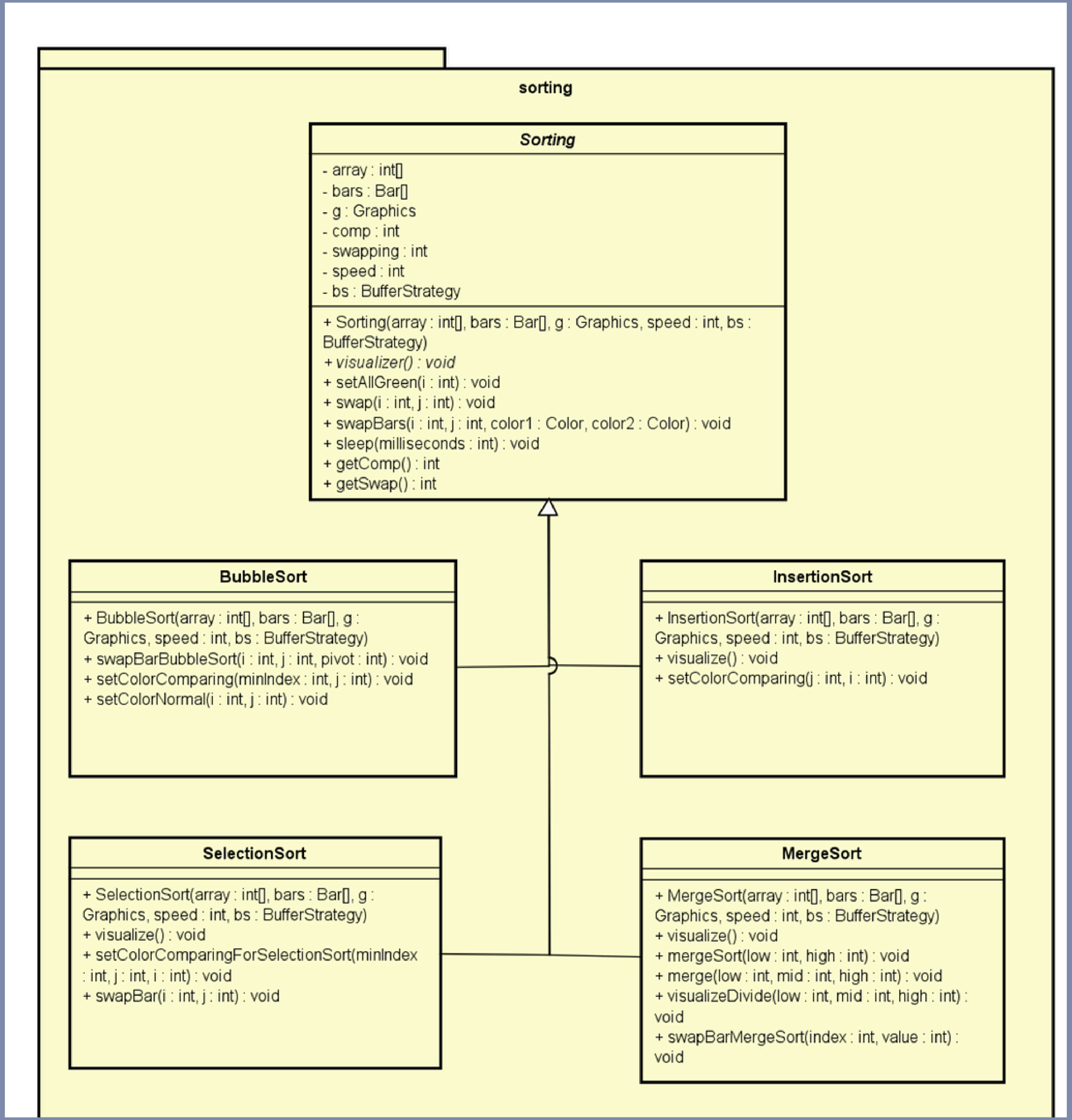
SelectionSort

+ SelectionSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ visualize() : void
+ setColorComparingForSelectionSort(minIndex : int, j : int, i : int) : void
+ swapBar(i : int, j : int) : void

MergeSort

+ MergeSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ visualize() : void
+ mergeSort(low : int, high : int) : void
+ merge(low : int, mid : int, high : int) : void
+ visualizeDivide(low : int, mid : int, high : int) : void
+ swapBarMergeSort(index : int, value : int) : void

Class diagram for package sorting



sorting

Sorting

- array : int[]
- bars : Bar[]
- g : Graphics
- comp : int
- swapping : int
- speed : int
- bs : BufferStrategy

+ Sorting(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ *visualizer() : void*
+ setAllGreen(i : int) : void
+ swap(i : int, j : int) : void
+ swapBars(i : int, j : int, color1 : Color, color2 : Color) : void
+ sleep(milliseconds : int) : void
+ getComp() : int
+ getSwap() : int

