KARE PUZZLE

Batuğ Can AKPINAR
201307002
Kocaeli Üniversitesi
İstanbul, Ümraniye
batugcann@gmail.com

Ömer MERCAN
201307022
Kocaeli Üniversitesi
Kocaeli, İzmit
omermercan41@gmail.com

Özet-proje kapsamında bir puzzle çözme oyunu tasarlanması beklenmektedir. Puzzle parça eşleştirmeleri için bağlı liste veri yapısı kullanılması zorunludur. Oyun için bir kullanıcı arayüzü (GUI) hazırlanacak ve proje, masaüstü uygulaması olarak veya web tabanlı olarak geliştirilebilir. Bu projenin amaçları ve gereklilikleri ayrıntılı olarak belirtilmiştir ve en yüksek skorun kaydedilmesi, puanlama sistemleri ve kullanıcı adı bilgilerinin skor belgesine kaydedilmesi de dahil olmak üzere birçok özellik içermektedir.

Anahtar Kelimeler — ;Puzzle, Bağlı liste, VS Code, Java, Masaüstü Uygulaması

GİRİŞ

Bir puzzle çözme oyununun tasarımı ve geliştirilmesi sürecini anlatmaktadır. Bu projenin amacı, kullanıcıların keyifli bir oyun deneyimi yaşamalarını sağlamak ve aynı zamanda veri yapıları ve algoritmalar konusunda bilgi sahibi olmalarına yardımcı olmaktır. Projede, bağlı liste veri yapısı kullanılarak puzzle parçalarının eşleştirilmesi sağlanmaktadır. Ayrıca, bir kullanıcı arayüzü (GUI) hazırlanarak oyunun kullanıcı dostu bir şekilde tasarlanması amaçlanmıştır. Proje, masaüstü uygulaması veya web tabanlı olarak geliştirilebilir. puzzle oyunu tasarımı ve geliştirme süreci hakkında ayrıntılı bir bilgi sağlamakla kalmayıp, aynı zamanda proje yönetimi ve yazılım geliştirme süreci hakkında da fikir vermektedir.

Puzzle Game projesi, kullanıcının oyunu oynamasını kolaylaştırmak için özenle tasarlanmış bir oyun platformudur. GUI tasarımı, kullanıcının oyunu oynamasını kolaylaştırmak için basit ve kullanışlıdır ve oyunun atmosferini yansıtmak için özenle seçilmiş renkler ve yazı tipi kullanılmıştır.

Projenin veri analizi kısmı, kullanıcının oyunu oynarken yaptığı hamlelerin sayısı ve oyunu tamamlama süresi gibi verilerin toplanması ile başlamıştır. Bu veriler, Java fonksiyonları kullanılarak toplanmış ve istatistiksel analizler için kullanılmıştır.

Hamle sayısı ve tamamlama süresi gibi verilerin ortalaması, standart sapması ve diğer istatistiksel özellikleri hesaplanmıştır. Verilerin analizi, oyunun zorluk seviyesinin belirlenmesi ve kullanıcının performansının değerlendirilmesi için kullanılmıştır.

Örneğin, hamle sayısı ve tamamlama süresi gibi veriler, oyunun zorluk seviyesinin belirlenmesinde kullanılmıştır. Kullanıcının performansı ise, tamamlama süresi gibi verilerin analizi ile değerlendirilmiştir.

Verilerin analizi, grafikler ve tablolar kullanılarak görselleştirilmiştir. Bu sayede, verilerin daha kolay anlaşılması ve yorumlanması sağlanmıştır. Verilerin analizi, oyunun geliştirilmesi ve kullanıcının deneyimini iyileştirmek için önemli bir adımdır.

