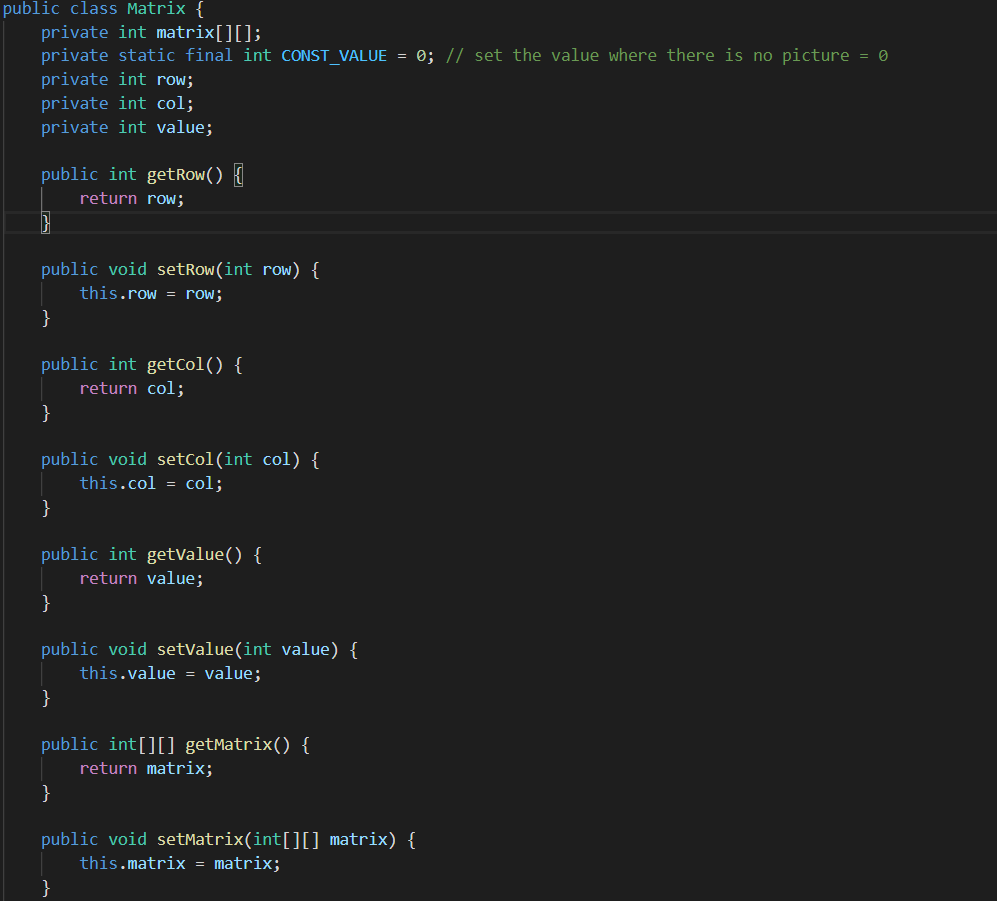
****Kiều Chí Huy – ITDSIU19004 **REPORT OOP PROJECT – Group 18**