BubbleSort

+ BubbleSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ swapBarBubbleSort(i : int, j : int, pivot : int) : void
+ setColorComparing(minIndex : int, j : int) : void
+ setColorNormal(i : int, j : int) : void

InsertionSort

+ InsertionSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ visualize() : void
+ setColorComparing(j : int, i : int) : void

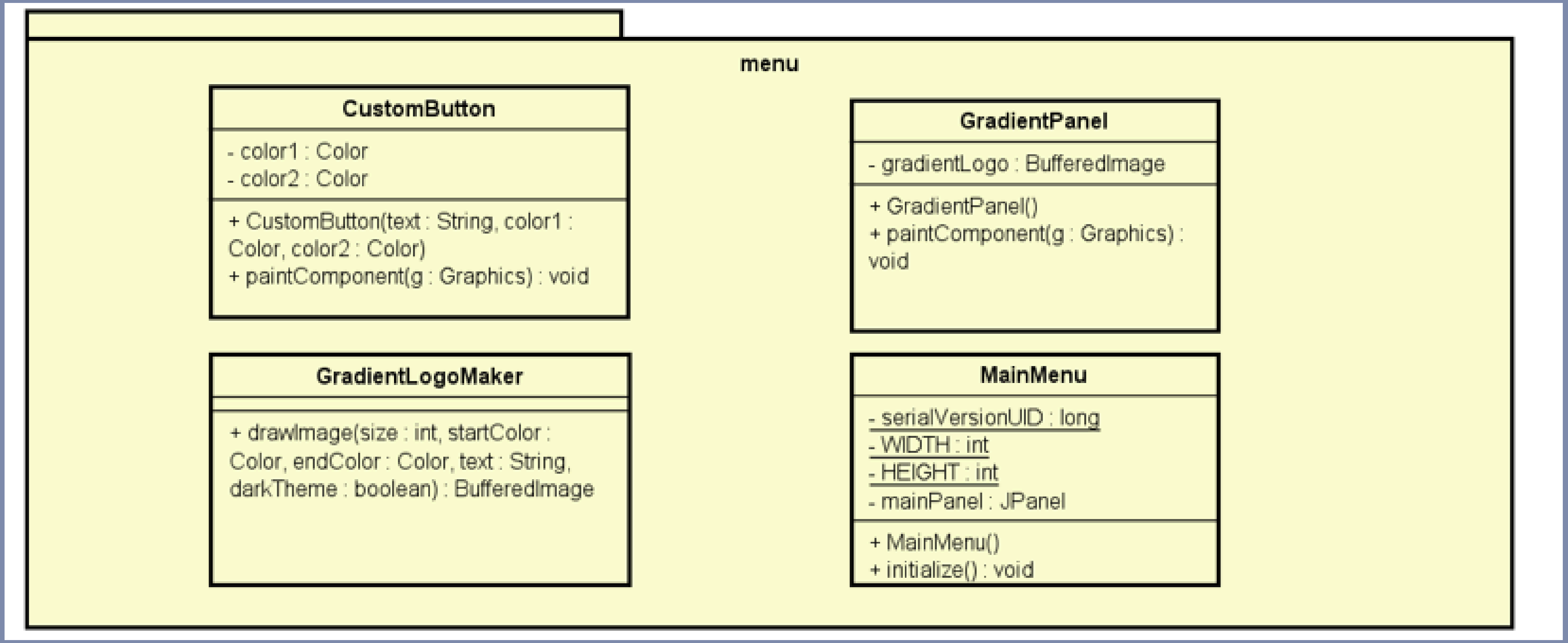
SelectionSort

+ SelectionSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ visualize() : void
+ setColorComparingForSelectionSort(minIndex : int, j : int, i : int) : void
+ swapBar(i : int, j : int) : void

MergeSort

+ MergeSort(array : int[], bars : Bar[], g : Graphics, speed : int, bs : BufferStrategy)
+ visualize() : void
+ mergeSort(low : int, high : int) : void
+ merge(low : int, mid : int, high : int) : void
+ visualizeDivide(low : int, mid : int, high : int) : void
+ swapBarMergeSort(index : int, value : int) : void

Class Diagram for package menu



menu

CustomButton

- color1 : Color
- color2 : Color

+ CustomButton(text : String, color1 : Color, color2 : Color)
+ paintComponent(g : Graphics) : void

