Struktura e të dhënave: Vargjet

Struktura e të dhënave: Vargjet

Objektivat:

- Të ndërtohen vargje të cilat përmbajnë vlera të tipeve primitive ose objekte, dhe pastaj të zbatohen vargje si parametra dhe rezultate metodash.
- Të zbatohen vargje për të modeluar koleksione nga bota reale.
- Të kuptohet rangu i aplikimeve të vargjeve njëdimensionale dhe vargjeve dydimensionale.

Vargjet njëdimensionale

Agregat (strukturë të dhënash): objekt i cili përmban një koleksion vlerash të të dhënave, qoftë primitive ose të referencës (objekte).

Varg (array): agregat i cili përmban një koleksion të përbërë nga numër i pandryshueshëm vlerash, të quajtura *elemente*, të të njëjtit tip. Elementet indeksohen sipas *indeksave* numra të plotë jonegativë.

Varg njëdimensional: varg koleksioni i të cilit është „një sekuencë“ vlerash, d.m.th., secili element indeksohet me një indeks të vetëm.

Vargjet njëdimensionale – Vazhdim

```
int[] r = new int[6];
r[1] = 7;
int i = 6;
r[3] = r[i - 5] + 2;

int[] s = r;
```

int[] r == a1

int[] s == a1

a1 : int[6]

0	1	2	3	4	5
0	7	0	9	0	0

Vargjet njëdimensionale – Vazhdim

```
int[] r = new int[6];
r[0] = 1;
for ( int i = 1; i < r.length; i++ )
{ r[i] = r[i - 1] * 2; }
```

int[] r == a1

a1 : int[6]

0	1	2	3	4	5
1	2	4	8	16	32

Vargjet njëdimensionale – Vazhdim

```
public double[] invert(double[] r)
{ int size = r.length;
  double[] answer = new double[size];
  for ( int i = 0; i != size; i++ )
  { answer[size - 1 - i] = r[i]; }
  return answer;
}

//...
double[] d = {0.2, -1.7, 3.14, 2};
double[] e = invert(d);

public static void main(String[] args)
{ // ... args[0] ... args[1] ...
}
```

Grumbullimi i të dhënave hyrëse në vargje

```
import javax.swing.*;
/** Numëron votat për kandidatët elektoralë.
 *  input: një varg votash, i terminuar nga -1
 *  output: lista e rezultateve të votave për kandidatë */
public class VoteCount
{ public static void main(String[] args)
  { int numCandidates = 4;
    int[] votes = new int[numCandidates];
```

Grumbullimi i të dhënave hyrëse në vargje – Vazhdim

```
    boolean processing = true;
    while ( processing )
    { int v = new Integer(JOptionPane.showInputDialog
                  ("Votoni për (0,1,2,3):")).intValue();
      if ( v == -1 )
      { processing = false; }
      else if ( v >= 0 && v < numCandidates )
      { votes[v]++; }
      else { JOptionPane.showMessageDialog(null,
                  "Gabim në votim: " + v);
           }
    }
    for ( int i = 0; i < numCandidates; i++ )
    { System.out.println("Kandidati " + i + " ka "
          + votes[i] + " vota"); }
  }
}
```

Tabelat përkthyese

```
int[] code = new int[27];
int key = new Integer(JOptionPane.showInputDialog
              ("Jepni çelësin: ")).intValue();
code[0] = key;
for ( int i = 1; i != code.length; i++ )
{ code[i] = (int)(code[i - 1] * 1.3 + 1); }
```

Tabelat përkthyese – Vazhdim

```
String input = JOptionPane.showInputDialog
               ("Rradhisni një fjali:");
for ( int j = 0; j != input.length(); j++ )
{ char c = input.charAt(j);
  if ( c == ' ' )
  { System.out.println(code[0]); }
  else if ( c >= 'a' && c <= 'z' )
  { int index = c - 'a' + 1;
    System.out.println(code[index]);
  }
  else { System.out.println
               ("Gabim: Karakter jokorrekt hyrës"); }
}
```

Vargje objektesh

```
BankAccount[] account = new BankAccount[100];
account[1] = new BankAccount(20);

for ( int i = 0; i < account.length; i++ )
{ if ( account[i] != null )
  { System.out.println("Balansi i kontos " + i
                + " është " + account[i].getBalance());
  }
}
```

Vargje objektesh – Vazhdim

```
BankAccount[] account == a1
```

a1 : BankAccount[100]

0	1	2	...	99
null	a2	null	...	null

a2 : BankAccount

```
int balance == 20
public boolean deposit(int amount) {...}
public boolean withdraw(int amount) {...}
int getBalance() {...}
```

Modelimi me vargje. Case study: bazat e të dhënave

Bazë e të dhënash (database): koleksion i madh vlerash të të dhënave i cili duhet të mirëmbahet me anë të veprimeve të insertimit, kthimit dhe fshierjes së vlerave të të dhënave.