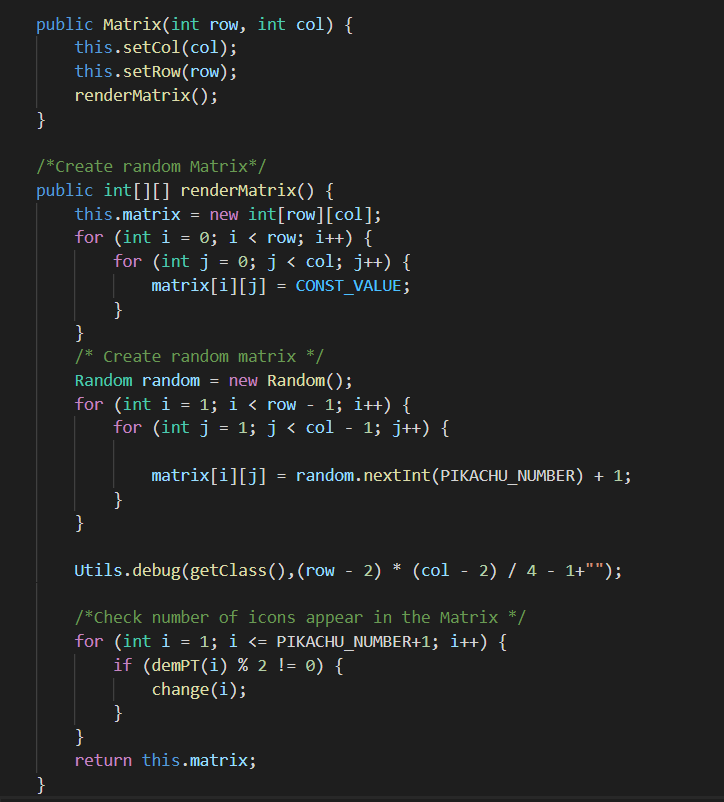
Nguyễn Chí Trung – ITDSIU19024

Nguyễn Đào Trung Hiếu – ITDSIU19035

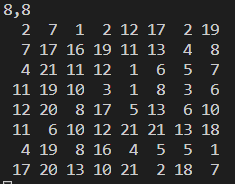
POKEMON – I CHOOSE YOU

1. **Create gameplay**:
2. Create matrix:



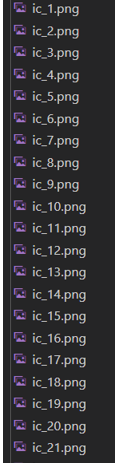


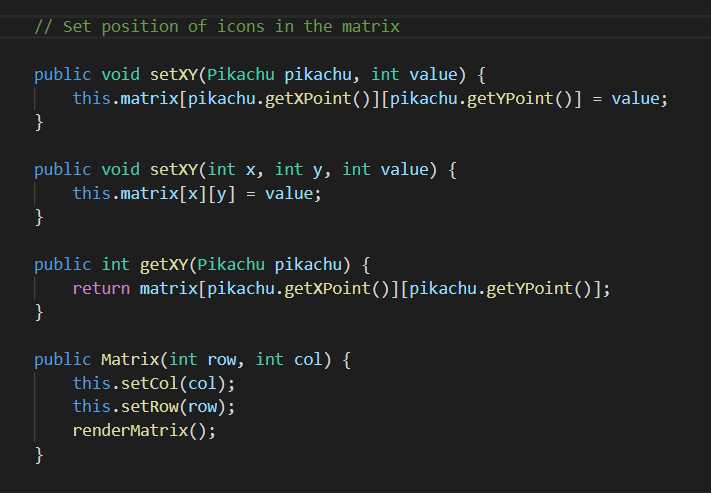
Result:

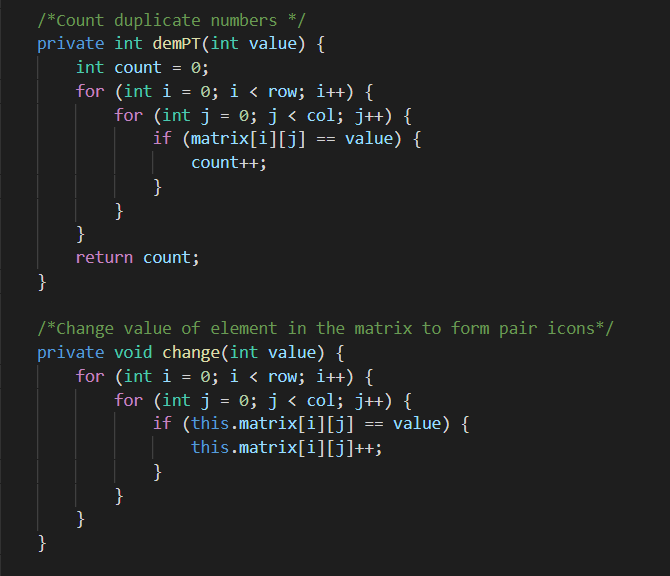


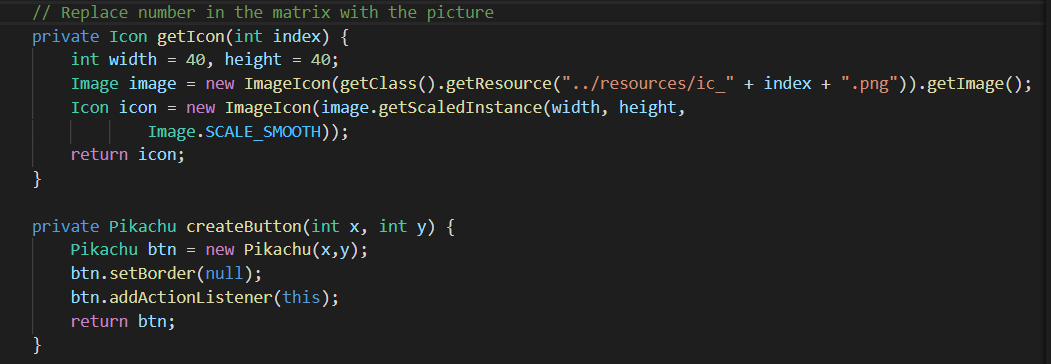
1. Create icons in the matrix:

