

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.imageio.*;
7  import javax.swing.*;
8
9  public class Background {
10     private Image background;
11     private ArrayList<Shrub> shrubs;
12     private ArrayList<BufferedImage> images;
13
14     public Background() {
15         try {
16             background = ImageIO.read(new File("background.png"));
17             shrubs = new ArrayList<>();
18             images = new ArrayList<>();
19             images.add(ImageIO.read(new File("tree.png")));
20             images.add(ImageIO.read(new File("shrub.png")));
21             shrubs.add(new Shrub(1000, 400 - images.get(0).getHeight(), images.get(0)));
22             //images.add(ImageIO.read(new File("")));
23         }
24         catch(IOException e) {
25             System.out.println("io");
26         }
27     }
28
29     public void scroll(double v) {
30         for(Shrub s : shrubs) {
31             s.scroll();
32             s.updateV(v);
33         }
34         if(shrubs.get(0).getXCoord() <= -70) {
35             BufferedImage img = images.get((int)(Math.random() * images.size()));
36             shrubs.add(new Shrub(1000, 400 - img.getHeight(), img));
37             shrubs.remove(0);
38         }
39     }
40
41     public void draw(Graphics g) {
42         g.drawImage(background, 0, 0, null);
43         for(Shrub s : shrubs) {
44             s.draw(g);
45         }
46     }
47
48 }
```

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.swing.*;
7
8  public class Floor {
9
10     private int xCoord;
11     private int yCoord;
12
13     public Floor(int x, int y) {
14         xCoord = x;
15         yCoord = y;
16     }
17     public void draw(Graphics g) {
18         g.setColor(Color.black);
19         g.fillRect(xCoord, yCoord, 1000, 500);
20     }
21 }
```

PDF document made with CodePrint using [Prism](#)

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.imageio.*;
7  import javax.swing.*;
8
9  public class Highscore extends Score {
10     private BufferedImage img;
11     private Image imageNumber;
12     private int highscore;
13
14     public Highscore(int x) {
15         highscore = x;
16         try {
17             img = ImageIO.read(new File("highscore.png"));
18         }
19         catch(IOException e) {}
20     }
21
22     public void draw(Graphics g) {
23         Integer inte = new Integer(highscore);
24         String num = inte.toString();
25         char c;
26         int x = 0;
27         try {
28             for(int i = 0; i < num.length(); i++) {
29                 c = num.charAt(i);
30                 imageNumber = changeNumber(c);
31                 x = 1000 - ((num.length() - i) * 50);
32                 g.drawImage(imageNumber, x, 0, null);
33             }
34         }
35         catch(IOException e) {}
36         g.drawImage(img, x - img.getWidth() - (50 * (num.length() - 1)), 0, null);
37     }
38
39 }
```

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.imageio.*;
7  import javax.swing.*;
8
9  public class LoseScreen {
10     private Image youLose;
11     private Image playAgain;
12     private Image screwThisGame;
13     private Image dot;
14     private int doty = 200;
15
16     public LoseScreen() {
17         try {
18             youLose = ImageIO.read(new File("you lose.png"));
19             playAgain = ImageIO.read(new File("play again.png"));
20             screwThisGame = ImageIO.read(new File("screw this game.png"));
21             dot = ImageIO.read(new File("dot.png"));
22         }
23         catch(IOException e) {
24
25         }
26     }
27
28     public void dotDown() {
29         doty = 250;
30     }
31
32     public void dotUp() {
33         doty = 200;
34     }
35
36     public int getDot() {
37         return doty;
38     }
39
40     public void draw(Graphics g) {
41         g.drawImage(dot, 435, doty, null);
42         g.drawImage(youLose, 400, 100, null);
43         g.drawImage(playAgain, 450, 200, null);
44         g.drawImage(screwThisGame, 450, 250, null);
45     }
46 }
```

