1)
GameControl Class is created to implement JSlider to control the game difficulty.
Tick spacing set to 1 to show the tick marks in every unit.
JSlider is placed horizontal and set O as the minimum value and 2 as the maximum value. 0 defined as initial value where JSlider tick pointed during game start.
To display the Difficulty levels, I had to create Hashtable and store difficulty levels with key values as JLables.
Then those keys matched with the keys in JSlider levels and placed into JSlider.
Two JButtons created for the Start and Restart button.
Finally, all JLables,JSlider and JButtons placed into the JPannel.

```
public class GameControl extends JFrame implements ActionListener{
JSlider Slider;
  JPanel panel;
  boolean GameStarted=false;
  JButton startButton;

  GameControl(){
    setSize(500,200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("GAME CONTROL");
    buildPanel();
    add(panel);
    setVisible(true);
    panel.setLayout(new FlowLayout());

  }
  public void buildPanel(){
    Slider=new JSlider(JSlider.HORIZONTAL,0,2,0);
    Slider.setMajorTickSpacing(1);
    Slider.setPaintTicks(true);
    Slider.setPaintLabels(true);
    Slider.setSnapToTicks(true);

    Hashtable<Integer, JLabel> SliderLabel = new Hashtable<>();
    SliderLabel.put(0, new JLabel("Easy"));
    SliderLabel.put(1, new JLabel("Medium"));
    SliderLabel.put(2, new JLabel("Hard"));
    Slider.setLabelTable(SliderLabel);

    JLabel lableDifficulty=new JLabel("Difficulty Level");
       startButton=new JButton("Start");
    JButton restartButton=new JButton("Restart");
    startButton.addActionListener(this);
    restartButton.addActionListener(this);

    panel=new JPanel();
    panel.add(lableDifficulty);
    panel.add(Slider);
    panel.add(startButton);
    panel.add(restartButton);
```

Action Listener interface implemented to read user feedback and Actionlistner is added to Start and Restart button to get user feedback.
In the Action Performed method I created if else statement to check which button is pressed. GameStarted flag is created to check whether the start button is previously pressed. If so, start button is StartButton.setEnabled(false); will ensure that it's disabled to pressed again.

If the game is not previously started and Start button is pressed. Then the switch cases check which tick is selected. Depending on the tick Game launch in three different difficulty levels. Below code demonstrate how game play easy start when user click the start button.

SwingUtilities.invokeLater(() -> ensures that the GUI to launches on Event dispatch thread(EDT) execute every GUI inside EDT helped to avoid the game anomalies I was having during game play.
This helps to run the game in single thread.

```
if(e.getActionCommand().equals("Start")&&!GameStarted){
      int value=Slider.getValue();
      switch(value){
        case 0:
            SwingUtilities.invokeLater(() -> {
              JFrame frameEasy = new JFrame("Game Play Easy");

frameEasy.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frameEasy.getContentPane().add(new GamePlayEasy());
                frameEasy.pack();
                frameEasy.setVisible(true);
              GameStarted=true;
              startButton.setEnabled(false);

            });
          break;
```

Once user click restart button similar set of switch statements check the conditions as described above to launch the game in different difficulty levels.

All these difficulty levels, JFrame is created inside the Anonymous inner Class and JPannel is added to that depending on which condition in JSlider is matched.

Three separate classes are created for easy medium and difficult game levels, however the implementation of these classes is same except the number of bubbles drawing depending on the difficulty level. This increases code redundancy and not a best practice in coding. However, this make to implement functions in 'easy' level at first and apply same into the other difficulty levels later depending on the implementation success during the limited time I had . however, I would implement object-oriented principles thoroughly in future versions to ensure to reduce code redundancy and streamline the functionality.