Çelës (key): kodi i identitetit (ID) i cili zbatohet për të identifikuar dhe kthyer një vlerë të dhënash të ruajtur në një bazë të dhënash.

Regjistrim (record): vlerë e të dhënash element i një baze të dhënash.

Case study: bazat e të dhënave – Vazhdim

`class Database`	Konteiner për elemente të dhënash të quajtura `Record`.
Konstruktori:	
`Database(int initialSize)`	Inicializon bazën me madhësinë e dhënë fillestare.
Metodat:	
`boolean insert(Record r)`	Përpiqet të insertojë një regjistrim në bazën e të dhënave.
`Record find(Key k)`	Përpiqet të alokojë në bazën e të dhë-nave regjistrimin me çelësin `k`.
`boolean delete(Key k)`	Përpiqet të fshijë nga baza e të dhënave regjistrimin me çelësin `k`.

Tabela 1. Interfejsi për bazë të dhënash

Case study: bazat e të dhënave – Vazhdim

`Record`	Element i të dhënash i cili mund të ruhet në një bazë të dhënash.
Metodat:	
`Key keyOf()`	Kthen çelësin i cili e identifikon në mënyrë unike regjistrimin.

`Key`	Vlerë identifikuese ose „çelës“.
Metodat:	
`boolean equals(Key m)`	Krahason në barazim veten me një çelës tjetër.
`boolean lessthan(Key m)`	Krahason në mosbarazim veten me një çelës tjetër.

Case study: bazat e të dhënave – Vazhdim

```
/** Implementon një bazë të dhënash. */
public class Database
{ private Record[] base;
  private int count;
  private int NOT_FOUND = -1;
  /** Inicializon bazën.
   *  @param initialSize - madhësia fillestare e bazës */
  public Database(int initialSize)
  { if ( initialSize > 0 )
    { base = new Record[initialSize]; }
    else { base = new Record[1]; }
    count = 0;
  }
```

Case study: bazat e të dhënave – Vazhdim

```
private int locationOf(Key k)
{ int result = NOT_FOUND;
  boolean found = false;
  int i = 0;
  while ( !found && i != base.length )
  { if ( base[i] != null && base[i].keyOf().equals(k) )
    { found = true;
      result = i;
    }
    else { i++; }
  }
  return result;
}
```

Case study: bazat e të dhënave – Vazhdim

```
/** Alokon një regjistrim mbështetur në çelësin.
 *  @param k - çelësi i regjistrimit të kërkuar
 *  @return  rrokja e bazës (adresa e saj);
 *    null në qoftë se rrokja nuk është gjetur */
public Record find(Key k)
{ Record answer = null;
  int index = locationOf(k);
  if ( index != NOT_FOUND )
  { answer = base[index]; }
  return answer;
}
```

Case study: bazat e të dhënave – Vazhdim

```
/** Inserton një regjistrim në bazën e të dhënave.
 *  @param r - rrokja
 *  @return  true në qoftë se rrokja është shtuar;
 *    false në qoftë se rrokja nuk është shtuar */
public boolean insert(Record r)
{ boolean success = false;
  if ( locationOf(r.keyOf()) == NOT_FOUND )
  { boolean foundSlot = false;
    int i = 0;
    while ( !foundSlot && i != base.length )
    { // deri më tani base[0]...base[i-1] janë të zëna
      if ( base[i] == null )
      { foundSlot = true; }
      else { i++; }
    }
```

Case study: bazat e të dhënave – Vazhdim

```
    if ( foundSlot )
    { base[i] = r; }
    else { Record[] temp = new Record[2 * base.length];
           for ( int j = 0; j != count; j++ )
           { // kopjohet përmbajtja e base në temp
             temp[j] = base[j];
           }
           base = temp;
           base[count] = r;
         }
    count++;
    success = true;
  }
  return success;
}
```

