Snake obhajoba 21.12.23 - 12:30

- Snake obhajoba 21.12.23 - 12:30
  - Aplikacia.java
    - atribúty, konštruktor, main(String[] args)
    - uvodneOkno()
    - nastavenieObtiaznosti()
    - vlastnaObtiaznost()
    - prevratenieRychlosti(int tempoHry)
    - prehralSi(Snake snake, Prekazka prekazka)
    - spustiHru(int pocetPrekazok, int rychlostHry)
    - restartujHru(Snake snake, Prekazka prekazka)
  - HraciaPlocha.java
    - atribúty & konštruktor
    - vykresli(Graphics g)
      - mriežka hracieho poľa
      - hracie pole šachovnicovom štýle
      - vykreslenie informácií o hre
      - kontroluje sa, či hra je pozastavená & vypísanie hlášok
  - Jedlo.java
    - umiestniJedlo(int sirkaPlchy, int vyskaPlochy, int velkostPolicka, Prekazka prekazky)
    - koliziaJedlaSPrekazkou(Prekazka prekazky)
    - vykresli(Graphics g, int velkostPolicka)
  - Manazer.java
    - keyPressed(KeyEvent e)
    - vykresliHada(Graphics g)
  - Policko.java
  - Prekazka.java
    - atribúty && konštruktor
    - generujNahodnePrekazky(int sirkaPlochy, int vyskaPlochy, int velkostPolicka, int pocetPrekazok)
    - koliziaHadaSPrekazkou(Policko hlavaHada)
    - vykresli (Graphics g, int velkostPolicka)
  - Smer.java
  - Snake.java
    - konštruktor Snake(int sirkaPlochy, int vyskaPlochy, int rychlostHry, int pocetPrekazok, int velkostPolicka)
    - paintComponeent(Graphics g)
    - vykresli (Graphics g)
    - umiestniJedlo()
    - pohyb()
      - kolízia hlavyHada a jedla
      - časť jedla sa presunie na jej predchádzajúcu pozíciu
      - kontrola kolízie s okrajmi hracej plochy
      - pohyb hlavyHada

        - kontrola kolízie hlavyHada s telom
        - kontrola kolízie hlavyHada s prekážkou
        - kontrola kolízie jedla s prekážkou
    - kolizia(Object objekt1, Object objekt2)
    - actionPerformed(ActionEvent e)
    - keyPressed (KeyEvent e)
    - generujNahodnePolicko()

Aplikacia.java

atribúty, konštruktor, main(String[] args)

```
private Zvuk zvuk;

public Aplikacia() {
    this.zvuk = new Zvuk();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        Aplikacia aplikacia = new Aplikacia();
        aplikacia.uvodneOkno();
    });
}
```

- Tento kód vytvára triedu `Aplikacia` s premennou `zvuk` typu `Zvuk`.
- V konštruktore sa inicializuje premenná `zvuk` novou inštanciou triedy `Zvuk`.
- Metóda `main` spustí aplikáciu v rozhraní Java Swing, vytvorí inštanciu `Aplikacia` a zavolá metódu `uvodneOkno()`.

uvodneOkno()

```
public void uvodneOkno() {
    JPanel panel = new JPanel();
    JLabel uvodnyText = new JLabel("<html><b>Inštrukcie hry:</b><br>"
            + "Na pohyb hada použite šípky alebo klávesy WASD.<br>"
            + "Klávesa ESC slúži na pozastavenie hry.<br>"
            + "<br>"
            + "V ďalšom kroku si vyberte preferovanú obtiažnosť hry.
</html>");
    panel.add(uvodnyText);

    Object[] options = {"Ďalej", "Koniec"};

    int vysledok = JOptionPane.showOptionDialog(
        null,
```

```
            panel,
            "Vitajte v hre Snake!",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.INFORMATION_MESSAGE,
            null,
            options,
            options[0]);

        if (vysledok == 0) {
            this.nastavenieObtiaznosti();
        } else if (vysledok == 1) {
            System.exit(0);
        }
    }
```

- Kód metódy `uvodneOkno()` v Jave vykonáva nasledujúce kroky:

1. **Vytvorenie komponentov okna:**
    - Vytvára nový panel (`JPanel`) a `JLabel` s HTML formátovaným textom, ktorý obsahuje inštrukcie hry. Tento text vysvetľuje ovládanie hada a používanie kláves na pohyb a pozastavenie hry.
2. **Vytvorenie dialógového okna (`JOptionPane`):**
    - Definuje pole možností ("Ďalej", "Koniec").
    - Používa `JOptionPane.showOptionDialog()` na vytvorenie dialógového okna s uvítacím textom a tlačidlami na výber ďalšieho kroku alebo ukončenie hry.
3. **Spracovanie výberu hráča:**
    - Získava výber hráča z dialógového okna.
    - Ak hráč vyberie "Ďalej" (hodnota `0`), volá metódu `nastavenieObtiaznosti()`.
    - Ak hráč vyberie "Koniec" (hodnota `1`), ukončuje program pomocou `System.exit(0)`.