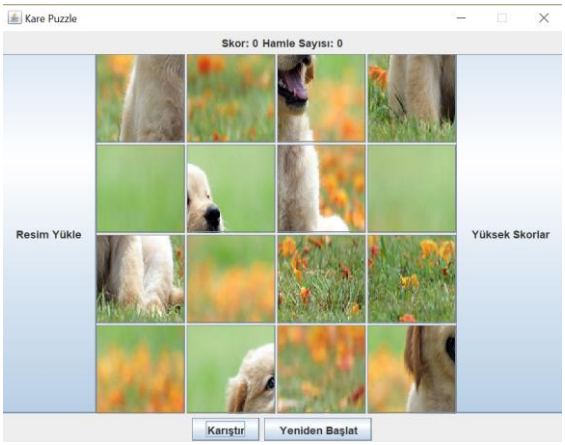
Sonuç olarak, Puzzle Game projesi, kullanıcının oyunu oynamasını kolaylaştırmak için özenle tasarlanmış bir oyun platformudur. Veri analizi kısmı, oyunun zorluk seviyesinin belirlenmesi ve kullanıcının performansının değerlendirilmesi için önemli bir adımdır. Verilerin analizi, grafikler ve tablolar kullanılarak görselleştirilmiş ve verilerin daha kolay anlaşılması ve yorumlanması sağlanmıştır.

FRONTEND

Java, nesne yönelimli bir programlama dilidir, bu da büyük ve karmaşık uygulamaların geliştirilmesini kolaylaştırır. Bu özellik, puzzle oyunu gibi karmaşık bir uygulamanın geliştirilmesinde oldukça yararlı olabilir.

Java'nın bir diğer avantajı, güçlü bir grafik arayüzü kütüphanesi olan Swing ve JavaFX'e sahip olmasıdır. Bu kütüphaneler, çeşitli grafik nesneleri ve bileşenleri sağlayarak kullanıcı arayüzlerinin geliştirilmesini kolaylaştırır.

Sonuç olarak, Java dilinin platform bağımsızlığı, nesne yönelimli programlama yetenekleri ve güçlü grafik kütüphaneleri nedeniyle puzzle uygulamanızda bir frontend dil olarak seçilebileceği söylenebilir. Bununla birlikte, front-end geliştirme için birçok farklı dil ve araçlar mevcuttur, bu nedenle seçiminiz projenizin gereksinimlerine ve ekibinizin uzmanlığına bağlı olacaktır.



Şekil 1.1

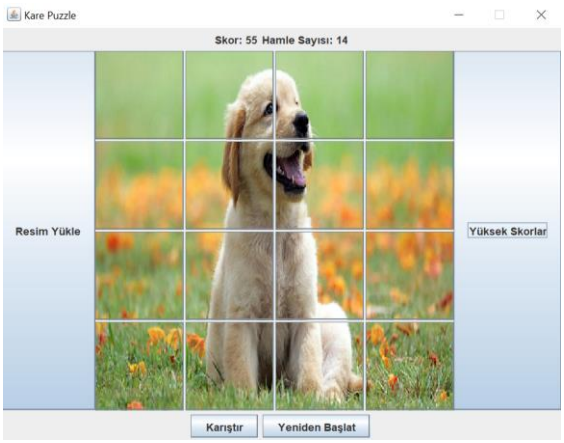
Puzzle Game, Java programlama dili kullanılarak geliştirilmiş bir bulmaca oyunudur. Oyunun amacı, 4x4'lük bir kare tahtada yer alan 15 parçayı doğru sıraya yerleştirmektir. Oyunun GUI'si, kullanıcının oyunu oynamasını kolaylaştırmak için tasarlanmıştır.

GUI, JFrame sınıfından türetilmiştir ve GridLayout kullanılarak düzenlenmiştir. Bu düzen, 4x4'lük bir kare tahtada yer alan 15 parçanın yerleştirilmesi için idealdir. Her bir parça, JButton sınıfından türetilmiştir ve ActionListener kullanılarak tıklanabilir hale getirilmiştir.

Oyunun başlangıcında, parçalar rastgele bir şekilde karıştırılır ve kullanıcının doğru sıraya yerleştirmesi gereken bir bulmaca oluşturulur. Kullanıcının parçaları doğru sıraya yerleştirmesi için, her bir parçanın tıklanması gerekmektedir. Tıklanan parça, boş bir yere taşınır ve kullanıcının diğer parçaları doğru sıraya yerleştirmesi için yeni bir fırsat verilir.

