```java
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

public class Background {
    private Image background;
    private ArrayList<Shrub> shrubs;
    private ArrayList<BufferedImage> images;

    public Background() {
        try {
            background = ImageIO.read(new File("background.png"));
            shrubs = new ArrayList<>();
            images = new ArrayList<>();
            images.add(ImageIO.read(new File("tree.png")));
            images.add(ImageIO.read(new File("shrub.png")));
            shrubs.add(new Shrub(1000, 400 - images.get(0).getHeight(), images.get(0)));
            //images.add(ImageIO.read(new File("")));
        }
        catch(IOException e) {
            System.out.println("io");
        }
    }

    public void scroll(double v) {
        for(Shrub s : shrubs) {
            s.scroll();
            s.updateV(v);
        }
        if(shrubs.get(0).getXCoord() <= -70) {
            BufferedImage img = images.get((int)(Math.random() * images.size()));
            shrubs.add(new Shrub(1000, 400 - img.getHeight(), img));
            shrubs.remove(0);
        }
    }

    public void draw(Graphics g) {
        g.drawImage(background, 0, 0, null);
        for(Shrub s : shrubs) {
            s.draw(g);
        }
    }

}
```

PDF document made with CodePrint using Prism

```java
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.swing.*;

public class Floor {

    private int xCoord;
    private int yCoord;

    public Floor(int x, int y) {
        xCoord = x;
        yCoord = y;
    }
    public void draw(Graphics g) {
        g.setColor(Color.black);
        g.fillRect(xCoord, yCoord, 1000, 500);
    }
}
```

PDF document made with CodePrint using Prism

```java
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

public class Highscore extends Score {
    private BufferedImage img;
    private Image imageNumber;
    private int highscore;

    public Highscore(int x) {
        highscore = x;
        try {
            img = ImageIO.read(new File("highscore.png"));
        }
        catch(IOException e) {}
    }

    public void draw(Graphics g) {
        Integer inte = new Integer(highscore);
        String num = inte.toString();
        char c;
        int x = 0;
        try {
            for(int i = 0; i < num.length(); i++) {
                c = num.charAt(i);
                imageNumber = changeNumber(c);
                x = 1000 - ((num.length() - i) * 50);
                g.drawImage(imageNumber, x, 0, null);
            }
        }
        catch(IOException e) {}
        g.drawImage(img, x - img.getWidth() - (50 * (num.length() - 1)), 0, null);
    }

}
```

PDF document made with CodePrint using Prism

```java
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

public class LoseScreen {
    private Image youLose;
    private Image playAgain;
    private Image screwThisGame;
    private Image dot;
    private int doty = 200;

    public LoseScreen() {
        try {
            youLose = ImageIO.read(new File("you lose.png"));
            playAgain = ImageIO.read(new File("play again.png"));
            screwThisGame = ImageIO.read(new File("screw this game.png"));
            dot = ImageIO.read(new File("dot.png"));
        }
        catch(IOException e) {

        }
    }

    public void dotDown() {
        doty = 250;
    }

    public void dotUp() {
        doty = 200;
    }

    public int getDot() {
        return doty;
    }

    public void draw(Graphics g) {
        g.drawImage(dot, 435, doty, null);
        g.drawImage(youLose, 400, 100, null);
        g.drawImage(playAgain, 450, 200, null);
        g.drawImage(screwThisGame, 450, 250, null);
    }
}
```

PDF document made with CodePrint using Prism

```java
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

public class Obstacle {
    private static int enemySprite;

    private int xCoord;
    private int yCoord;
    private double velocity;
    private int width;
    private int height;

    private BufferedImage img;

    public Obstacle(int x, int y, double v) {
        velocity = v;
        xCoord = x;
        yCoord = y;
        try {
            enemySprite = (int)(Math.random() * 5);
            switch(enemySprite) {
                case 0:
                    img = ImageIO.read(new File("angryface.png"));
                    break;
                case 1:
                    img = ImageIO.read(new File("apple.png"));
                    break;
                case 2:
                    img = ImageIO.read(new File("toxic.png"));
                    break;
                case 3:
                    img = ImageIO.read(new File("star.png"));
                    break;
                case 4:
                    img = ImageIO.read(new File("recyclebin.png"));
                    break;
            }
        }
        catch(IOException e) {}
        width = img.getWidth();
        height = img.getHeight();
    }

    public int getXCoord() {
        return xCoord;
    }

    public int getYCoord() {
        return yCoord;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

    public void scroll() {
        xCoord -= velocity;
    }

    public double getV() {
        return velocity;
    }

    public void updateV(double x) {
        velocity = x;
    }

    public void draw(Graphics g) {
        g.drawImage(img, xCoord, yCoord, null);
    }

    public void drawDeath(Graphics g) {
        try {
            switch(enemySprite) {
                case 0:
                    img = ImageIO.read(new File("angryangryface.png"));
                    break;
                case 1:
                    img = ImageIO.read(new File("angryapple.png"));
                    break;
                case 2:
                    img = ImageIO.read(new File("angrytoxic.png"));
                    break;
                case 3:
                    img = ImageIO.read(new File("angrystar.png"));
                    break;
                case 4:
                    img = ImageIO.read(new File("angryrecyclebin.png"));
                    break;
            }
            draw(g);
        }
        catch(IOException e) {}
        width = img.getWidth();
        height = img.getHeight();
    }

    public String toString() {
        return "Obstacle-> X: " + xCoord + "\tY: " + yCoord;

    }
}
```