2)
To set the playing field JPannel is set to 700,700 pixels and through the game , JPannel diamention is checked in various situations. Such as,

To draw the Bubble,
To Generate random potions

```
setPreferredSize(new Dimension(700, 700));

………………………..

private boolean CheckFrameInside(int x, int y) {
    int radius = 25;
    return x - radius >= 0 && x + radius <=700&& y - radius >= 0 && y + radius 
<=700;
  }

………………………….
}while((Collison||!CheckFrameInside(NewX,NewY)&&Attemps<MaxAttemps));

    if(!Collison&&CheckFrameInside(NewX,NewY)){
      CircleX[index]=NewX;
      CircleY[index]=NewY;
      repaint();
```

III)

Immediately when the Constructor is called in particular game level, message displays and ask user to select locations to draw the bubbles.

```
JOptionPane.showMessageDialog(null, "select 4 locations to draw the bubbles", 
"Message",
        JOptionPane.INFORMATION_MESSAGE);
```

Then in mouseClicked method, iteration implemented trough the circleCount which is either 4,5 or 6 depending on the game difficulty. In that iteration it checks the conditions where bubble count should be less than the maximum count of that particular level and whether the mouse click is allowing program to draw the bubble inside the JFrame fully. If both conditions true. It takes both X and Y coordinates and save into integer arrays created to store the coordinates. Each iteration, program calls repaint() method to draw each bubble. With having condition to check whether the Round number is equal to 1 ensures this loop is only implements in Round 1.

```
if (circleCount < 4 && CheckFrameInside(x, y) && Round == 1) {
      CircleX[circleCount] = x;
      CircleY[circleCount] = y;
      circleCount++;
…………………………………
      repaint();
```

Bubble is drawn by paintComponent() method using the same logic as assignment 4.additionally the program iterates to draw the number of bubbles depending on the game level.

```
super.paintComponent(g);
    g.setColor(Color.PINK);
    for (int i = 0; i < circleCount; i++) {
      g.fillOval(CircleX[i] - NeighborhoodRadius, CircleY[i] - NeighborhoodRadius,
2 * NeighborhoodRadius,
          2 * NeighborhoodRadius);
```

However, if the user clicked the mouse where the bubble cannot draw inside the JFrame, program promts JOption pane error message and ask user to do another selection. This implemented using if else loop
To draw the bubble if fully contain inside the JFrame or else prompt error message(**using the same logic in assignment 4**)

```
private boolean CheckFrameInside(int x, int y) {
    int radius = 25;
    return x - radius >= 0 && x + radius <=700&& y - radius >= 0 && y + radius
<=700;
  }
```

```
else if(!isAllBubbleDrawn) {
      JOptionPane.showMessageDialog(null, "please select where circle can be drawn
inside the frame","Information",JOptionPane.INFORMATION_MESSAGE);
```

Once number of bubbles for the particular level drawn, Random Hop, Game Timers start which will further discussed in following answers.

From Round2-10 bubbles are randomly drown using following loop inside the mouseClicked method

```
  for(int j=0;j<CircleX.length;j++){
              CircleX[j] = random.nextInt(700- 2 * NeighborhoodRadius) +
NeighborhoodRadius;
            CircleY[j] = random.nextInt(700- 2 * NeighborhoodRadius) +
NeighborhoodRadius;
             }
            circleCount=CircleX.length;
            BouncingBoxCalculator();
            repaint();
```

By using random.nextint method, the program generates random numbers within provided limit which is 700 pixels. By subtracting twice as the NeighborhoodRadius which is 50 ensures bubbles draw within the JFrame.

IV)

Program draws bounding boxes by iterating through number of bubbles depending on the game difficulty. Java Rectangle class is used In order to store the bounding box coordinates. Each and every iteration new object of the class is created to store four coordinates.
to position the bounding box around the bubble, initially, 50 is deducted from both X and Y coordinates.

```
private void BouncingBoxCalculator() {

    for (int i = 0; i < circleCount; i++) {
      BoxArray[i] = new Rectangle(CircleX[i] - (InitialSize + (Round - 1) *
BoxSizeIncrement) / 2,
          CircleY[i] - (InitialSize + (Round - 1) * BoxSizeIncrement) / 2,
InitialSize + (Round - 1) * BoxSizeIncrement,
          InitialSize + (Round - 1) * BoxSizeIncrement);

    `
```