GradientPanel

- gradientLogo : BufferedImage

+ GradientPanel()
+ paintComponent(g : Graphics) : void

GradientLogoMaker

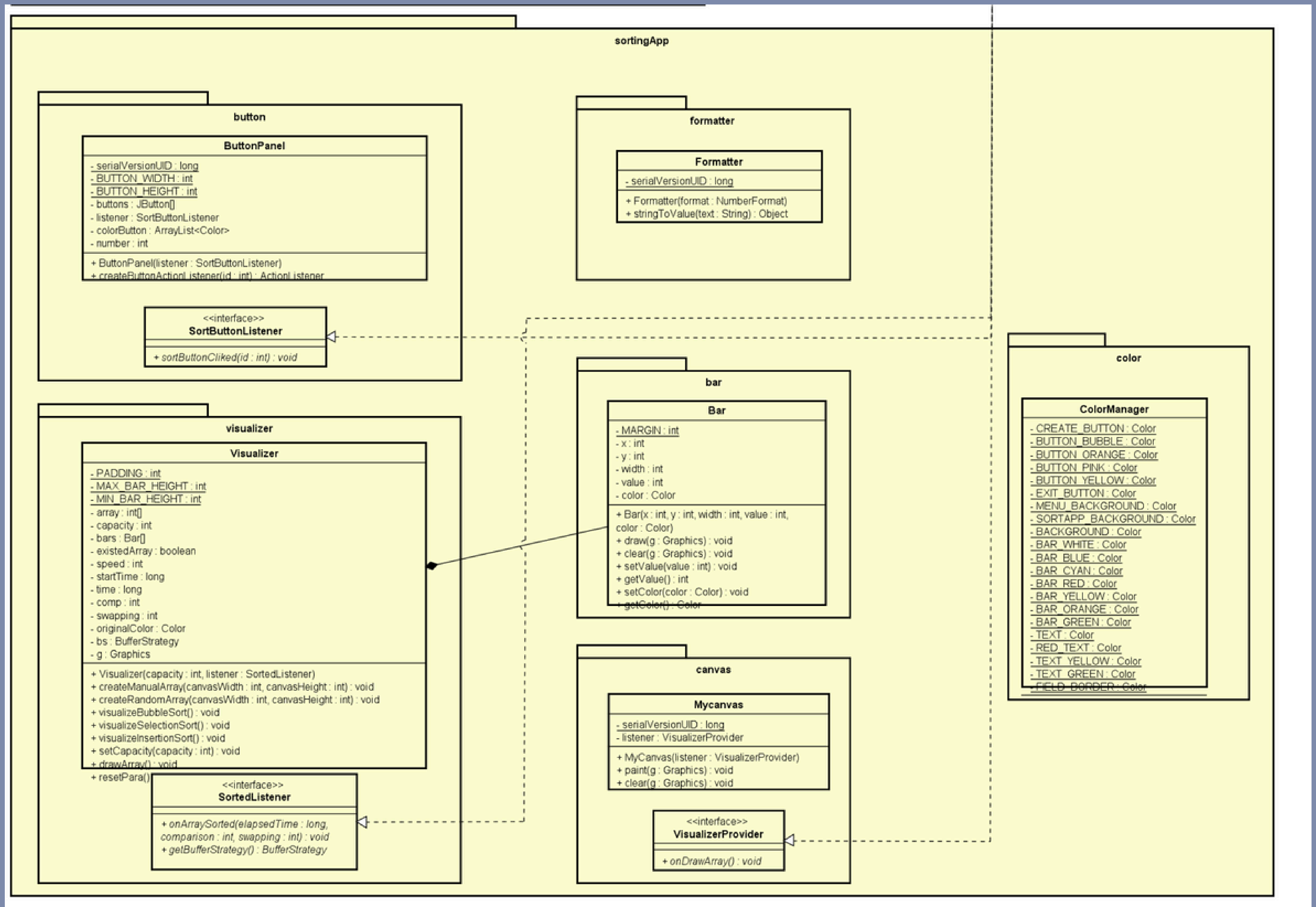
+ drawImage(size : int, startColor : Color, endColor : Color, text : String, darkTheme : boolean) : BufferedImage