GUI, kullanıcının oyunu oynamasını kolaylaştırmak için birkaç özellik içerir. Örneğin, kullanıcının pencere boyutunu değiştirmesine izin verilmez ve oyunun sonunda bir mesaj kutusu görüntülenir. Bu mesaj kutusu, kullanıcının oyunu tamamladığını ve yeniden başlatmak isteyip istemediğini sormak için kullanılır.

GUI'nin tasarımı, kullanıcının oyunu oynamasını kolaylaştırmak için basit ve kullanışlıdır. Parçaların düzeni, kullanıcının oyunu oynamasını kolaylaştırmak için özenle seçilmiştir. Ayrıca, GUI'nin renkleri ve yazı tipi, oyunun atmosferini yansıtmak için özenle seçilmiştir.

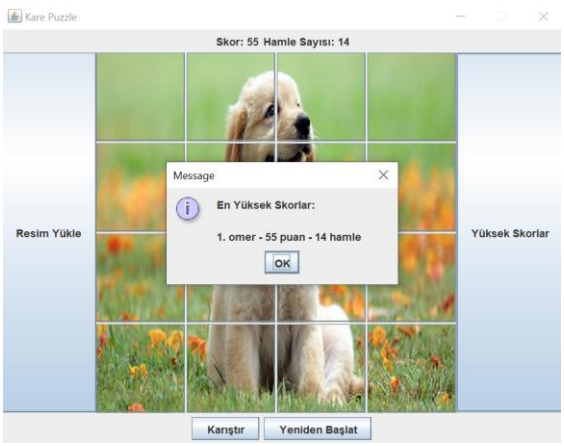


Şekil 1.2

Sonuç olarak, Puzzle Game projesinin tasarımı ve FrontEnd kısmı, kullanıcının oyunu oynamasını kolaylaştırmak için özenle tasarlanmıştır. GUI, kullanıcının oyunu oynamasını kolaylaştırmak için basit ve kullanışlıdır ve oyunun atmosferini yansıtmak için özenle seçilmiş renkler ve yazı tipi kullanılmıştır.

GUI, kullanıcının oyunu oynarken zamanı takip etmesine olanak tanır. GUI'nin alt kısmında bulunan bir sayaç, kullanıcının oyunu ne kadar sürede tamamladığını gösterir. Bu özellik, kullanıcının oyunu daha rekabetçi hale getirir ve daha fazla zorluk sağlar.

Puzzle Game projesinin FrontEnd kısmı, kullanıcının oyunu oynamasını kolaylaştırmak için responsive bir tasarıma sahiptir. Bu sayede, oyun farklı ekran boyutlarında ve cihazlarda da rahatlıkla oynanabilir.



Şekil 1.3

BACKEND

Backend kısmı, projenin ana işlevselliğini sağlayan kısımdır ve kullanıcı arayüzünden aldığı girdileri işleyerek, veri tabanı işlemlerini gerçekleştirerek ve oyun mantığını yürüterek, tam bir puzzle çözme oyunu deneyimi sunar. Bu bölümde, Java programlama dilini kullanarak, bağlı liste veri yapısı yardımıyla puzzle parçalarını yöneten kodları ele alacağız. Ayrıca, oyun için skor tutma ve puanlama sistemleri, kullanıcı adı bilgilerinin skor belgesine kaydedilmesi ve diğer birçok özellik de bu bölümde ele alınacaktır.

```
private void createUI() {
    frame = new JFrame();
    frame.setTitle(title:"Kare Puzzle");
    frame.setBounds(x:100, y:100, width:600, height:600);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(new BorderLayout(hgap:0, vgap:0));
    frame.setResizable(resizable:false);

    mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(rows:4, cols:4));
    frame.getContentPane().add(mainPanel, BorderLayout.CENTER);

    JButton restartButton = new JButton(text:"Yeniden Başlat");
    frame.getContentPane().add(restartButton, BorderLayout.NORTH);
    restartButton.addActionListener(e -> {
        frame.dispose();
        PuzzleGame game = new PuzzleGame();
        game.frame.setVisible(b:true);
    });

    puzzleButtons = new JButton[4][4];
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            puzzleButtons[i][j] = new JButton("P" + (i * 4 + j));
            puzzleButtons[i][j].setText(text:"");
```