PDF document made with CodePrint using Prism

```
import javax.swing.*;
import java.awt.image.*;
import java.awt.*;
import java.io.*;
import javax.imageio.*;
public class Player {
    private BufferedImage img;
    private int xCoord;
    private int yCoord;
    private int width;
    private int height;
    private int sprite;
    private double velocity;

    public Player(int x, int y) {
        xCoord = x;
        yCoord = y;
        try {
            img = ImageIO.read(new File("cat.png"));
        }
        catch(IOException e) {
            System.out.println("Failed to load image");
        }
        width = img.getWidth();
        height = img.getHeight();
    }
    public void jump() {
        if(yCoord == 300)
            velocity = -10;
    }
    public void gravity() {
        sprite++;
        velocity += 0.378;
        yCoord += (velocity + 0.0072);
        if(yCoord > 300) {
            yCoord = 300;
        }
    }
    public boolean collision(int x, int y, int w, int h) {
        boolean left = x > xCoord;
        boolean right = x < xCoord + width;
        boolean up = y < yCoord + height;
        boolean down = y + h > yCoord;
        return left && right && down && up;
    }
    public int getXCoord() {
        return xCoord;
    }
    public int getYCoord() {
        return yCoord;
    }
    public void draw(Graphics g) {
        try {
            if(Stage.lose)
                img = ImageIO.read(new File("deadcat.png"));
            else if(sprite % 3 == 0)
                img = ImageIO.read(new File("cat.png"));
            else
                img = ImageIO.read(new File("cat2.png"));
        }
        catch(IOException e) {}
        g.drawImage(img, xCoord, yCoord, null);
    }
    public String toString() {
        return "Player-> X: " + xCoord + "\tY: " + yCoord;
    }
}
```

PDF document made with CodePrint using Prism

```java
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

public class Score {
    private Image score;
    private Image imageNumber;

    public Score() {
        try {
            score = ImageIO.read(new File("score.png"));
        }
        catch(IOException e) {}
    }

    public Image changeNumber(char c) throws IOException {
        Image number = null;
        switch(c) {
            case '0':
                number = ImageIO.read(new File("zero.png"));
                break;
            case '1':
                number = ImageIO.read(new File("one.png"));
                break;
            case '2':
                number = ImageIO.read(new File("two.png"));
                break;
            case '3':
                number = ImageIO.read(new File("three.png"));
                break;
            case '4':
                number = ImageIO.read(new File("four.png"));
                break;
            case '5':
                number = ImageIO.read(new File("five.png"));
                break;
            case '6':
                number = ImageIO.read(new File("six.png"));
                break;
            case '7':
                number = ImageIO.read(new File("seven.png"));
                break;
            case '8':
                number = ImageIO.read(new File("eight.png"));
                break;
            case '9':
                number = ImageIO.read(new File("nine.png"));
                break;
        }
        return number;
    }

    public void draw(Graphics g, int s) {
        g.drawImage(score, 0, 0, null);
        Integer inte = new Integer(s);
        String num = inte.toString();
        char c;
        try {
            for(int i = 0; i < num.length(); i++) {
                c = num.charAt(i);
                imageNumber = changeNumber(c);
                g.drawImage(imageNumber, i * 50 + 75, 0, null);
            }
        }
        catch(IOException e) {}
    }
}
```