Celkovo táto metóda vytvára uvítaciu obrazovku pre hru Snake s inštrukciami a umožňuje hráčovi vybrať medzi pokračovaním a ukončením hry.

nastavenieObtiaznosti()

```
public void nastavenieObtiaznosti() {
    String[] vyberObtiaznost = {"Vlastná", "Ťažká", "Stredná", "Ľahká"};
    int obtiaznosti = JOptionPane.showOptionDialog(null, "Vyberte si
obtiažnosť hry.",
                        "Nastavenie obtiažnosti",
JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE,
                        null, vyberObtiaznost, vyberObtiaznost[0]);

    switch (obtiaznosti) {
        case 0:
            // vlastna
```

```
                this.vlastnaObtiaznost();
                break;
            case 1:
                // tazka
                this.spustiHru(15, 60);
                break;
            case 2:
                // stredna
                this.spustiHru(10, 90);
                break;
            case 3:
                // lahka
                this.spustiHru(5, 120);
                break;
            default:
                System.exit(0);
        }
    }
```

Táto Java metóda `nastavenieObtiaznosti()` vykonáva nasledujúce úlohy:

1. **Definícia možností pre obtiažnosť:**
    - Vytvára pole reťazcov `vyberObtiaznost` obsahujúce možnosti pre hráča: "Vlastná", "Ťažká", "Stredná", "Ľahká".
2. **Zobrazenie dialógového okna s výberom obtiažnosti:**
    - Používa `JOptionPane.showOptionDialog()` na zobrazenie dialógového okna, v ktorom hráč vyberá obtiažnosť hry.
3. **Spracovanie výberu hráča pomocou `switch`:**
    - Na základe výberu hráča v dialógovom okne používa `switch` pre vetvenie na príslušnú časť kódu.
    - Ak hráč vyberie "Vlastná" (hodnota `0`), spúšťa metódu `vlastnaObtiaznost()`.
    - Pre prípady "Ťažká", "Stredná" a "Ľahká" spúšťa metódu `spustiHru()` s príslušnými parametrami.
    - V prípade nečakaného výberu (default) program sa ukončuje pomocou `System.exit(0)`.

Celkovo táto metóda umožňuje hráčovi zvoliť obtiažnosť hry a podľa toho sa spustí príslušná časť kódu.

vlastnaObtiaznost()

```
public void vlastnaObtiaznost() {
    JTextField pocetPrekazokPolicko = new JTextField();
    JTextField tempoHryPolicko = new JTextField();

    tempoHryPolicko.setForeground(Color.GRAY);
    tempoHryPolicko.setText("Zadaj 1 až 10");
    tempoHryPolicko.addFocusListener(new FocusAdapter() {
```

```
        //takto vyzera policko, ktore je aktivne
        @Override
        public void focusGained(FocusEvent e) {
            if (tempoHryPolicko.getText().equals("Zadaj 1 až 10")) {
                tempoHryPolicko.setText("");
                tempoHryPolicko.setForeground(Color.BLACK);
            }
        }

        //takto vyzera policko, ked je neaktivne
        @Override
        public void focusLost(FocusEvent e) {
            if (tempoHryPolicko.getText().isEmpty()) {
                tempoHryPolicko.setForeground(Color.GRAY);
                tempoHryPolicko.setText("Zadaj 1 až 10");
            }
        }
    });

    //samotne okno na vpisanie zelanych hodnot (pocet prekazok & rychlost
hry)
    JPanel panel = new JPanel(new GridLayout(2, 2));
    panel.add(new JLabel("Počet prekážok:"));
    panel.add(pocetPrekazokPolicko);
    panel.add(new JLabel("Tempo hry:"));
    panel.add(tempoHryPolicko);

    UIManager.put("OptionPane.okButtonText", "Hrať");
    UIManager.put("OptionPane.cancelButtonText", "Späť");

    int vysledok = JOptionPane.showConfirmDialog(null, panel, "Vlastná
obtiažnosť",
        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    if (vysledok == JOptionPane.OK_OPTION) {
        try {
            int pocetPrekazok =
Integer.parseInt(pocetPrekazokPolicko.getText());
            int tempoHry =
this.prevratenieRychlosti(Integer.parseInt(tempoHryPolicko.getText()));
            this.spustiHru(pocetPrekazok, tempoHry);
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Zadajte číselnú
hodnotu.", "Chyba", JOptionPane.ERROR_MESSAGE);
            this.vlastnaObtiaznost();
        }
    } else {
        this.nastavenieObtiaznosti();
    }
}
```