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.imageio.*;
7  import javax.swing.*;
8
9  public class Obstacle {
10     private static int enemySprite;
11
12     private int xCoord;
13     private int yCoord;
14     private double velocity;
15     private int width;
16     private int height;
17
18     private BufferedImage img;
19
20     public Obstacle(int x, int y, double v) {
21         velocity = v;
22         xCoord = x;
23         yCoord = y;
24         try {
25             enemySprite = (int)(Math.random() * 5);
26             switch(enemySprite) {
27                 case 0:
28                     img = ImageIO.read(new File("angryface.png"));
29                     break;
30                 case 1:
31                     img = ImageIO.read(new File("apple.png"));
32                     break;
33                 case 2:
34                     img = ImageIO.read(new File("toxic.png"));
35                     break;
36                 case 3:
37                     img = ImageIO.read(new File("star.png"));
38                     break;
39                 case 4:
40                     img = ImageIO.read(new File("recyclebin.png"));
41                     break;
42             }
43         }
44         catch(IOException e) {}
45         width = img.getWidth();
46         height = img.getHeight();
47     }
48
49     public int getXCoord() {
50         return xCoord;
51     }
52
53     public int getYCoord() {
```

```
54     return yCoord;
55 }
56
57 public int getWidth() {
58     return width;
59 }
60
61 public int getHeight() {
62     return height;
63 }
64
65 public void scroll() {
66     xCoord -= velocity;
67 }
68
69 public double getV() {
70     return velocity;
71 }
72
73 public void updateV(double x) {
74     velocity = x;
75 }
76
77 public void draw(Graphics g) {
78     g.drawImage(img, xCoord, yCoord, null);
79 }
80
81 public void drawDeath(Graphics g) {
82     try {
83         switch(enemySprite) {
84             case 0:
85                 img = ImageIO.read(new File("angryangryface.png"));
86                 break;
87             case 1:
88                 img = ImageIO.read(new File("angryapple.png"));
89                 break;
90             case 2:
91                 img = ImageIO.read(new File("angrytoxic.png"));
92                 break;
93             case 3:
94                 img = ImageIO.read(new File("angrystar.png"));
95                 break;
96             case 4:
97                 img = ImageIO.read(new File("angryrecyclebin.png"));
98                 break;
99         }
100         draw(g);
101     }
102     catch(IOException e) {}
103     width = img.getWidth();
104     height = img.getHeight();
105 }
106
107 public String toString() {
108     return "Obstacle-> X: " + xCoord + "\tY: " + yCoord;
109 }
```

```
110 |    }  
    }
```

PDF document made with CodePrint using [Prism](#)

```
1  import javax.swing.*;
2  import java.awt.image.*;
3  import java.awt.*;
4  import java.io.*;
5  import javax.imageio.*;
6  public class Player {
7      private BufferedImage img;
8      private int xCoord;
9      private int yCoord;
10     private int width;
11     private int height;
12     private int sprite;
13     private double velocity;
14
15     public Player(int x, int y) {
16         xCoord = x;
17         yCoord = y;
18         try {
19             img = ImageIO.read(new File("cat.png"));
20         }
21         catch(IOException e) {
22             System.out.println("Failed to load image");
23         }
24         width = img.getWidth();
25         height = img.getHeight();
26     }
27     public void jump() {
28         if(yCoord == 300)
29             velocity = -10;
30     }
31     public void gravity() {
32         sprite++;
33         velocity += 0.378;
34         yCoord += (velocity + 0.0072);
35         if(yCoord > 300) {
36             yCoord = 300;
37         }
38     }
39     public boolean collision(int x, int y, int w, int h) {
40         boolean left = x > xCoord;
41         boolean right = x < xCoord + width;
42         boolean up = y < yCoord + height;
43         boolean down = y + h > yCoord;
44         return left && right && down && up;
45     }
46     public int getXCoord() {
47         return xCoord;
48     }
49     public int getYCoord() {
50         return yCoord;
51     }
52     public void draw(Graphics g) {
53         try {
```