PDF document made with CodePrint using Prism

```
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.imageio.*;
import javax.swing.*;

public class Shrub {

    private BufferedImage img;
    private int xCoord;
    private int yCoord;
    private int width;
    private int height;
    private double velocity;

    public Shrub(int x, int y, BufferedImage i) {
        img = i;
        xCoord = x;
        yCoord = y;
        width = i.getWidth();
        height = i.getHeight();
    }

    public int getXCoord() {
        return xCoord;
    }

    public int getYCoord() {
        return yCoord;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

    public void scroll() {
        xCoord -= velocity;
    }

    public void updateV(double v) {
        velocity = v;
    }

    public void draw(Graphics g) {
        g.drawImage(img, xCoord, yCoord, null);
    }
}
```

```java
//imports
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.util.*;
import javax.swing.*;

public class Stage extends JPanel implements KeyListener, ActionListener {

    //stage height and width
    private static final int STAGE_HEIGHT = 500;
    private static final int STAGE_WIDTH = 1000;

    //private
    private File dat;
    private javax.swing.Timer timer;
    private Player player;
    private Floor floor;
    private ArrayList<Obstacle> obstacles;
    private LoseScreen losescreen;
    private Score scoreboard;
    private Highscore highscore;
    private int score;
    private int high_score;
    private Background background;
    private double velocity;

    //lose varriable
    public static boolean lose;


    public Stage(int width, int height) {
        setPreferredSize(new Dimension(width, height));
        player = new Player(50, 310);
        floor = new Floor(0, STAGE_HEIGHT - 100);
        timer = new javax.swing.Timer(20, this);
        losescreen = new LoseScreen();
        scoreboard = new Score();
        initializeDataFile();
        highscore = new Highscore(high_score);
        background = new Background();
        obstacles = new ArrayList<>();
        obstacles.add(new Obstacle(800, 290, 7));
        this.setFocusable(true);
        addKeyListener(this);
        lose = false;
        timer.start();
    }

    public void initializeDataFile() {
        try {
            dat = new File("info.dat");
            Scanner scan = new Scanner(dat);
            high_score = Integer.valueOf(scan.nextLine());
        }
        catch(IOException e) {}
    }

    public void closeDataFile() {
        try {
            PrintWriter pw = new PrintWriter(dat);
            pw.println(score);
            pw.close();
        }
        catch(IOException e) {}
    }

    public void actionPerformed(ActionEvent e) {
        player.gravity();
        status();
        if(!lose) {
            for(Obstacle o : obstacles) {
                o.scroll();
                velocity = o.getV() + (score * 0.01);
                o.updateV(velocity);
            }
            background.scroll(velocity);
        }
        else {
            timer.stop();
        }
        repaint();
    }

    public void keyPressed(KeyEvent e) {
        if(lose) {
            switch(e.getKeyCode()) {
                case KeyEvent.VK_UP:
                    losescreen.dotUp();
                    repaint();
                    break;
                case KeyEvent.VK_DOWN:
                    losescreen.dotDown();
                    repaint();
                    break;
                case KeyEvent.VK_ENTER:
                    if(losescreen.getDot() - 200 == 0) {
                        start();
                        lose = false;
                    }
                    else {
                        System.exit(0);
                    }
                    break;
            }
        }
        else if(e.getKeyCode() == KeyEvent.VK_SPACE) {
            player.jump();
        }
    }

    public void keyTyped(KeyEvent e) {}
    public void keyReleased(KeyEvent e) {}

    public void paintComponent(Graphics g) {
        super.paintComponent(g);

        background.draw(g);
        scoreboard.draw(g, score);
        highscore.draw(g);
        player.draw(g);
        floor.draw(g);
        for(Obstacle o : obstacles) {
            o.draw(g);
        }
        if(lose) {
            for(Obstacle o : obstacles) {
                o.drawDeath(g);
            }
            losescreen.draw(g);
        }
    }

    public void status() {
        for(Obstacle o : obstacles) {
            //System.out.print(player + "\t" + o + "\r");
            if(player.collision(o.getXCoord(), o.getYCoord(), o.getWidth(), o.getHeight())) {
                lose = true;
                if(score > high_score) {
                    closeDataFile();
                }
                repaint();
            }
        }

        if(obstacles.get(0).getXCoord() <= -70) {
            int y = ((int)(Math.random() * 10) % 2) * 100 + 200;
            obstacles.add(new Obstacle(1000, y, 7));
            obstacles.remove(0);
            score++;
        }
    }

    public void start() {
        score = 0;
        player = new Player(50, 310);
        floor = new Floor(0, STAGE_HEIGHT - 100);
        timer = new javax.swing.Timer(20, this);
        losescreen = new LoseScreen();
        scoreboard = new Score();
        initializeDataFile();
        highscore = new Highscore(high_score);
        obstacles = new ArrayList<>();
        obstacles.add(new Obstacle(800, 290, 7));
        this.setFocusable(true);
        addKeyListener(this);
        lose = false;
        timer.start();
    }

    public static void main(String args[]) {
        JFrame frame = new JFrame();
        frame.setTitle("Cat run");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel panel = new Stage(STAGE_WIDTH, STAGE_HEIGHT);
        Container c = frame.getContentPane();
        c.add(panel);
        frame.pack();
        frame.setVisible(true);
        frame.setResizable(false);
    }
}
```

PDF document made with CodePrint using Prism