And also in order to increase the neighborhood size depending on the game level bouncingBoxSizeIncreaser() is added. This ensures that the box coordinates changes as the box getting bigger round by round.
Then box height and width also pass using same logic to ensure size is increase by 18 every round.

```
private void BouncingBoxSizeIncreaser() {
    for (int i = 0; i < circleCount; i++) {
      BoxArray[i].setSize(BoxArray[i].width + BoxSizeIncrement, BoxArray[i].height
+ BoxSizeIncrement);
      BoxArray[i].setLocation(BoxArray[i].x - BoxSizeIncrement / 2, BoxArray[i].y -
BoxSizeIncrement / 2);
    }
  }
```

Depending on the saved coordinates, paintComponent method draws the bounding boxes iterating through number of bubbles.

```
for (int i = 0; i < circleCount; i++) {
……………………..
g.setColor(Color.RED);
      Rectangle boundingbox = BoxArray[i];
      g.drawRect(boundingbox.x, boundingbox.y, boundingbox.width,
boundingbox.height);
```

Bubbles makes hop inside the bounding box once all the bubbles are drawn.
In order to achieve this in Round 1 and Round2-10 calls RandomHopStarter() method after the bubbles are drawn and RandomHopStarter () method calls RandomHop() method by passing integer parameter to the method to iterate the random hop through all the bubbles.

RaondomHop() method take integer parameter which is the particular index of the bounding box and depending on the bounding box it generates new coordinates to update the bubble location inside the bounding box.

Maximum X and Y coordinate limit is now set to contain bubble inside the bounding box. Therefore, the mximum X and Y coordinates are calculated based on current position of the BoXArray index. Same logic used to draw the bubbles first place used here as well. However, modified to contain within the box.

V)
Collision avoidance is implemented in RandomHop() method.
MaxX and MaxY variables delaired to save maximum possible coordinate which bubble can hop.

Then Do while loop implemented to find possible coordinates to hop. Maximum attempts are set to 100 as Replit is having limited processing power and was causing program to crash in initial phase by running infinite loop. However, in a PC it can set to much higher limit without an issue.

Random.nextInt method used to generate the random number within the provided limits and Math.min(MaxX,CircleC[index] ensures that the bubble X coordinates stay within bounding box.
Same is implemented to generate Y coordinates as well.

Both possible coordinates are saved into NewX and NewY variables.

```
Rectangle boundingbox = BoxArray[index];
    int MaxX = boundingbox.x +boundingbox.width-2*NeighborhoodRadius;
    int MaxY = boundingbox.y +boundingbox.height-2*NeighborhoodRadius;



…………….


NewX
=Math.max(boundingbox.x+NeighborhoodRadius,Math.min(MaxX,CircleX[index]+random.next
Int(2*NeighborhoodRadius)-NeighborhoodRadius));
      NewY
=Math.max(boundingbox.y+NeighborhoodRadius,Math.min(MaxY,CircleY[index]+random.next
Int(2*NeighborhoodRadius)-NeighborhoodRadius));
```

After that the coordinates go through iteration to check the collision, for loop iterate through bubble array to check whether the bubbles are colliding. Euclidean formula is used to calculate the distance between the points and if the distance between two points is less than 50 the collision set true and loop breaks.

This continues until there is no collision and the bubbles stays in frame and maximum attems are not reach the limit.

```
for(int i=0;i<circleCount;i++){
        if(i!=index&&Math.sqrt(Math.pow(NewX-CircleX[i],2)+Math.pow(NewY-
CircleY[i],2))<=2*NeighborhoodRadius){
          Collison=true;
          break;
………………..

  }while((Collison||!CheckFrameInside(NewX,NewY)&&Attemps<MaxAttemps));
```

If all the conditions are met new coordinates set to CircleX and CircleY as new coordinates and repaint the bubble to visualize movement.