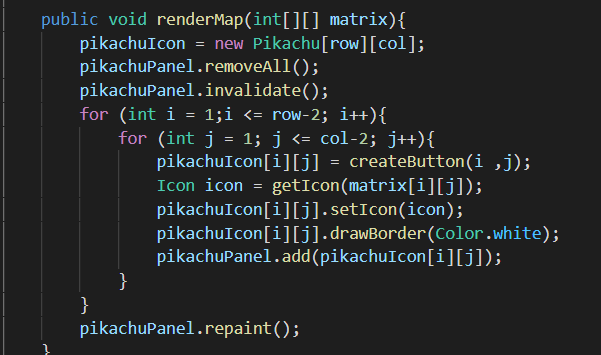
* Create package resource and upload all the picture of the icon in ascending order



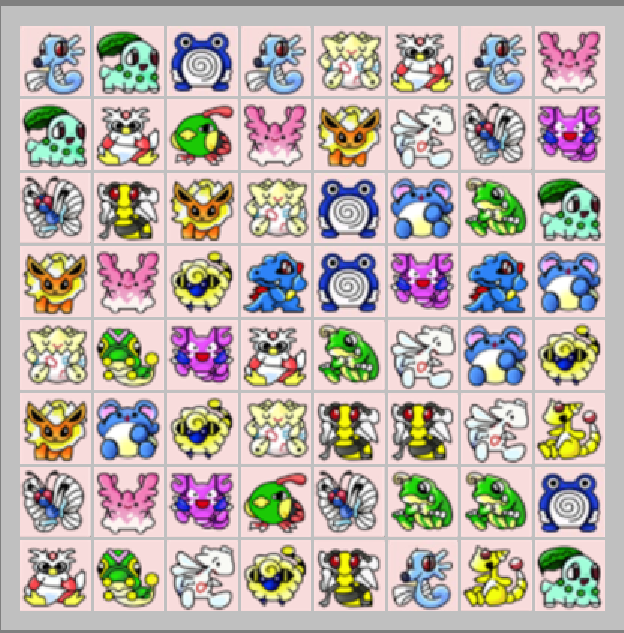








The result:

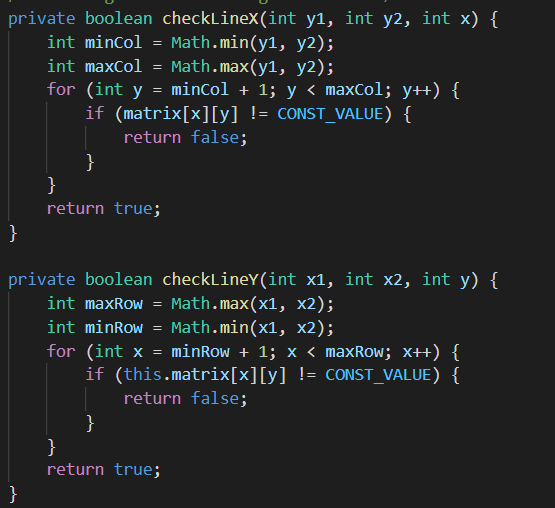


1. Check move:

* Two identical icons will disappear when player click them if they have appropriate move path that does not have any block on that path and less than 3 moves:
* 



* Case 1: Two icons in the same side (same row or column):