MainMenu

- serialVersionUID : long
- WIDTH : int
- HEIGHT : int
- mainPanel : JPanel

+ MainMenu()
+ initialize() : void

Class Diagram for package sortingApp


sortingApp
button
ButtonPanel
- serialVersionUID : long
- BUTTON_WIDTH : int
- BUTTON_HEIGHT : int
- buttons : JButton[]
- listener : SortButtonListener
- colorButton : ArrayList<Color>
- number : int
+ ButtonPanel(listener : SortButtonListener)
+ createButtonActionListener(id : int) : ActionListener
<<interface>>
SortButtonListener
+ sortButtonCliked(id : int) : void
formatter
Formatter
- serialVersionUID : long
+ Formatter(format : NumberFormat)
+ stringToValue(text : String) : Object
visualizer
Visualizer
- PADDING : int
- MAX_BAR_HEIGHT : int
- MIN_BAR_HEIGHT : int
- array : int[]
- capacity : int
- bars : Bar[]
- existedArray : boolean
- speed : int
- startTime : long
- time : long
- comp : int
- swapping : int
- originalColor : Color
- bs : BufferStrategy
- g : Graphics
+ Visualizer(capacity : int, listener : SortedListener)
+ createManualArray(canvasWidth : int, canvasHeight : int) : void
+ createRandomArray(canvasWidth : int, canvasHeight : int) : void
+ visualizeBubbleSort() : void
+ visualizeSelectionSort() : void
+ visualizeInsertionSort() : void
+ setCapacity(capacity : int) : void
+ drawArray() : void
+ resetPara()
<<interface>>
SortedListener
+ onArraySorted(elapsedTime : long, comparison : int, swapping : int) : void
+ getBufferStrategy() : BufferStrategy
bar
Bar
- MARGIN : int
- x : int
- y : int
- width : int
- value : int
- color : Color
+ Bar(x : int, y : int, width : int, value : int, color : Color)
+ draw(g : Graphics) : void
+ clear(g : Graphics) : void
+ setValue(value : int) : void
+ getValue() : int
+ setColor(color : Color) : void
+ getColor() : Color
canvas
Mycanvas
- serialVersionUID : long
- listener : VisualizerProvider
+ MyCanvas(listener : VisualizerProvider)
+ paint(g : Graphics) : void
+ clear(g : Graphics) : void
<<interface>>
VisualizerProvider
+ onDrawArray() : void
color
ColorManager
- CREATE_BUTTON : Color
- BUTTON_BUBBLE : Color
- BUTTON_ORANGE : Color
- BUTTON_PINK : Color
- BUTTON_YELLOW : Color
- EXIT_BUTTON : Color
- MENU_BACKGROUND : Color
- SORTAPP_BACKGROUND : Color
- BACKGROUND : Color
- BAR_WHITE : Color
- BAR_BLUE : Color
- BAR_CYAN : Color
- BAR_RED : Color
- BAR_YELLOW : Color
- BAR_ORANGE : Color
- BAR_GREEN : Color
- TEXT : Color
- RED_TEXT : Color
- TEXT_YELLOW : Color
- TEXT_GREEN : Color
- FIELD_BORDER : Color

OOP Techniques used for project

1. Inheritance

- BubbleSort, SelectionSort, InsertionSort, MergeSort inherits from abtract Sorting class
- Formatter inherits from javax.swing.text.NumberFormatter.
- ButtonPanel inherits from javax.swing.JPanel
- MyCanvas inherits from java.awt.Canvas.
- MainMenu inherits from javax.swing.JFrame
- CustomButton inherits from javax.swing.JButton
- GradientPanel inherits from javax.swing.JPanel

2. Overriding

- MyCanvas overrides paint() method from java.awt.Canvas
- Formatter overides stringToValue methd from javax.swing.text.InternationalFormatter

OOP Techniques used for project

3. Aggregation

- Visualizer has bars
- MainMenu has MainFrame
- Visualizer use createManualArray, createRandomArray, visualizerBubbleSort, clear....

4. Interface

- ButtonPanel interacts with MainFrame via SortButtonListener
- MyCanvas interacts with MainFrame via VisualizerProvider
- Visualizer interacts with MainFrame via the SortedListener