```
54         if(Stage.lose)
55             img = ImageIO.read(new File("deadcat.png"));
56         else if(sprite % 3 == 0)
57             img = ImageIO.read(new File("cat.png"));
58         else
59             img = ImageIO.read(new File("cat2.png"));
60     }
61     catch(IOException e) {}
62     g.drawImage(img, xCoord, yCoord, null);
63 }
64 public String toString() {
65     return "Player-> X: " + xCoord + "\tY: " + yCoord;
66 }
67 }
```

PDF document made with CodePrint using [Prism](#)

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.imageio.*;
7  import javax.swing.*;
8
9  public class Score {
10     private Image score;
11     private Image imageNumber;
12
13     public Score() {
14         try {
15             score = ImageIO.read(new File("score.png"));
16         }
17         catch(IOException e) {}
18     }
19
20     public Image changeNumber(char c) throws IOException {
21         Image number = null;
22         switch(c) {
23             case '0':
24                 number = ImageIO.read(new File("zero.png"));
25                 break;
26             case '1':
27                 number = ImageIO.read(new File("one.png"));
28                 break;
29             case '2':
30                 number = ImageIO.read(new File("two.png"));
31                 break;
32             case '3':
33                 number = ImageIO.read(new File("three.png"));
34                 break;
35             case '4':
36                 number = ImageIO.read(new File("four.png"));
37                 break;
38             case '5':
39                 number = ImageIO.read(new File("five.png"));
40                 break;
41             case '6':
42                 number = ImageIO.read(new File("six.png"));
43                 break;
44             case '7':
45                 number = ImageIO.read(new File("seven.png"));
46                 break;
47             case '8':
48                 number = ImageIO.read(new File("eight.png"));
49                 break;
50             case '9':
51                 number = ImageIO.read(new File("nine.png"));
52                 break;
53         }
```

```
54     return number;
55 }
56
57 public void draw(Graphics g, int s) {
58     g.drawImage(score, 0, 0, null);
59     Integer inte = new Integer(s);
60     String num = inte.toString();
61     char c;
62     try {
63         for(int i = 0; i < num.length(); i++) {
64             c = num.charAt(i);
65             imageNumber = changeNumber(c);
66             g.drawImage(imageNumber, i * 50 + 75, 0, null);
67         }
68     }
69     catch(IOException e) {}
70 }
71 }
```

PDF document made with CodePrint using [Prism](#)

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.awt.image.*;
4  import java.io.*;
5  import java.util.*;
6  import javax.imageio.*;
7  import javax.swing.*;
8
9  public class Shrub {
10
11     private BufferedImage img;
12     private int xCoord;
13     private int yCoord;
14     private int width;
15     private int height;
16     private double velocity;
17
18     public Shrub(int x, int y, BufferedImage i) {
19         img = i;
20         xCoord = x;
21         yCoord = y;
22         width = i.getWidth();
23         height = i.getHeight();
24     }
25
26     public int getXCoord() {
27         return xCoord;
28     }
29
30     public int getYCoord() {
31         return yCoord;
32     }
33
34     public int getWidth() {
35         return width;
36     }
37
38     public int getHeight() {
39         return height;
40     }
41
42     public void scroll() {
43         xCoord -= velocity;
44     }
45
46     public void updateV(double v) {
47         velocity = v;
48     }
49
50     public void draw(Graphics g) {
51         g.drawImage(img, xCoord, yCoord, null);
52     }
53 }
```

PDF document made with CodePrint using [Prism](#)

```
1 //imports
2 import java.awt.*;
3 import java.awt.event.*;
4 import java.awt.image.*;
5 import java.io.*;
6 import java.util.*;
7 import javax.swing.*;
8
9 public class Stage extends JPanel implements KeyListener, ActionListener {
10
11     //stage height and width
12     private static final int STAGE_HEIGHT = 500;
13     private static final int STAGE_WIDTH = 1000;
14
15     //private
16     private File dat;
17     private javax.swing.Timer timer;
18     private Player player;
19     private Floor floor;
20     private ArrayList<Obstacle> obstacles;
21     private LoseScreen losescreen;
22     private Score scoreboard;
23     private Highscore highscore;
24     private int score;
25     private int high_score;
26     private Background background;
27     private double velocity;
28
29     //lose varriable
30     public static boolean lose;
31
32
33     public Stage(int width, int height) {
34         setPreferredSize(new Dimension(width, height));
35         player = new Player(50, 310);
36         floor = new Floor(0, STAGE_HEIGHT - 100);
37         timer = new javax.swing.Timer(20, this);
38         losescreen = new LoseScreen();
39         scoreboard = new Score();
40         initializeDataFile();
41         highscore = new Highscore(high_score);
42         background = new Background();
43         obstacles = new ArrayList<>();
44         obstacles.add(new Obstacle(800, 290, 7));
45         this.setFocusable(true);
46         addKeyListener(this);
47         lose = false;
48         timer.start();
49     }
50
51     public void initializeDataFile() {
52         try {
53             dat = new File("info.dat");
```