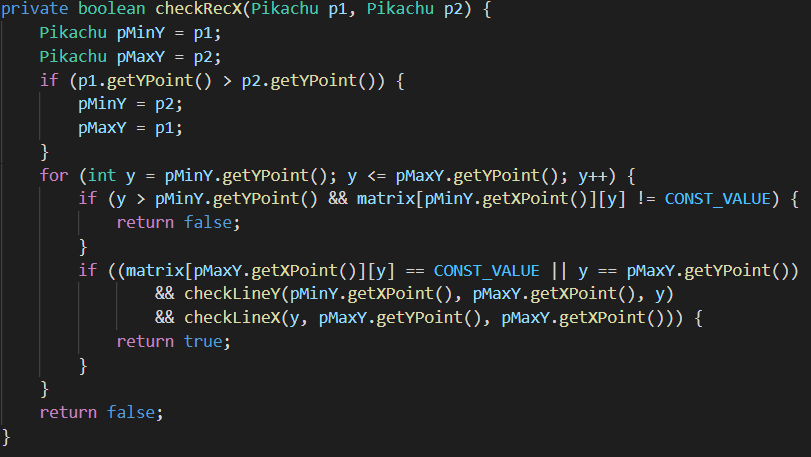


Explain: If two icons player choose are in the same row (x1=x2), check position y1 and y2 to find which is smaller (y1<y2 or y2<y1), then keep checking the other position in that row x1 from y1 to y2 (or y2 to y1 if y2 is smaller) and return true if they all equal 0. (do the same with the column case)



Case 2: Two icons are connected by up to 3 moves within a rectangle formed from the coordinates of the two icons:





Explain: Start with horizontal direction, choose two position y1 and y2 and compare them to find which is smaller in order to increase the position from the smaller to the larger with for loop and value increase =1. With each time increase y + 1, check whether that position have value = 0 or not. If it equal 0, check the last 2 moves that connected these two icons whether they appropriate or not with **checkLineX()** and **checkLineY()** then return true if they all satisfied. (Do the same with vertical direction).

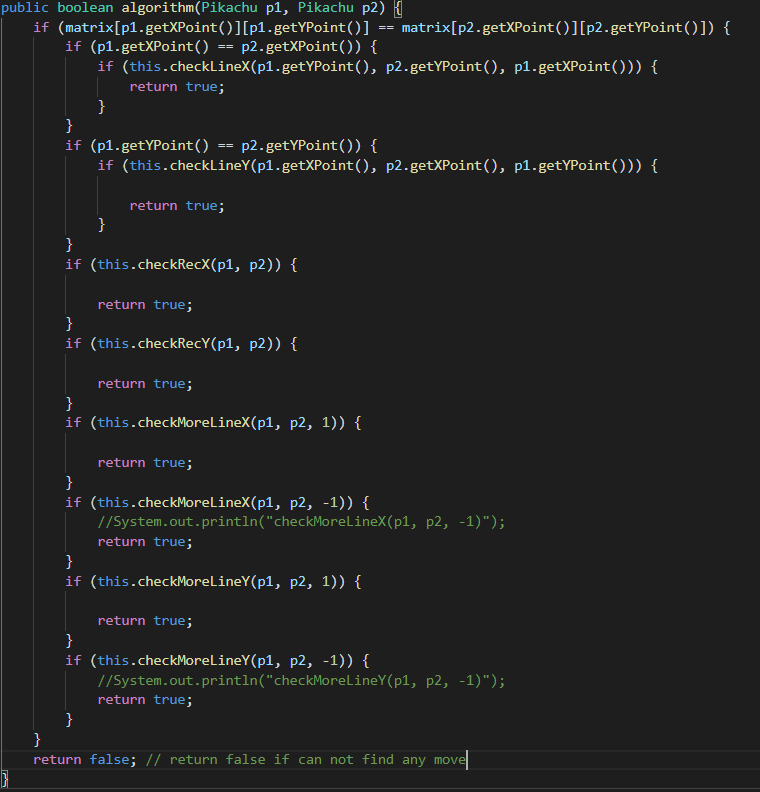
Case 3: Two icons are connected by up to 3 moves outside a rectangle formed from the coordinates of the two icons:





Explain: In general, this case is quiet the same with the case above but we consider the move outside the range of the rectangle that formed from the coordinate of two icons. Start with horizontal direction, create “type” variable with only receive 1 and -1 represent for 2 direction of the move: forward or backward. Compare two icons to find which has y position value smaller (pMinY and pMaxY). With “type” value, if type =1, create y variable = pMaxY.y + type and y = pMinY.y + type if type = -1. In the code above, create two more variable row and colFinish which is the start and finish for checking. After checking for the first move, use while loop to check the last 2 move since there is no boundary in this turn. (Do the same for the vertical direction).