Sorting algorithm

Bubble Sort

1. **Best case:**

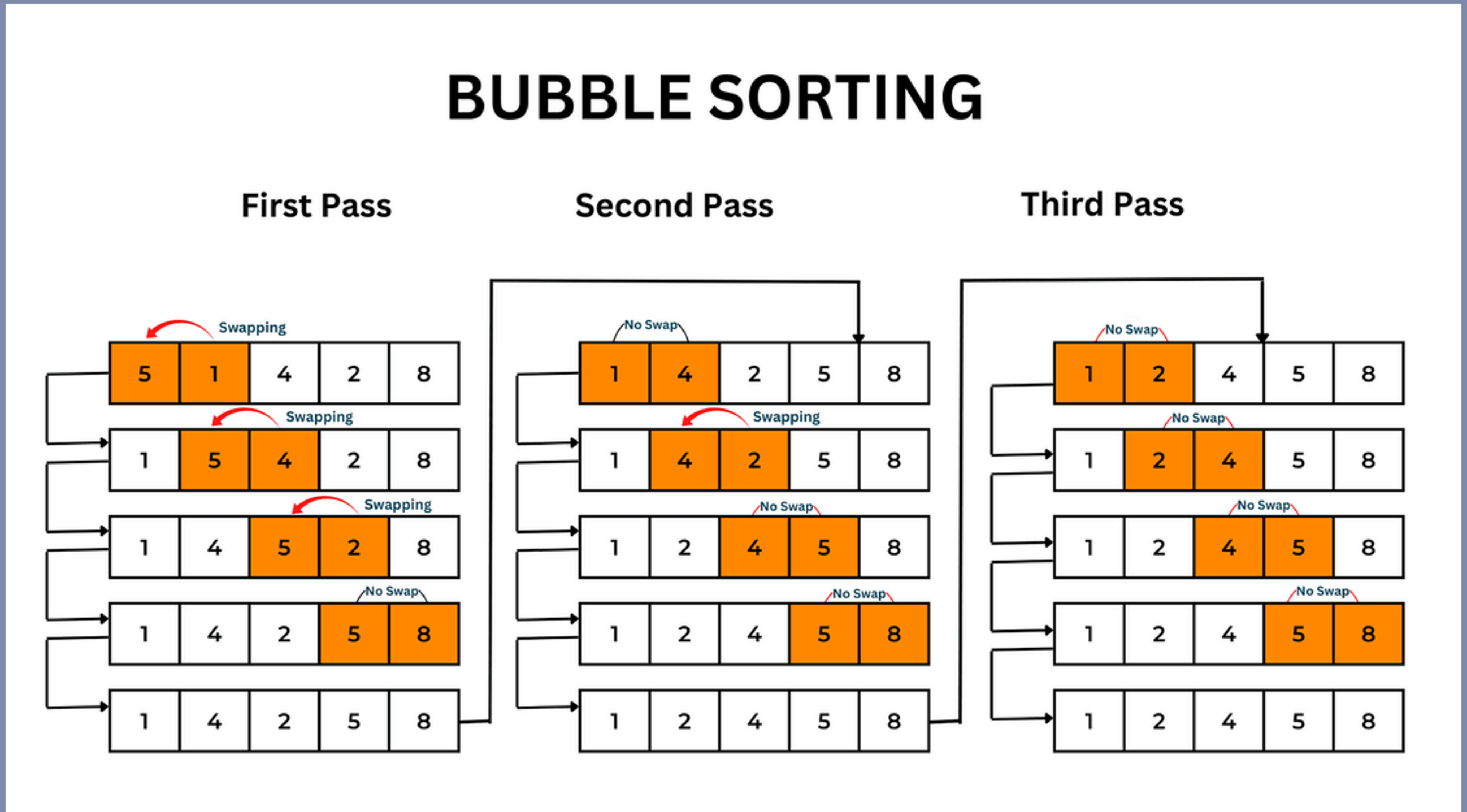
- Complexity: O(n)
- When the array is already sorted

2. **Average case:**

- Complexity: O(n^2)
- Each element needs to be compared with every other element in the array.

3. Worst case:

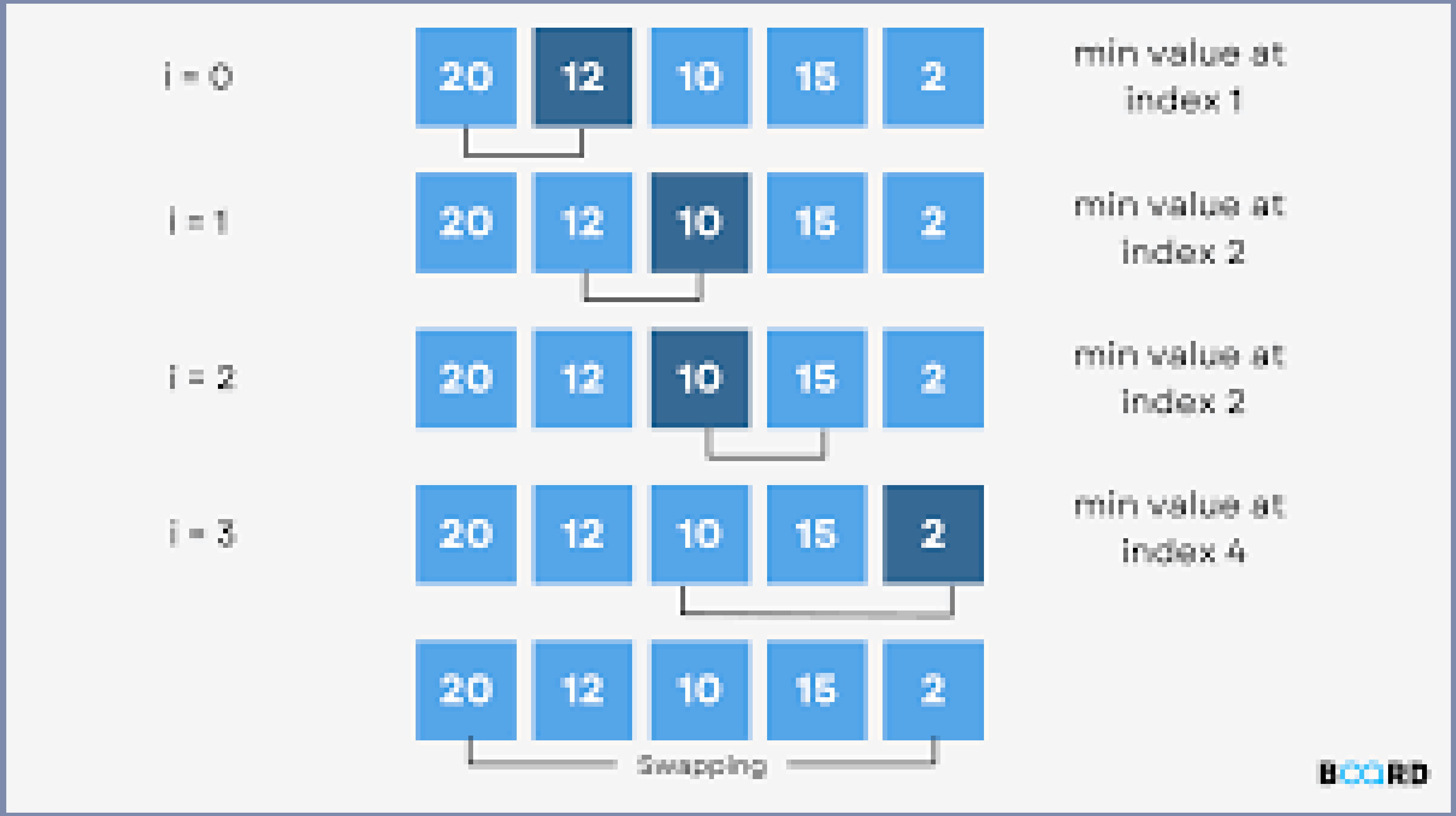
- Complexity: O(n^2)
- When array is at reversed sort.