Şekil 2.1

Şekil 2.1’ de bulunan kod bloğu, kare puzzle oyununun kullanıcı arayüzünü oluşturur. Oyunun ana paneli, yeniden başlatma, resim yükleme, karıştırma, yüksek skorlar gibi butonlar ve skor ve hamle sayısı gibi bilgilerin görüntülendiği bir durum paneli içerir. Ayrıca, puzzle parçalarının yerlerini değiştirmek için butonlar da oluşturulur.

```
private void shufflePuzzle(ActionEvent e) {
    if (puzzlePieces != null) {
        Collections.shuffle(puzzlePieces);
        updatePuzzleButtons();
        checkPuzzleCompletion();
    }
}
```

Şekil 2.2

Şekil 2.2’de bulunan kod bloğu, puzzle parçalarını karıştırmak için kullanılır. Eğer puzzle parçaları yüklenmişse, parçaların sırasını rastgele karıştırır. Daha sonra, karıştırılmış parçaların yerlerini güncelleyerek, kullanıcının puzzle'ı yeniden düzenlemesine olanak tanır. Son olarak, puzzle'ın tamamlanıp tamamlanmadığını kontrol eder.

```
private boolean checkPuzzleCompletion() {
    boolean completed = true;
    for (int i = 0; i < puzzlePieces.size(); i++) {
        PuzzlePiece piece = puzzlePieces.get(i);
        if (piece.position != i) {
            completed = false;
            break;
        }
    }

    if (completed) {
        JOptionPane.showMessageDialog(frame, message:"Tebrikler! Puzzle'ı tamamladınız!");
    }
    return completed;
}
```

Şekil 1.3

Şekil 1.3’ de bulunan kod bloğunda, puzzle'ın tamamlanıp tamamlanmadığını kontrol eder. Tüm puzzle parçalarının doğru pozisyonda olup olmadığını kontrol ederek, puzzle'ın tamamlandığını belirler. Eğer puzzle tamamlanmışsa, bir tebrik mesajı gösterir. Fonksiyon, tamamlanma durumunu boolean bir değer olarak döndürür.

```
private void loadImage() {
    JFileChooser fileChooser = new JFileChooser();
    int returnValue = fileChooser.showOpenDialog(frame);

    if (returnValue == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        try {
            originalImage = ImageIO.read(selectedFile);
            puzzleImages = splitImage(originalImage, rows:4, cols:4);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    puzzlePieces = new LinkedList<>();
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            int position = i * 4 + j;
            puzzlePieces.add(new PuzzlePiece(position, puzzleImages[i][j]));
        }
    }

    updatePuzzleButtons();
```

Şekil 1.4

Şekil a.4’de bulunan kod bloğunda, kullanıcının bir resim dosyası seçmesine olanak tanır ve seçilen resmi puzzle parçalarına ayırır. Kullanıcı, bir dosya seçmek için bir dosya seçici açar ve seçilen dosya, ImageIO sınıfı kullanılarak okunur. Okunan resim, splitImage() fonksiyonu kullanılarak 4x4 boyutunda puzzle parçalarına ayrılır. Daha sonra, her bir puzzle parçası, pozisyonu ve resmi içeren bir PuzzlePiece nesnesi olarak LinkedList'e eklenir. Son olarak, puzzle butonları güncellenir.

```
private BufferedImage[][] splitImage(BufferedImage image, int rows, int cols) {
    int imgWidth = image.getWidth() / cols;
    int imgHeight = image.getHeight() / rows;
    BufferedImage[][] images = new BufferedImage[rows][cols];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            images[i][j] = image.getSubimage(j * imgWidth, i * imgHeight, imgWidth, imgHeight);
        }
    }

    return images;
}
```