* **FINAL CODE FOR CHECKING MOVE:**

****

1. **Setting background, menu and pause menu**
2. Menu view:

In the menu, there will be 5 button which 5 different uses:

-**Easy**: Start the game tin easy mode.

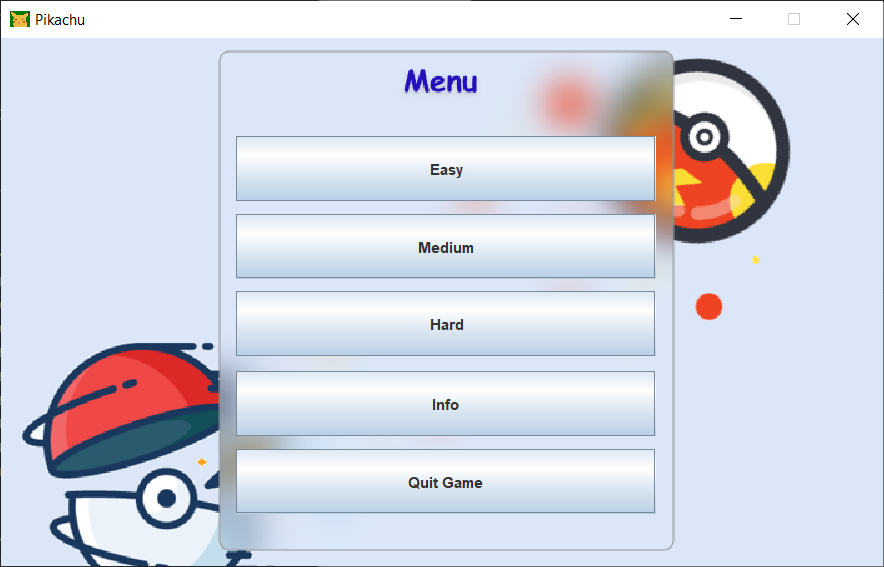
-**Medium**: Start the game in medium mode.

-**Hard**: Start the game in hard mode.

-**Info**: Show the team’s members.

-**Quit**: quit the game.

We set each of the buttons to a specific action-listener to do its separately. This is how the menu output to be:



To visualize each button, we first create a group of buttons and separate it to individualize each of its command. This is a 2d dimension so its need specific setting in horizontal and vertical side:

* + Create a group of layouts that contain the buttons:
* GroupLayout layout = new GroupLayout(this);
* this.setLayout(layout);
  + Horizontal button layer:
* layout.setHorizontalGroup(
* layout.createParallelGroup(GroupLayout.Alignment.LEADING)
* .addGroup(layout.createSequentialGroup()
* .addGap(188, 188, 188)
* .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
* .addComponent(btQuit, GroupLayout.PREFERRED\_SIZE, 336, GroupLayout.PREFERRED\_SIZE)
* .addComponent(btInfo, GroupLayout.PREFERRED\_SIZE, 336, GroupLayout.PREFERRED\_SIZE)
* .addComponent(btHard, GroupLayout.PREFERRED\_SIZE, 336, GroupLayout.PREFERRED\_SIZE)
* .addComponent(btMedium, GroupLayout.PREFERRED\_SIZE, 336, GroupLayout.PREFERRED\_SIZE)
* .addComponent(btEasy, GroupLayout.PREFERRED\_SIZE, 336, GroupLayout.PREFERRED\_SIZE))
* .addContainerGap(196, Short.MAX\_VALUE))
* );
  + Vertical button layer:
* layout.setVerticalGroup(
* layout.createParallelGroup(GroupLayout.Alignment.LEADING)
* .addGroup(GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
* .addContainerGap(77, Short.MAX\_VALUE)
* .addComponent(btEasy, GroupLayout.PREFERRED\_SIZE, 52, GroupLayout.PREFERRED\_SIZE)
* .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
* .addGap(10, 10, 10)
* .addComponent(btMedium, GroupLayout.PREFERRED\_SIZE, 52, GroupLayout.PREFERRED\_SIZE)
* .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
* .addGap(10, 10, 10)
* .addComponent(btHard, GroupLayout.PREFERRED\_SIZE, 52, GroupLayout.PREFERRED\_SIZE)
* .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
* .addComponent(btInfo, GroupLayout.PREFERRED\_SIZE, 52, GroupLayout.PREFERRED\_SIZE)
* .addGap(10, 10, 10)
* .addComponent(btQuit, GroupLayout.PREFERRED\_SIZE, 52, GroupLayout.PREFERRED\_SIZE)
* .addGap(79, 79, 79))
* );