Sorting Algorithm

Selection Sort

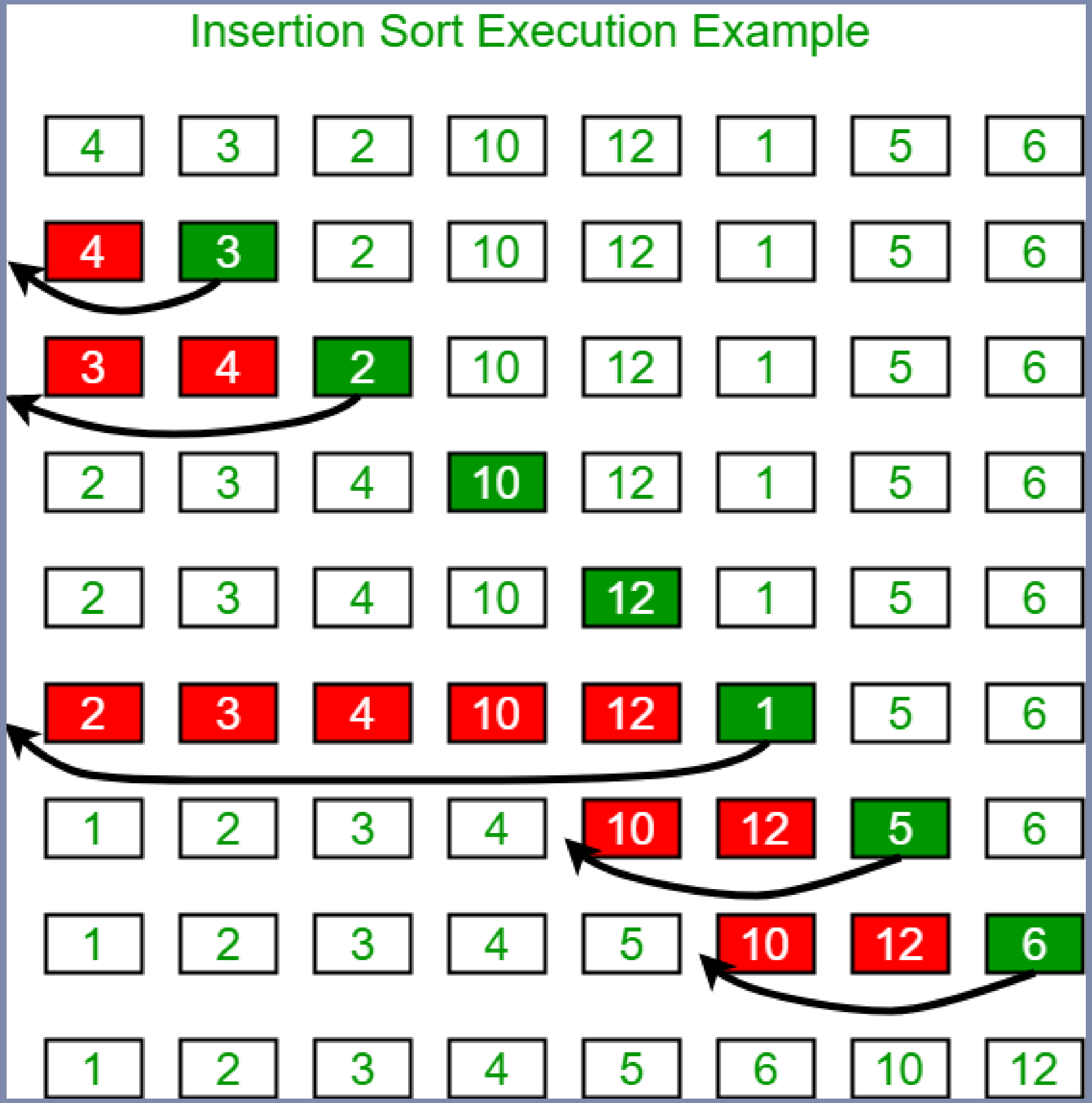
- Complexity always at O(n^2)
- Because in all case, this algorithm will need n*(n-1)/2 comparisons