Táto Java metóda `vlastnaObtiaznost()` vykonáva nasledujúce úlohy:

1. **Vytvorenie textových polí pre vstup od hráča:**
    - Vytvára dve textové polia (`JTextField`) pre vstup od hráča: `pocetPrekazokPolicko` a `tempoHryPolicko`.
2. **Nastavenie šedého textu a obsahu pre `tempoHryPolicko`:**
    - Nastavuje šedú farbu textu a prednastavený text "Zadaj 1 až 10" do textového poľa `tempoHryPolicko`.
    - Implementuje metódu `FocusAdapter` na zmenu textu v poli pri získaní alebo stratí fókus.
3. **Vytvorenie panelu s komponentami na vstupe:**
    - Vytvára panel (`JPanel`) s mriežkovým rozložením 2x2 obsahujúci popisky a textové polia pre počet prekážok a tempo hry.
4. **Nastavenie textu tlačidiel v dialógovom okne:**
    - Nastavuje texty tlačidiel v dialógovom okne na "Hrať" a "Späť".
5. **Zobrazenie dialógového okna s potvrdením:**
    - Používa `JOptionPane.showConfirmDialog()` na zobrazenie dialógového okna s panelom pre vstup od hráča.
6. **Spracovanie výberu hráča:**
    - Ak hráč stlačí tlačidlo "Hrať", snaží sa získať hodnoty zo vstupných polí.
    - Ak sa hodnoty podarí získať, konvertuje ich na celočíselné hodnoty a volá metódu `spustiHru()` s týmito hodnotami.
    - Ak sa získanie hodnôt neúspešne skončí (napr. vstup nie je číslo), zobrazuje chybovú správu a znovu volá `vlastnaObtiaznost()`.
    - Ak hráč stlačí tlačidlo "Späť", vráti sa na obrazovku s výberom obtiažnosti volaním `nastavenieObtiaznosti()`.

Celkovo táto metóda umožňuje hráčovi nastaviť vlastnú obtiažnosť hry zadávaním počtu prekážok a tempa hry prostredníctvom dialógového okna.

prevratenieRychlosti(int tempoHry)

```
public int prevratenieRychlosti(int tempoHry) {
    int minRychlost = 25;
    int maxRychlost = 300;
    int prevratenaRychlost;

    if (tempoHry >= 10) {
        prevratenaRychlost = 1;
    } else {
        prevratenaRychlost = 120 - tempoHry * 12;
    }
    return Math.min(maxRychlost, Math.max(minRychlost,
```

```
prevratenaRychlost));
}
```

Tento kód implementuje metódu s názvom `prevratenieRychlosti(int tempoHry)`, ktorá má za úlohu transformovať zadané tempo hry na rýchlosť používanú v hre. Tu je popis toho, ako táto metóda funguje:

1. **Definícia hodnôt pre rýchlosť:**
   - Vytvára tri premenné: `minRychlost` s hodnotou 25, `maxRychlost` s hodnotou 300 a `prevratenaRychlost`, ktorá bude slúžiť na uchovávanie výslednej transformovanej rýchlosti.
2. **Podmienkový príkaz:**
   - Kontroluje, či zadané tempo hry (`tempoHry`) je väčšie alebo rovnaké ako 10.
   - Ak áno, nastavuje `prevratenaRychlost` na hodnotu 1.
3. **Inak vetva:**
   - Ak je tempo hry menšie ako 10, vypočítava `prevratenaRychlost` podľa vzorca: `120 - tempoHry * 12`.
4. **Ohraničenie rýchlosti:**
   - Používa `Math.min` a `Math.max` na ohraničenie `prevratenaRychlost` medzi hodnotami `minRychlost` a `maxRychlost`.
   - Týmto sa zabezpečuje, aby rýchlosť bola v určenom rozsahu (25 až 300).
5. **Návratová hodnota:**
   - Vracia transformovanú a ohraničenú rýchlosť (`prevratenaRychlost`) ako výsledok metódy.