VI)

Check circleInside() method checks whether the mouse is clicked by inside or outside by calculating the distance from the center of the bubble to the point where user clicked second time. This is also modification from the assignment 4.

```
private double CheckCircleInside(int x1, int y1, int x2, int y2) {
    return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
```

This method returns the distance as double number. inside the mouseClicked() method check whether this distance is less than 25 which is the radius of a bubble. If so the DeleteCircle() method is called.

```
for (int i = 0; i < circleCount; i++) {
      double distance = CheckCircleInside(x, y, CircleX[i], CircleY[i]);
      if (distance < NeighborhoodRadius) {// radius 25
        DeleteCirlce(i);
        repaint();
```

In the deleteCircle() method, the for loop iterate through bubble index and bounding box index replacing the current index with the index next to it. Therefore, once the repaint() is called it manipulate bubble deleting by replacing a bubble and bounding box on top of next one in the index.

```
private void DeleteCirlce(int index) {
    for (int i = index; i < circleCount - 1; i++) {
      CircleX[i] = CircleX[i + 1];
      CircleY[i] = CircleY[i + 1];
      BoxArray[i] = BoxArray[i + 1];
    }
    circleCount--;
  }
```

VII)

This implemented inside the mouse Clicked method using if for loop. If the Round is less or equal to maximum rounds the loop gets coordinates within the JPannel and save in Circle X and Y coordinates as described in question number (IV).
Each and every round starts with a JOptionPane message as follows.

```
JOptionPane.showMessageDialog(null, "Next Round " + Round, " Completed",
JOptionPane.INFORMATION_MESSAGE);
```

VIII)

Gave is over if user deletes all the bubbles in Round 10. This implemented in mouse clicked method using if else statement with bubble delete logic. Once the bubble count reaches 0 the else statement prints JOptionPane message to user by congratulation for game completion.

All the logics from Round2-10 is implemented inside if statement in the same loop.

```
OptionPane.showMessageDialog(null, "All Rounds Completed", " Game Over",
JOptionPane.INFORMATION_MESSAGE);
            Window window=SwingUtilities.windowForComponent(GamePlayEasy.this);
            window.dispose();
            RoundsTimer.stop();
```

Once user click on message window is closed using window.dispose() method. And rounds timer also stops at the same time.

If user clicked outside of a bubble game declared over by again implementing if else statement inside the mouse clicked method where it checks whether the mouse is clicked inside or outside the jframe to delete the bubble as previously explained. Window closes once the game is over.

```
else{
      JOptionPane.showMessageDialog(null, "You missed the Bubble! Game
Over!","Error",JOptionPane.ERROR_MESSAGE);
      Window window=SwingUtilities.windowForComponent(GamePlayEasy.this);
      window.dispose();
      RoundsTimer.stop();
```

Game is declared over once time runs out for game level. This implemented inside the time calculation method. Once the timer reaches to 0 and the bubble count is still not reached to 0 same set of code execute to display the message and close the window.

```
if(circleCount>0){
          JOptionPane.showMessageDialog(null, "Time's up! Game Over", "Game Over",
JOptionPane.INFORMATION_MESSAGE);
          Window window=SwingUtilities.windowForComponent(GamePlayEasy.this);
          window.dispose();
```

IX)

Two timers are implemented for the requirements. One for the game ending time and other for the timer to display during game play.

In order to check the timer for each round RoundTimerCalculator() method is implemented.

Timer is set to 15 seconds initially and reduced each round by one second.

Timer starts once the method is called. Also, each and every time when the timer starts, It checks and stop the timer if there is a previous time is still running.

```
  private void RoundTimerCalculator(){
    if(RoundsTimer!=null){
      RoundsTimer.stop();

    }
    int TimerReduce=15000-(Round-1)*1000;
    RoundsTimer=new Timer(TimerReduce,new ActionListener(){
//once the timer is 0 game window closes here(question VIII)
        }
      }
    });
    RoundsTimer.setRepeats(false);
    RoundsTimer.start();
    System.out.println("Timer started");
}
```

Second timer method implemented to display the time to user.