Case study: bazat e të dhënave – Vazhdim

```
  /** Fshin një regjistrim nga baza e të dhënave.
   *  @param k - çelësi i regjistrimit
   *  @return  true në qoftë se rrokja është fshirë;
   *    false në qoftë se rrokja nuk është fshirë */
  public boolean delete(Key k)
  { boolean success = false;
    int index = locationOf(k);
    if ( index != NOT_FOUND )
    { base[index] = null;
      count--;
      success = true;
    }
    return success;
  }
}
```

Case study: bazat e të dhënave – Vazhdim

```
Record r1 = new Record(...);
Record r2 = new Record(...);
Record r3 = new Record(...);
Database db = new Database(4);
boolean ok = db.insert(r1);
ok = db.insert(r2);
ok = db.insert(r3);
boolean deletationResult = db.delete(r2.keyOf());
```

Case study: bazat e të dhënave – Vazhdim

Database db == a4

Record r1 == a1

Record r2 == a2

Record r3 == a3

a5 : Record[4]

0	1	2	3
a1	null	a3	null

a4 : Database

Record[] base == a5

int count == 2

```
public boolean insert(Record r) {...}
public Record find(Key k) {...}
public boolean delete(Key k) {...}
private int locationOf(Key k) {...}
```

a1 : Record

...

a2 : Record

...

a3 : Record

...

Case study: lojë kartash

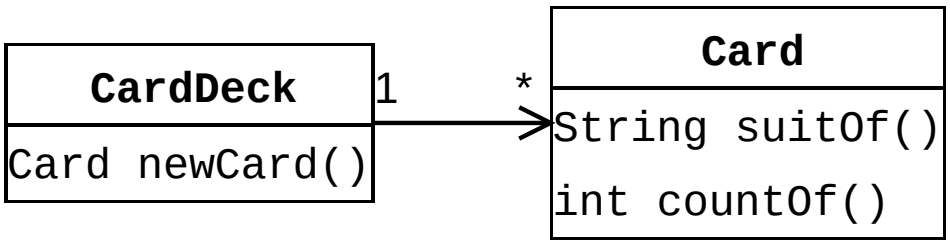


Figura 1. Arkitektura e nënkoleksionit të kartave të lojës dhe shpilit

Case study: lojë kartash – Vazhdim

`class CardDeck`	modelon një shpil kartash
Metodat:	
`Card newCard()`	merr një kartë të re nga shpili
`boolean moreCards()`	tregon se a ka akoma karta në shpil
Kolaborator:	`Card`

Tabela 2. Specifikimi për `class CardDeck`

Case study: lojë kartash – Vazhdim

`class Card`	modelon një kartë loje
Konstruktori:	
`Card(String s, int c)`	vë vlerën dhe ngjyrën e kartës
Metodat:	
`String suitOf()`	kthen ngjyrën e kartës
`int countOf()`	kthen vlerën e kartës

Tabela 3. Specifikimi për `class Card`

Case study: lojë kartash – Vazhdim

```
/** Modelon një kartë loje. */
public class Card
{ public static final String SPADES = "spades";
  public static final String HEARTS = "hearts";
  public static final String DIAMONDS = "diamonds";
  public static final String CLUBS = "clubs";
  public static final int SUIT_SIZE = 13;
  public static final int ACE = 1;
  public static final int JACK = 11;
  public static final int QUEEN = 12;
  public static final int KING = 13;

  private String suit;
  private int count;
```

Case study: lojë kartash – Vazhdim

```
  /** Vë vlerën dhe ngjyrën e kartës.
   *  @param s - ngjyra
   *  @param c - vlera  */
  public Card(String s, int c)
  { suit = s;
    count = c;
  }
  /** Kthen ngjyrën e kartës. */
  public String suitOf()
  { return suit; }
  /** Kthen vlerën e kartës */
  public int countOf()
  { return count; }
}
```

Case study: lojë kartash – Vazhdim

Variabël finale: variabël vlera e së cilës nuk mund të ndryshohet pas inicializimit; deklarohet me anë të fjalës kyçe `final`.

Variabël statike: variabël e cila nuk kopjohet si fushë në ndonjë objekt të konstruktuar nga klasa; deklarohet me anë të fjalës kyçe `static`.