Sorting algorithm

Insertion Sort

1. **Best case:**
   - Complexity: O(n)
   - When the array is already sorted
2. **Average case:**
   - Complexity: O(n^2)
   - Each element has to be compared with half of the other elements in the array.
3. **Worst case:**
   - Complexity: O(n^2)
   - When array is at reversed sort.



Insertion Sort Execution Example

4 3 2 10 12 1 5 6

4 3 2 10 12 1 5 6

3 4 2 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

1 2 3 4 10 12 5 6

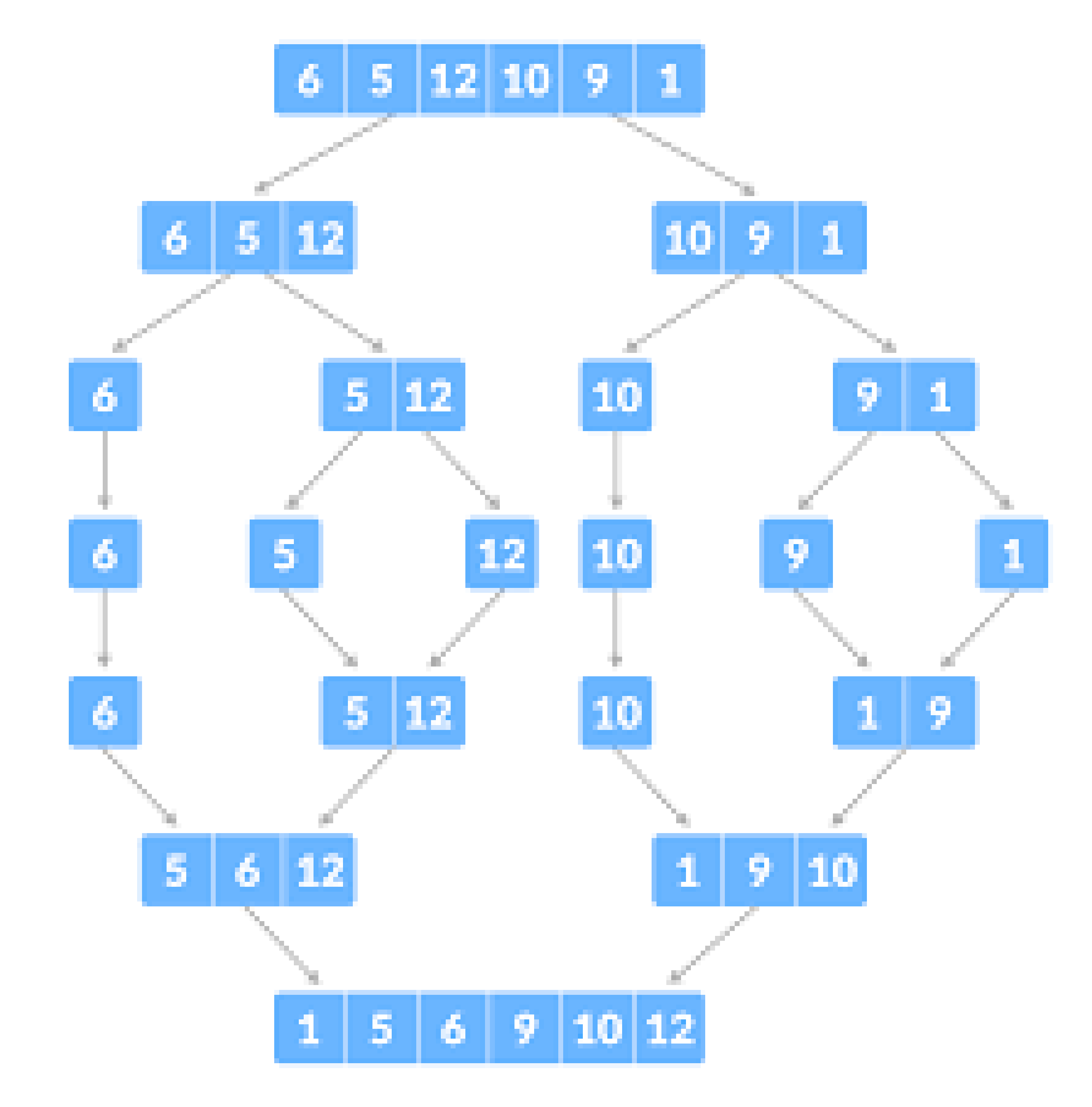
1 2 3 4 5 10 12 6

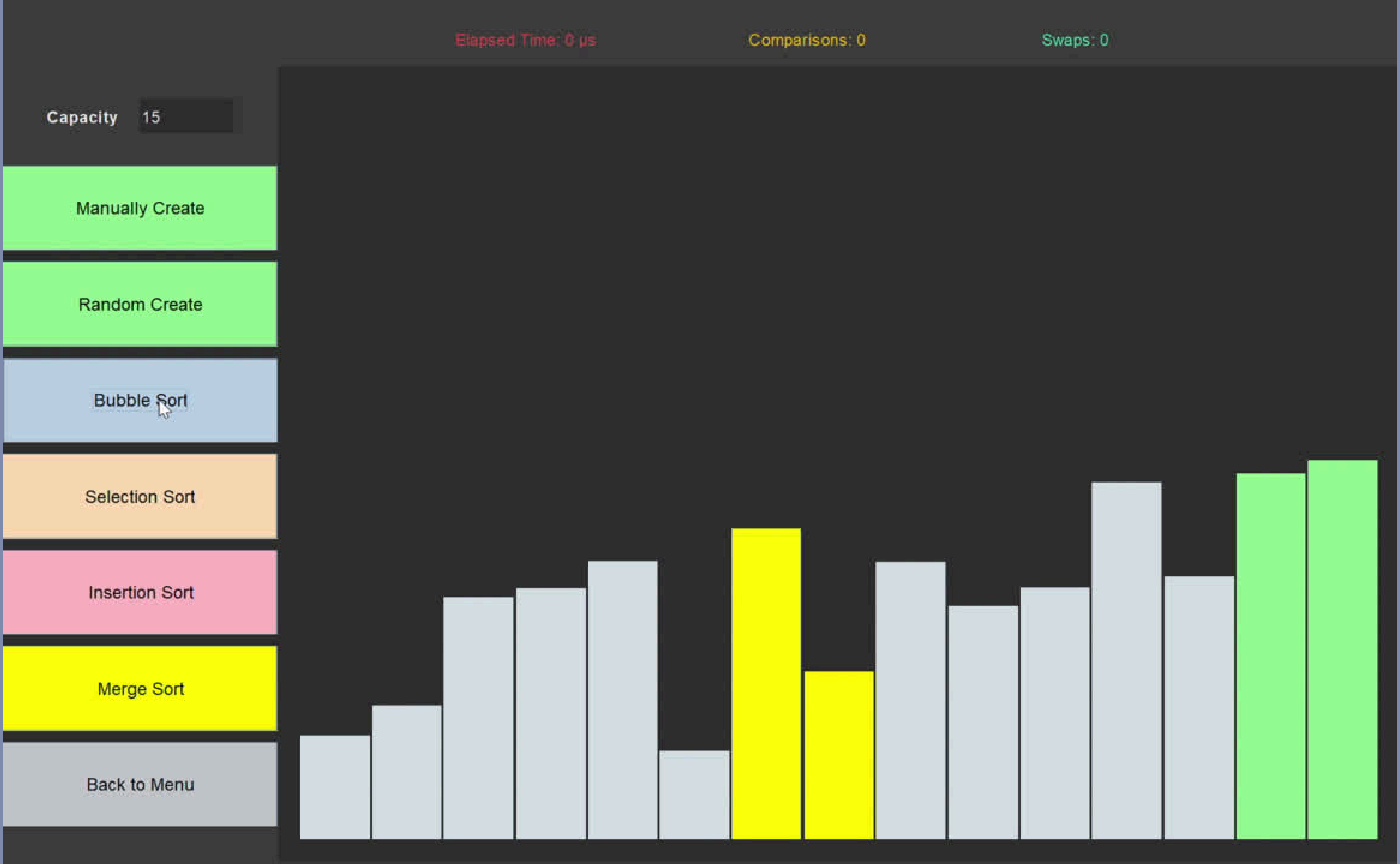
1 2 3 4 5 6 10 12

Sorting algorithm

Merge Sort

- Complexity O(nlogn)
- Merge Sort always divides the array into halves and then merges them, which gives it a logarithmic depth of recursion





Elapsed Time: 0 μs

Comparisons: 0

Swaps: 0

Capacity 15