Şekil 1.5

Şekil 1.5’de bulunan kod bloğu, bir BufferedImage nesnesini, belirtilen satır ve sütun

sayısına göre parçalara ayırır. İlk olarak, resmin genişliği ve yüksekliği, sütun ve satır sayısına bölünerek, her bir parçanın boyutu belirlenir. Daha sonra, her bir parça, getSubimage() fonksiyonu kullanılarak orijinal resimden alınır ve BufferedImage[][] türünde bir diziye eklenir. Son olarak, parçalara ayrılmış resimlerin bulunduğu dizi, çağrıldığı yere döndürülür.

```
private void swapPieces(int clickedPosition) {
    PuzzlePiece clickedPiece = puzzlePieces.get(clickedPosition);

    if (clickedPiece.image == null) {

        return;
    }

    if (selectedPiece == null) {

        selectedPiece = clickedPiece;
    } else {
        boolean selectedCorrectBefore = selectedPiece.position == puzzlePieces.indexOf(selectedP
        boolean clickedCorrectBefore = clickedPiece.position == clickedPosition;

        boolean selectedPieceCorrect = selectedPiece.position == puzzlePieces.indexOf(selectedPi
        boolean clickedPieceCorrect = clickedPiece.position == clickedPosition;


        if (selectedPieceCorrect || clickedPieceCorrect) {
            selectedPiece = null;
            return;
        }

        int tempPosition = selectedPiece.position;
```

Şekil 1.6

Şekil 1.6'da bulunan kod bloğu, kullanıcının puzzle parçalarını yer değiştirmesini sağlar. Kullanıcı, bir puzzle parçasına tıkladığında, seçilen parça olarak işaretlenir. Daha sonra, kullanıcı başka bir parçaya tıkladığında, seçilen parça ile tıklanan parçanın yerleri değiştirilir. Yer değiştirme işlemi, PuzzlePiece nesnelerinin pozisyonlarını ve resimlerini değiştirerek gerçekleştirilir. Ayrıca, kullanıcının hamle sayısı ve skoru takip edilir. Her doğru yer değiştirme için skor artar, yanlış yer değiştirme için skor azalır. Puzzle tamamlandığında, kullanıcıya bir tebrik mesajı gösterilir ve skor kaydedilir.

```
private void updatePuzzleButtons() {
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            int position = i * 4 + j;
            PuzzlePiece piece = puzzlePieces.get(position);
            if (piece.image != null) {
                ImageIcon imageIcon = new ImageIcon(piece.image);
                puzzleButtons[i][j].setIcon(imageIcon);
            } else {
                puzzleButtons[i][j].setIcon(defaultIcon:null);
            }


            puzzleButtons[i][j].setPreferredSize(new Dimension(piece.image.getWidth(), piece.
        }
    }

    frame.pack();
}
```

Şekil 1.7

Şekil 1.7'de bulunan kod bloğunda, puzzle parçalarının yerlerini güncellemek için kullanılır. Her bir puzzle parçası, bir JButton bileşeni ile temsil edilir. Bu kod, her bir JButton bileşeninin resmini, puzzle parçasının resmi ile günceller. Eğer puzzle parçası boş ise, JButton bileşeninin resmi null olarak ayarlanır. Daha sonra, her bir JButton bileşeninin boyutu, puzzle parçasının boyutuna göre ayarlanır. Son olarak, JFrame bileşeninin boyutu, puzzle parçalarının boyutuna göre ayarlanır. Bu işlem, kullanıcının puzzle parçalarını daha iyi görebilmesini sağlar.

```
private static class HighScoreEntry {
    String playerName;
    int moveCount;
    int score;

    public HighScoreEntry(String playerName, int moveCount, int score) {
        this.playerName = playerName;
        this.moveCount = moveCount;
        this.score = score;
    }

    public int getScore() {
        return score;
    }
}
```