Case study: lojë kartash – Vazhdim

```
/** Modelon një shpil kartash. */
public class CardDeck
{ private Card[] deck = new Card[4 * Card.SUIT_SIZE];
  private int count;
  /** Krijon shpil të ri me të gjitha kartat. */
  public CardDeck()
  { createSuit(Card.SPADES);
    createSuit(Card.HEARTS);
    createSuit(Card.DIAMONDS);
    createSuit(Card.CLUBS);
  }
```

Case study: lojë kartash – Vazhdim

```
/** Merr një kartë të re nga shpili.
 * @return  një kartë të re (ose null nëse nuk ka karta) */
public Card newCard()
{ Card nextCard = null;
  if ( count == 0 )
  { System.out.println("Gabim: nuk kanë mbetur karta"); }
  else
  { int index = (int)(Math.random() * count);
    nextCard = deck[index];
    for ( int i = index + 1; i != count; i++ )
    // kartat nga index+1...i-1 janë lëvizur në të majtë
    { deck[i-1] = deck[i]; }
    count--;
  }
  return nextCard;
}
```

Case study: lojë kartash – Vazhdim

```
  /** Tregon se a ka akoma karta në shpil.
   * @return  se a është shpili jo i zbrazët */
  public boolean moreCards()
  { return count > 0; }

  private void createSuit(String s)
  { for ( int i = 1; i <= Card.SUIT_SIZE; i++ )
    { deck[count] = new Card(s, i);
      count++;
    }
  }
}
```

Vargjet dydimensionale

Varg dydimensional: „varg vargjesh“ që tipikisht paraqitet si matricë. Secili element emërtohet me anë të një çifti indeksash. Indeksi i parë indekson *rreshtin*, kurse i dyti *shtyllën* e matricës.

Vargjet dydimensionale – Vazhdim

```
int numRegions = 4;
int numCandidates = 3;
int[][] votes = new int[numRegions][numCandidates];
for ( int i = 0; i != votes.length; i++ )
{ for ( int j = 0; j != votes[i].length; j++ )
  { int v = new Integer(JOptionPane.showInputDialog("Regjioni "
        + i + ", Kandidati " + j + ":")).intValue();
    if ( v >= 0 ) { votes[i][j] = v; }
    else { JOptionPane.showMessageDialog(null,
                    "Gabim në votim: " + v);
         }
  }
}
```