```
54         Scanner scan = new Scanner(dat);
55         high_score = Integer.valueOf(scan.nextLine());
56     }
57     catch(IOException e) {}
58 }
59
60 public void closeDataFile() {
61     try {
62         PrintWriter pw = new PrintWriter(dat);
63         pw.println(score);
64         pw.close();
65     }
66     catch(IOException e) {}
67 }
68
69 public void actionPerformed(ActionEvent e) {
70     player.gravity();
71     status();
72     if(!lose) {
73         for(Obstacle o : obstacles) {
74             o.scroll();
75             velocity = o.getV() + (score * 0.01);
76             o.updateV(velocity);
77         }
78         background.scroll(velocity);
79     }
80     else {
81         timer.stop();
82     }
83     repaint();
84 }
85
86 public void keyPressed(KeyEvent e) {
87     if(lose) {
88         switch(e.getKeyCode()) {
89             case KeyEvent.VK_UP:
90                 losescreen.dotUp();
91                 repaint();
92                 break;
93             case KeyEvent.VK_DOWN:
94                 losescreen.dotDown();
95                 repaint();
96                 break;
97             case KeyEvent.VK_ENTER:
98                 if(losescreen.getDot() - 200 == 0) {
99                     start();
100                     lose = false;
101                 }
102                 else {
103                     System.exit(0);
104                 }
105                 break;
106         }
107     }
108     else if(e.getKeyCode() == KeyEvent.VK_SPACE) {
109         player.jump();
110     }
```

```
110     }
111 }
112
113 public void keyTyped(KeyEvent e) {}
114 public void keyReleased(KeyEvent e) {}
115
116 public void paintComponent(Graphics g) {
117     super.paintComponent(g);
118
119     background.draw(g);
120     scoreboard.draw(g, score);
121     highscore.draw(g);
122     player.draw(g);
123     floor.draw(g);
124     for(Obstacle o : obstacles) {
125         o.draw(g);
126     }
127     if(lose) {
128         for(Obstacle o : obstacles) {
129             o.drawDeath(g);
130         }
131         losescreen.draw(g);
132     }
133 }
134
135 public void status() {
136     for(Obstacle o : obstacles) {
137         //System.out.print(player + "\t" + o + "\r");
138         if(player.collission(o.getXCoord(), o.getYCoord(), o.getWidth(), o.getHeight())) {
139             lose = true;
140             if(score > high_score) {
141                 closeDataFile();
142             }
143             repaint();
144         }
145     }
146
147     if(obstacles.get(0).getXCoord() <= -70) {
148         int y = ((int)(Math.random() * 10) % 2) * 100 + 200;
149         obstacles.add(new Obstacle(1000, y, 7));
150         obstacles.remove(0);
151         score++;
152     }
153 }
154
155 public void start() {
156     score = 0;
157     player = new Player(50, 310);
158     floor = new Floor(0, STAGE_HEIGHT - 100);
159     timer = new javax.swing.Timer(20, this);
160     losescreen = new LoseScreen();
161     scoreboard = new Score();
162     initializeDataFile();
163     highscore = new Highscore(high_score);
164     obstacles = new ArrayList<>();
165     obstacles.add(new Obstacle(800, 290, 7));
```



```
166         this.setFocusable(true);
167         addKeyListener(this);
168         lose = false;
169         timer.start();
170     }
171
172     public static void main(String args[]) {
173         JFrame frame = new JFrame();
174         frame.setTitle("Cat run");
175         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
176         JPanel panel = new Stage(STAGE_WIDTH, STAGE_HEIGHT);
177         Container c = frame.getContentPane();
178         c.add(panel);
179         frame.pack();
180         frame.setVisible(true);
181         frame.setResizable(false);
182     }
183 }
```

PDF document made with CodePrint using [Prism](https://bakerfranke.github.io/codePrint/)