Şekil 1.8

Şekil 1.8'de bulunan kod bloğunda, yüksek skorları temsil eden bir sınıf olan HighScoreEntry sınıfını tanımlar. HighScoreEntry sınıfı, her bir yüksek skor için oyuncu adı, hamle sayısı ve skor bilgilerini içerir. Bu sınıf, yüksek skorları saklamak ve karşılaştırmak için kullanılır. getScore() fonksiyonu, HighScoreEntry nesnesinin skorunu döndürür.

```
private void saveHighScore() {
    String playerName = JOptionPane.showInputDialog(frame, message:"Adınızı girin:");
    if (playerName == null || playerName.trim().isEmpty()) {
        return;
    }

    highScores.add(new HighScoreEntry(playerName, moveCount, score));
    highScores.sort(Comparator.comparingInt(HighScoreEntry::getScore).reversed());

    try (BufferedWriter writer = new BufferedWriter(new FileWriter(HIGH_SCORE_FILE))) {
        for (HighScoreEntry entry : highScores) {
            writer.write(String.format(format:"%s, %d, %d%n", entry.playerName, entry.moveCou
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Şekil 1.9

Şekil 1.9'da bulunan kod bloğunda, yüksek skorları dosyaya kaydetmek için kullanılır. Kullanıcı, puzzle tamamlandığında adını girmesi istenir. Daha sonra, HighScoreEntry sınıfı kullanılarak yeni bir yüksek skor oluşturulur ve highScores listesine eklenir. highScores listesi, skorları saklamak için kullanılır ve yüksek skorlar, skorları büyükten küçüğe doğru sıralanarak kaydedilir. Yüksek skorlar, HIGH_SCORE_FILE adlı dosyaya yazılır. Dosya yazma işlemi, BufferedWriter sınıfı kullanılarak gerçekleştirilir. Dosya yazma işlemi sırasında bir hata oluşursa, hata mesajı yazdırılır.

```
private void showHighScores() {
    StringBuilder message = new StringBuilder(str:"En Yüksek Skorlar:\n\n");
    for (int i = 0; i < highScores.size() && i < 10; i++) {
        HighScoreEntry entry = highScores.get(i);
        message.append(String.format(format:"%d. %s - %d puan - %d hamle%n", i + 1, entry.playe
                entry.moveCount));
    }
    JOptionPane.showMessageDialog(frame, message.toString());
}
```

Şekil 1.10

Şekil 1.10'da bulunan kod bloğunda, yüksek skorları göstermek için kullanılır. showHighScores() fonksiyonu, highScores listesindeki en yüksek 10 skoru alır ve bunları bir mesaj kutusunda gösterir. Mesaj kutusu, JOptionPane sınıfı kullanılarak oluşturulur. Mesaj kutusu, her bir yüksek skor için oyuncu adı, skor ve hamle sayısını içeren bir metin içerir. Mesaj kutusu, kullanıcıya en yüksek skorları gösterir ve oyunun ne kadar zor olduğunu gösterir.

```
private boolean isPuzzleShuffled() {
    for (int i = 0; i < puzzlePieces.size(); i++) {
        PuzzlePiece piece = puzzlePieces.get(i);
        if (piece.position != i) {
            return true;
        }
    }
    return false;

}
```

Şekil 1.11

Şekil 1.11'de bulunan kod bloğunda, puzzle parçalarının karıştırılıp karıştırılmadığını kontrol etmek için kullanılır. isPuzzleShuffled() fonksiyonu, her bir puzzle parçasının pozisyonunu kontrol eder ve parçaların doğru pozisyonda olup olmadığını kontrol eder. Eğer bir puzzle parçası doğru pozisyonda değilse, fonksiyon true değerini döndürür. Bu, puzzle parçalarının karıştırıldığını gösterir. Eğer tüm puzzle parçaları doğru pozisyonda ise, fonksiyon false değerini döndürür. Bu, puzzle parçalarının karıştırılmadığını gösterir

```
private static class PuzzlePiece {
    int position;
    BufferedImage image;

    public PuzzlePiece(int position, BufferedImage image) {
        this.position = position;
        this.image = image;
    }
}
```

Şekil 1.12

Şekil 1.12'de bulunan kod bloğunda, puzzle parçalarını temsil eden bir sınıf olan PuzzlePiece sınıfını tanımlar. PuzzlePiece sınıfı, her bir puzzle parçası için pozisyon ve resim bilgilerini içerir. Bu sınıf, puzzle parçalarını oluşturmak ve yönetmek için kullanılır. Her bir puzzle parçası, bir JButton bileşeni ile temsil edilir ve PuzzlePiece sınıfı, her bir puzzle parçasının pozisyonunu ve resmini tutar. Bu sınıf, puzzle parçalarının yerlerini değiştirmek ve orijinal resmi yeniden oluşturmak için kullanılır,