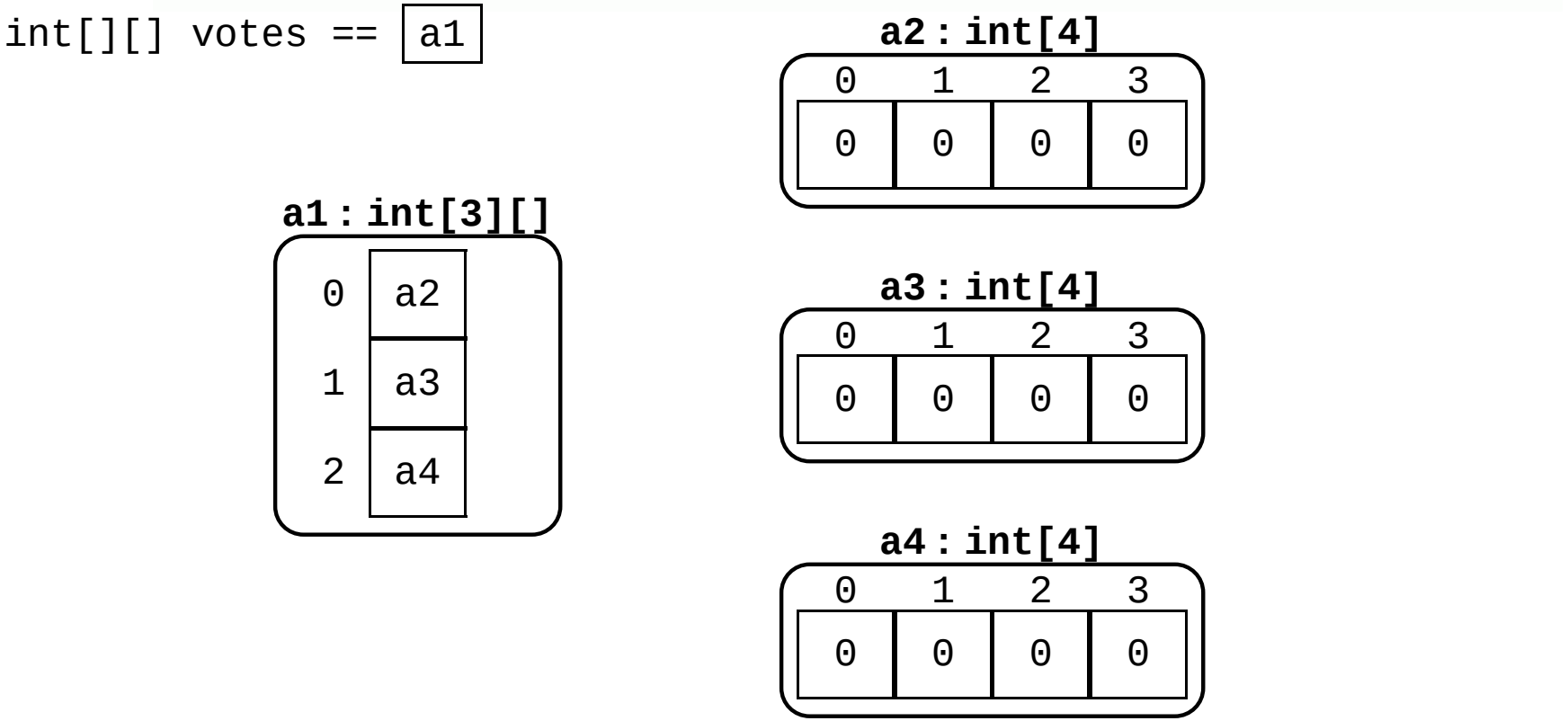
Vargjet dydimensionale – Vazhdim

```
votes[1][2]++;

for ( int j = 0; j != numCandidates; j++ )
{ votes[1][j] = 200; }

for ( int i = 0; i != numRegions; i++ )
{ votes[i][2] += 100; }
```

Vargjet dydimensionale – Vazhdim

```
for ( int j = 0; j != numCandidates; j++ )
{ int total = 0;
  for ( int i = 0; i != numRegions; i++ )
  { total += votes[i][j]; }
  System.out.println("Kandidati " + j + " ka "
        + total + " vota");
}

for ( int i = 0; i != numRegions; i++ )
{ int total = 0;
  for ( int j = 0; j != numCandidates; j++ )
  { total += votes[i][j]; }
  System.out.println(total + " vota në rajonin " + i);
}
```

Vargjet dydimensionale – Vazhdim



Vargjet dydimensionale – Vazhdim

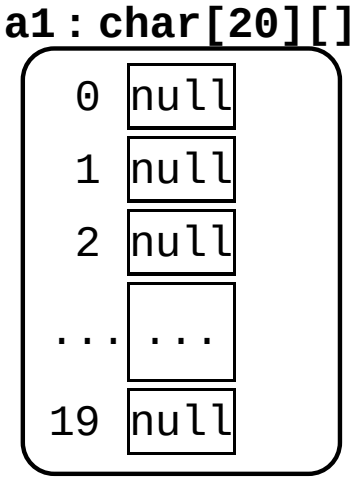
```
System.out.println("Numri i rajoneve: "
    + votes.length);

System.out.println("Numri i kandidatëve: "
    + votes[0].length);
```

Vargjet dydimensionale – Vazhdim

```
int maxWords = 20;
char[][] word = new char[maxWords][];
int count = 0;
boolean processing = true;
while ( processing )
{ String s = JOptionPane.showInputDialog("Fjala: ");
  if ( s.equals(""))
  { processing = false; }
  else { word[count] = new char[s.length()];
         for ( int j = 0; j != s.length(); j++ )
         { word[count][j] = s.charAt(j); }
         count++;
       }
}
```

Vargjet dydimensionale – Vazhdim

`char[][] word ==` a1



Vargjet dydimensionale – Vazhdim

`char[][] word ==` a1

a1 : char[20][]

0	a2
1	null
2	null
...	...
19	null

a2 : char[5]

0	1	2	3	4
0	0	0	0	0

Vargjet dydimensionale – Vazhdim

`char[][] word ==` a1

a1 : char[20][]

0	a2
1	null
2	null
...	...
19	null

a2 : char[5]

0	1	2	3	4
'T'	'u'	'n'	'g'	'!'