Same logic is applying as the previous timer however there are minor modifications to accommodate the needs. JLable is created inside the Jpannel and the label text is updating every second to display the time remaining. Inside the Anonymous inner class, the logic is implemented to update the JLabel time and repaint() once every iteration. Once timer riches 0, it stop to avoid displaying negative values.

```
private void GameTimeCalculator(){
    if(GameTimer!=null){
      GameTimer.stop();
    }
    int TimerReduce=15000-(Round-1)*1000;
    Time=TimerReduce/1000;
    TimerLable.setText("Time: "+Time+" Seconds");
    GameTimer=new Timer(1000,new ActionListener(){
      public void actionPerformed(ActionEvent e){
        Time--;
        TimerLable.setText("Time: "+Time+" Seconds");
        repaint();
        if(Time<=0){
          GameTimer.stop();
        }
      }

    });
    GameTimer.setRepeats(true);
    GameTimer.start();
```

Future Modifications and plans:

- There are sometimes that the game suddenly freezes on Replit as it having limited processing power .the reason is identified as when the bubbles are drawing first time automatically it should check the distances to avoid collision during initial drawing.
- Add more modifications to game appearance.
- Add basic sounds to make game more interesting.
- modify the code to implements proper OOP principles in future to avoid code redundancy and improve performance.
- upload the source code to Github in future to add Github link of the Completed Projects during Master Program to CV.

References:-

https://coderanch.com/t/645861/java/Detecting-mouse-click-box-painted

https://www.geeksforgeeks.org/java-util-random-nextint-java/

https://stackoverflow.com/questions/1234912/how-to-programmatically-close-a-jframe

https://docs.oracle.com/javase/8/docs/api/java/util/Timer.html

https://stackoverflow.com/questions/22366890/java-timer-action-listener

https://docs.oracle.com/javase/8/docs/api/javax/swing/Timer.html

https://docs.oracle.com/javase/8/docs/api/java/awt/Rectangle.html

https://stackoverflow.com/questions/61465380/mouselistener-to-instance-of-game-panel

https://www.geeksforgeeks.org/java-math-max-method-examples/

https://stackoverflow.com/questions/27009069/java-timers-and-actionlisteners

https://coderanch.com/t/444203/java/Line-delete-jpanel-mouse-clicked

https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI_2.html

https://www.tutorialspoint.com/how-can-we-call-the-invokelater-method-in-java

https://stackoverflow.com/questions/38425710/getwidth-getheight-in-java

https://www.geeksforgeeks.org/java-sqrt-method-examples/

https://coderanch.com/t/343522/java/Moving-resizing-rectangle

https://alvinalexander.com/java/java-swingutilities-invoke-later-example-edt/

https://docs.oracle.com/javase/8/docs/api/javax/swing/JSlider.html

https://www.baeldung.com/java-distance-between-two-points

Past Lecture Videos.

• https://stackoverflow.com/questions/10556369/mouselistener-in-canvas-not-working

• https://www.cuemath.com/euclidean-distance-formula/

• https://study.com/skill/learn/determining-if-a-point-lies-inside-outside-or-on-a-circle-given-the-center-point-a-radiusexplanation.html#:~:text=We%20will%20use%20the%20distance,lies%20outside%20of%20t
he%20circle.

• https://stackoverflow.com/questions/481144/equation-for-testing-if-a-point-is-inside-a-circle

• https://docs.oracle.com/javase/8/javafx/api/javafx/scene/input/Mnemonic.html

• https://www.geeksforgeeks.org/convert-double-to-integer-in-java/

https://stackoverflow.com/questions/71681960/how-to-control-how-many-times-something-is-shown-in-processing

https://www.geeksforgeeks.org/java-math-max-method-examples/

https://stackoverflow.com/questions/41414716/how-to-re-draw-in-swing-without-removing-previously-drawn-things