```
import javax.swing.*;
import java.awt.*;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.LinkedList;
import java.awt.event.ActionEvent;
import java.util.Collections;
import java.io.*;
import java.util.ArrayList;
import java.util.Comparator;
```

Şekil 1.13

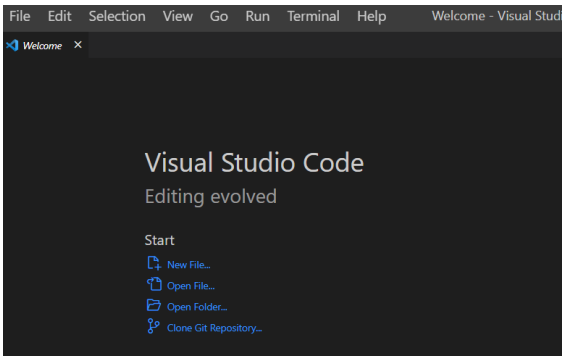
Şekil 1.13'de bulunan kod bloğunda, Bu paketler, Java programlama dilinde farklı işlevleri yerine getirmek için kullanılan hazır kütüphaneleri içerir. Aşağıdaki açıklamalar, bu kodda kullanılan paketlerin ne işe yaradığını özetler:

- javax.swing: Java Swing kütüphanesi, grafik kullanıcı arayüzü bileşenleri sağlar. Bu kodda, JFrame, JPanel, JButton, JLabel ve JOptionPane gibi Swing bileşenleri kullanılarak bir kare puzzle oyunu oluşturulur.
- java.awt: Java Abstract Window Toolkit (AWT) kütüphanesi, grafik kullanıcı arayüzü bileşenleri sağlar. Bu kodda, GridLayout ve BorderLayout gibi AWT bileşenleri kullanılarak oyunun düzeni oluşturulur.
- javax.imageio: Java Image I/O API, resim dosyalarını okumak ve yazmak için kullanılır. Bu kodda, kullanıcının seçtiği resim dosyası ImageIO sınıfı kullanılarak okunur.
- java.awt.image: Java AWT Image sınıfı, resimleri temsil etmek için kullanılır. Bu kodda, BufferedImage sınıfı kullanılarak resimler puzzle parçalarına ayrılır.
- java.io: Java Input/Output API, dosya işlemleri için kullanılır. Bu kodda, yüksek skorlar dosyaya yazılır ve dosyadan okunur.
- java.util: Java Utility API, farklı amaçlar için kullanılan yardımcı sınıflar sağlar. Bu

kodda, LinkedList, ArrayList ve Comparator gibi sınıflar kullanılarak puzzle parçaları, yüksek skorlar ve yüksek skorları karşılaştırmak için kullanılır.

TEKNOLOJİLER

1.VS Code



Şekil 3.1

A. VS Code Nedir?

Şekil b.1'de bulunan görselde bulunan VS Code, Microsoft tarafından geliştirilen ve açık kaynak kodlu bir kod editörüdür. Çoklu platform desteği sayesinde Windows, macOS ve Linux işletim sistemlerinde kullanılabilmektedir. Ayrıca, birçok programlama dili için destek sağlaması, eklenti sistemi ve hafif arayüzü ile geliştiriciler arasında popüler bir seçimdir.

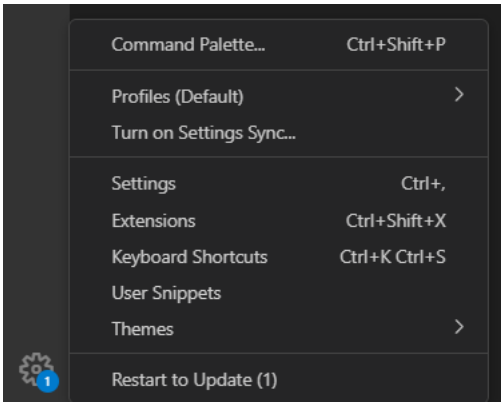
B. Neden VS Code?

Projenin geliştirilmesinde VS Code, yazılım geliştirme sürecini kolaylaştırmak için kullanılmıştır. Kod yazma, hata ayıklama ve kod refaktörleme işlemleri için gerekli araçları içermesi, projenin hızlı ve etkili bir şekilde geliştirilmesine olanak sağlamıştır.