Vargjet dydimensionale – Vazhdim

`char[][] word ==` a1

a1 : char[20][]

0	a2
1	a3
2	null
...	...
19	null

a2 : char[5]

0	1	2	3	4
'T'	'u'	'n'	'g'	'!'

a3 : char[3]

0	1	2
'I'	'k'	'a'

Vargjet dydimensionale – Vazhdim

```
for ( int i = 0; i != count; i++ )
{ System.out.print("Gjatësia: " + word[i].length + ", fjala: ");
  for ( int j = 0; j != word[i].length; j++ )
  { System.out.print(word[i][j]); }
  System.out.println();
}
```

Case study: Loja e rebusit rrëshqitës

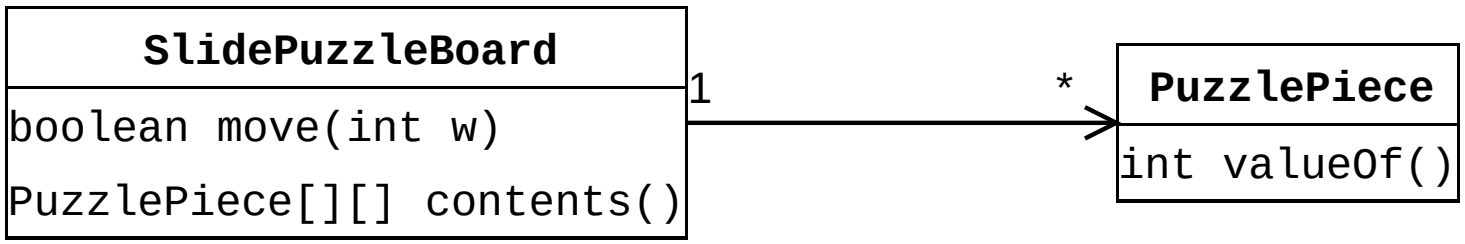


Figura 2. Arkitektura e modelit të rebusit rrëshqitës

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Definon një copëzë të lojës së rebusit rrëshqitës. */
public class PuzzlePiece
{ private int faceValue;
  /** Krijon copëzën.
   * @param value - vlera e cila paraqitet në copëzën */
  public PuzzlePiece(int value)
  { faceValue = value; }
  /** Kthen vlerën e copëzës. */
  public int valueOf()
  { return faceValue; }
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Modelon një rebus rrëshqitës. */
public class SlidePuzzleBoard
{ private int size;
  private PuzzlePiece[][] board;
  private int emptyRow;
  private int emptyCol;
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Konstrukton rebusin inicial me copëza
 * në renditje të anasjelltë.
 * @param s - madhësia e rebusit (s x s) */
public SlidePuzzleBoard(int s)
{ size = s;
  board = new PuzzlePiece[size][size];
  for ( int num = 1; num != size * size; num++ )
  { PuzzlePiece p = new PuzzlePiece(num);
    int row = num / size;
    int col = num % size;
    board[size - 1 - row][size - 1 - col] = p;
  }
  emptyRow = size - 1;
  emptyCol = size - 1;
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Kthen gjendjen vijuese të rebusit.
 * @return  matricë me adresa të copëzave */
public PuzzlePiece[][] contents()
{ PuzzlePiece[][] answer = new PuzzlePiece[size][size];
  for ( int i = 0; i != size; i++ )
  { for ( int j = 0; j != size; j++ )
    { answer[i][j] = board[i][j]; }
  }
  return answer;
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Lëviz një copëz në hapësirën e lirë nëse është e mundur.
 * @param w - vlera e copëzës që do të lëvizet
 * @return  suksesin e lëvizjes */
public boolean move(int w)
{ int NOT_FOUND = -1;
  int row = NOT_FOUND;
  int col = NOT_FOUND;
  if ( found(w, emptyRow - 1, emptyCol) )
  { row = emptyRow - 1;
    col = emptyCol;
  }
  else if ( found(w, emptyRow + 1, emptyCol) )
  { row = emptyRow + 1;
    col = emptyCol;
  }
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
  else if ( found(w, emptyRow, emptyCol - 1) )
  { row = emptyRow;
    col = emptyCol - 1;
  }
  else if ( found(w, emptyRow, emptyCol + 1) )
  { row = emptyRow;
    col = emptyCol + 1;
  }
  if ( row != NOT_FOUND )
  { board[emptyRow][emptyCol] = board[row][col];
    emptyRow = row;
    emptyCol = col;
    board[emptyRow][emptyCol] = null;
  }
  return row != NOT_FOUND;
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