Each button contains a different action event, divide it into cases and put it in “Switch…case” makes it easier to switching actions base on chosen buttons:

@Override

    public void actionPerformed(ActionEvent e) {

        switch (e.getActionCommand()){

            case BT\_EASY: if (onClickMenuListener!=null){

                onClickMenuListener.onNewGameClicked(TYPE\_EASY);

            } break;

            case BT\_MEDIUM: if (onClickMenuListener!=null){

                onClickMenuListener.onNewGameClicked(TYPE\_MEDIUM);

            } break;

            case BT\_HARD: if (onClickMenuListener!=null){

                onClickMenuListener.onNewGameClicked(TYPE\_HARD);

            } break;

            case BT\_INFO: if (onClickMenuListener!=null){

                onClickMenuListener.onInfoClicked();

            } break;

            case BT\_QUIT: if (onClickMenuListener!=null){

                onClickMenuListener.onQuitClicked();

            } break;

            default:

                break;

        }

    }

3 game mode buttons make the same action by start the game, while info shows group members and Quit to turn off the game. Divide the game mode button in to 3 different types on 1 command to make the code more convenience. So with 5 different buttons, we have 3 actions:

public interface OnClickMenuListener{

        void onNewGameClicked(int type);

        void onInfoClicked();

        void onQuitClicked();

    }

1. Pause menu view:

* While playing, players are able to pause the game, it’s necessary to make a fully function pause menu with 3 buttons:
  + Continue: continue playing.
  + Back Menu: back to main menu.
  + Quit Game: Quit the game.
* The same as the menu view, after assign the buttons names, it is necessary to make a layout that contain all the needed components. Then set the positions of the buttons follows the horizontal and vertical sides:
  + Create layout:
* this.setLayout(layout);
  + Vertical:
* layout.setVerticalGroup(
* layout.createParallelGroup(GroupLayout.Alignment.LEADING)
* .addGroup(layout.createSequentialGroup()
* .addGap(100, 100, 100)
* .addComponent(continueGame, GroupLayout.PREFERRED\_SIZE, 70, GroupLayout.PREFERRED\_SIZE)
* .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
* .addGap(10, 10, 10)
* .addComponent(backMenu, GroupLayout.PREFERRED\_SIZE, 70, GroupLayout.PREFERRED\_SIZE)
* .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
* .addGap(10, 10, 10)
* .addComponent(quitGame, GroupLayout.PREFERRED\_SIZE, 70, GroupLayout.PREFERRED\_SIZE)
* .addContainerGap(100, Short.MAX\_VALUE))
* );
  + Horizontal:
* layout.setHorizontalGroup(
* layout.createParallelGroup(GroupLayout.Alignment.LEADING)
* .addGroup(GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
* .addContainerGap(150, Short.MAX\_VALUE)
* .addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING)
* .addComponent(quitGame, GroupLayout.PREFERRED\_SIZE, 353, GroupLayout.PREFERRED\_SIZE)
* .addComponent(backMenu, GroupLayout.PREFERRED\_SIZE, 353, GroupLayout.PREFERRED\_SIZE)
* .addComponent(continueGame, GroupLayout.PREFERRED\_SIZE, 353, GroupLayout.PREFERRED\_SIZE))
* .addGap(180, 180, 180))
* );
* After visualize the buttons, the next step is to set their action, this time its simpler because there’s only 3 buttons in the “Switch…case” code fragment with 3 different actions:
* public interface PauseMenuListener{
* void onContinueCliked();
* void onBackMenuClicked();
* void onQuitClicked();
* }
* @Override
* public void actionPerformed(ActionEvent e) {
* Utils.debug(getClass(),e.getActionCommand());
* if (pauseMenuListener!=null){
* switch (e.getActionCommand()){
* case BT\_CONTINUE : pauseMenuListener.onContinueCliked();  break;
* case BT\_BACK\_MENU : pauseMenuListener.onBackMenuClicked();   break;
* case BT\_QUIT : pauseMenuListener.onQuitClicked();  break;
* default:break;
* }
* }
* }
* And there’s an action -listener for the pause button:

public void setPauseMenuListener(PauseMenuListener pauseMenuListener) {     this.pauseMenuListener = pauseMenuListener;

    }