C. VS Code Avantajları Nedir?

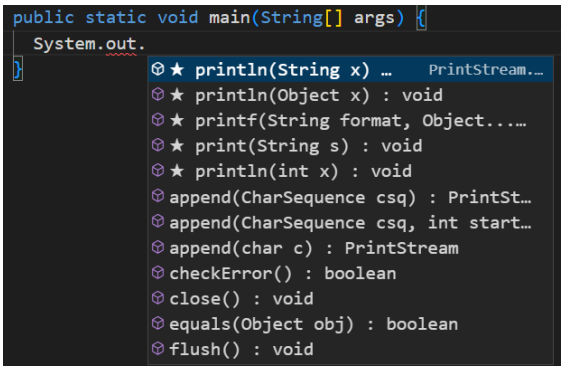
VS Code'un birçok avantajı bulunmaktadır. İlk olarak, açık kaynak kodlu olması sayesinde geliştiricilerin ihtiyaçlarına uygun şekilde özelleştirilebilir. Ayrıca, zengin eklenti sistemi sayesinde farklı programlama dilleri ve araçları için destek sağlar. VS Code, kod yazarken otomatik tamamlama, hata ayıklama, renklendirme gibi birçok özelliği de içermektedir. Bunun yanı sıra, Git ve diğer sürüm kontrol sistemleriyle entegre çalışması, projelerin kolaylıkla yönetilmesini sağlamaktadır.

Aşağıdaki görsellerde VS Code'un birkaç avantajı gösterilmiştir:



Şekil 3.2

VS Code'un özelleştirme seçenekleri (Şekil b.2)



Şekil 3.3

VS Code'un otomatik tamamlama özelliği (Şekil b.3)

D. Eklentiler

Şekil 3.4

Extension Pack for Java, Visual Studio Code için bir eklenti paketidir ve Java programlama dili için geliştirme ortamı sağlar. Bu eklenti paketi, Java geliştirme için gerekli olan tüm araçları içerir. Örneğin, Java kodu düzenlemek için gerekli olan kod tamamlama, hata ayıklama, semantik analiz ve kod biçimlendirme gibi özelliklerin yanı sıra, Maven, Gradle ve Ant gibi popüler yapılandırma yöneticilerini de destekler.

Extension Pack for Java ayrıca, Java uygulamalarının paketlenmesi, derlenmesi, dağıtımı

ve yürütülmesi gibi işlemleri yapmak için gerekli olan araçları da sağlar. Bu araçlar, Java Virtual Machine (JVM) için kodu optimize etmek için kullanılan Just-In-Time (JIT) derleyicisi gibi, uygulamaları hızlandırmak için faydalıdır.

VS Code'da Extension Pack for Java kullanmak, Java geliştirme sürecini kolaylaştırır ve verimliliği artırır. Ayrıca, VS Code'un hafif ve hızlı yapısından da yararlanarak, daha az kaynak kullanımıyla daha iyi bir performans elde edebilirsiniz.

KAYNAKÇA

https://www.geeksforgeeks.org/java/

https://www.w3schools.com/java/java_linkedlist.asp

https://stackoverflow.com/

https://code.visualstudio.com/

https://www.java.com/tr/

https://www.geeksforgeeks.org/data-structures/linked-list/

https://www.geeksforgeeks.org/data-structures/

https://www.w3schools.io/algorithms/

https://www.w3schools.com/java/java_packages.asp

https://picsum.photos/

https://stackoverflow.com/questions/tagged/java