  private boolean found(int v, int row, int col)
  { boolean answer = false;
    if ( row >= 0 && row < size && col >= 0 && col < size )
    { answer = ( board[row][col].valueOf() == v ); }
    return answer;
  }
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
import javax.swing.*;
/** Kontrollon lëvizjet e rebusit rrëshqitës. */
public class PuzzleController
{ private SlidePuzzleBoard board;
  private PuzzleWriter writer;
  /** Inicializon kontrolluesin.
   *  @param b - modeli, tabela e rebusit
   *  @param w - output view */
  public PuzzleController(SlidePuzzleBoard b, PuzzleWriter w)
  { board = b;
    writer = w;
  }
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
  /** Lejon shfrytëzuesin të luajë rebusin. */
  public void play()
  { while ( true )
    { writer.displayPuzzle();
      int i = new Integer
        (JOptionPane.showInputDialog("Lëvizja juaj:")).intValue();
      boolean ok = board.move(i);
      if ( !ok )
      { writer.printError("Lëvizje jokorrekte."); }
    }
  }
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Implementon një rebus rrëshqitës.
 *  Hyrja: varg numrash ndërmjet 1...15 */
public class SlidePuzzle
{ public static void main(String[] args)
  { int size = 4;
    SlidePuzzleBoard board = new SlidePuzzleBoard(size);
    PuzzleWriter writer = new PuzzleWriter(board, size);
    PuzzleController controller
      = new PuzzleController(board, writer);
    controller.play();
  }
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
import java.awt.*;
import javax.swing.*;
/** Afishon përmbajtjen e një rebusi rrëshqitës. */
public class PuzzleWriter extends JPanel
{ private SlidePuzzleBoard board;
  private int size;
  private int pieceSize = 30;
  private int panelWidth;
  private int panelHeight;
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Ndërton dritaren grafike.
 * @param b - rebusi që afishohet
 * @param s - madhësia e rebusit */
public PuzzleWriter(SlidePuzzleBoard b, int s)
{ board = b;
  size = s;
  panelWidth = pieceSize * size + 100;
  panelHeight = pieceSize * size + 100;
  JFrame frame = new JFrame();
  frame.getContentPane().add(this);
  frame.setTitle("Rebusi rrëshqitës");
  frame.setSize(panelWidth, panelHeight);
  frame.setVisible(true);
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
/** Vizaton copëzën p në pozitën i,j në dritaren. */
private void paintPiece(Graphics g, PuzzlePiece p, int i, int j)
{ int initialOffset = pieceSize;
  int xPos = initialOffset + (pieceSize * j);
  int yPos = initialOffset + (pieceSize * i);
  if ( p != null )
  { g.setColor(Color.white);
    g.fillRect(xPos, yPos, pieceSize, pieceSize);
    g.setColor(Color.black);
    g.drawRect(xPos, yPos, pieceSize, pieceSize);
    g.drawString(p.valueOf() + "", xPos + 10, yPos + 20);
  }
  else
  { g.setColor(Color.black);
    g.fillRect(xPos, yPos, pieceSize, pieceSize);
  }
}
```

Case study: Loja e rebusit rrëshqitës – Vazhdim

```
  /** Afishon rebusin në kornizën. */
  public void paintComponent(Graphics g)
  { g.setColor(Color.yellow);
    g.fillRect(0, 0, panelWidth, panelHeight);
    PuzzlePiece[][] r = board.contents();
    for ( int i = 0; i != size; i++ )
    { for ( int j = 0; j != size; j++ )
      { paintPiece(g, r[i][j], i, j); }
    }
  }
  /** Afishon gjendjen vijuese të rebusit. */
  public void displayPuzzle()
  { this.repaint(); }
  /** Afishon mesazh gabimi.
  * @param s - mesazhi i gabimit */
  public void printError(String s)
  { JOptionPane.showMessageDialog(null, "Gabim: " + s ); }
}
```

Case study: Loja e rebusit rrëshqitës

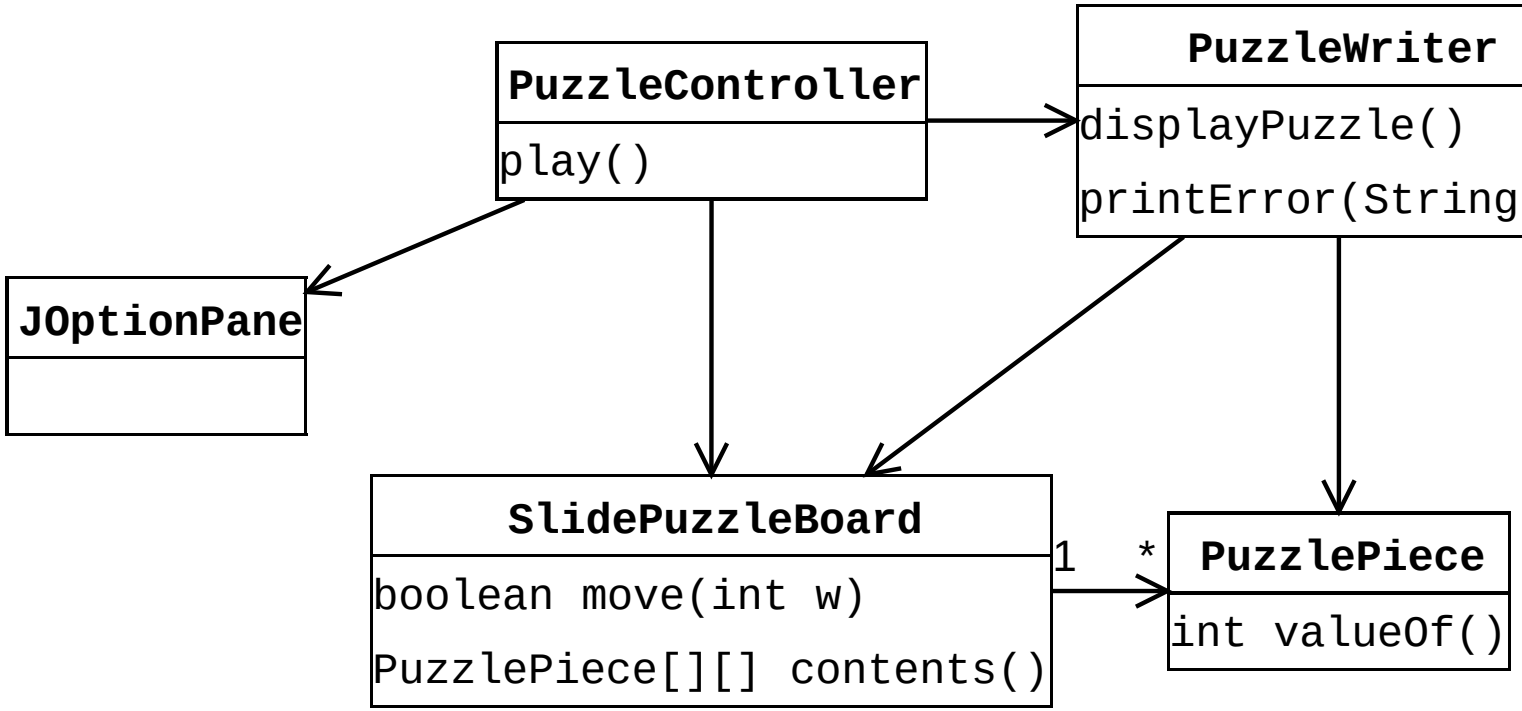


Figura 3. Diagrami i klasave të rebusit rrëshqitës

Testimi i programeve me vargje

```
/** Ua ndërron vendet vlerave në r[i] dhe r[i-1]. */
public void exchange(int[] r, int i)
{ int temp = r[i];
  r[i] = r[i - 1];
  r[i - 1] = temp;
}

int[] testArray = new int[10];
testArray[3] = 3;
exchange(testArray, 4);
```

Testimi i programeve me vargje – Vazhdim

Testet vijuese shkaktojnë `RunTimeException` të tipit `ArrayIndexOutOfBoundsException`:

```
int[] testArray = new int[10];
testArray[9] = 3;
exchange(testArray, 10);

int[] testArray = new int[10];
testArray[0] = 3;
exchange(testArray, 0);
```