Celkovo táto metóda prevráti (transformuje) zadané tempo hry na rýchlosť, ktorá je následne ohraničená v stanovenom rozsahu.

prehralSi(Snake snake, Prekazka prekazka)

```
public void prehralSi(Snake snake, Prekazka prekazky) {
    this.zvuk.zvukKoncaHry();
    snake.getHernyCyklus().stop();
    Object[] options = {"Opakovať", "Skončiť", "Zmeniť obtiažnosť"};
    int odpoved = JOptionPane.showOptionDialog(snake, "Prehral si! Čo
chceš urobiť?", "Koniec hry",
                    JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE, null, options, options[0]);
    if (odpoved == JOptionPane.YES_OPTION) {
        this.restartujHru(snake, prekazky);
    } else if (odpoved == JOptionPane.CANCEL_OPTION) {
        //uzatvorenie predosleho herneho okna pri zmene obtiaznosti
        JFrame okno = (JFrame)SwingUtilities.getRoot(snake);
        okno.dispose();
        this.nastavenieObtiaznosti();
```

```
        } else {
            System.exit(0);
        }
    }
```

Tento kód implementuje metódu s názvom `prehralSi(Snake snake, Prekazka prekazky)`, ktorá sa volá v prípade, že hráč prehral hru. Tu je vysvetlenie toho, ako táto metóda funguje:

1. **Zvuk konca hry:**
    - Spúšťa zvuk konca hry pomocou metódy `zvukKoncaHry()` z objektu `zvuk`.
2. **Zastavenie herného cyklu:**
    - Volá metódu `stop()` na hernom cykle hada, ktorý je predaný ako parameter (`snake`).
3. **Zobrazenie dialógového okna s možnosťami:**
    - Vytvára pole možností pre hráča: "Opakovať", "Skončiť", "Zmeniť obtiažnosť".
    - Používa `JOptionPane.showOptionDialog()` na zobrazenie dialógového okna s otázkou, čo hráč chce urobiť po prehre.
4. **Spracovanie výberu hráča:**
    - Ak hráč vyberie "Opakovať" (hodnota `JOptionPane.YES_OPTION`), volá metódu `restartujHru()` s objektami `snake` a `prekazky` ako parametrami.
    - Ak hráč vyberie "Zmeniť obtiažnosť" (hodnota `JOptionPane.CANCEL_OPTION`), zatvára predchádzajúce herné okno a volá metódu `nastavenieObtiaznosti()`.
    - Ak hráč vyberie "Skončiť" alebo zatvorí dialógové okno, program sa ukončuje pomocou `System.exit(0)`.

Celkovo táto metóda reaguje na prehru hráča, spúšťa zvuk, zastavuje herný cyklus a ponúka mu možnosti ako reagovať na prehru (opakovať, skončiť alebo zmeniť obtiažnosť).

spustiHru(int pocetPrekazok, int rychlostHry)

```
public void spustiHru(int pocetPrekazok, int rychlostHry) {
    int sirkaPlochy = 1400;
    int vyskaPlochy = 800;
    int velkostPolicka = 40;

    Prekazka prekazka = new Prekazka(sirkaPlochy, vyskaPlochy,
velkostPolicka, pocetPrekazok);
    prekazka.generujNahodnePrekazky(sirkaPlochy, vyskaPlochy, rychlostHry,
pocetPrekazok);

    JFrame okno = new JFrame("Snake - Semestrálna práca - Filip Klein");
    okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Snake snake = new Snake(sirkaPlochy, vyskaPlochy, rychlostHry,
pocetPrekazok, velkostPolicka);
    okno.add(snake);
```

```
        okno.pack();
        okno.setLocationRelativeTo(null);
        okno.setResizable(false);
        okno.setVisible(true);
    }
```

Tento kód implementuje metódu s názvom `spustiHru(int pocetPrekazok, int rychlostHry)`, ktorá sa stará o inicializáciu a spustenie hry Snake s určitými parametrami. Tu je vysvetlenie toho, ako táto metóda funguje:

1. **Definícia parametrov a veľkosti plochy:**
    - Vytvára premenné `sirkaPlochy` a `vyskaPlochy` pre šírku a výšku herného okna, a `velkostPolicka` pre veľkosť jedného políčka na hracej ploche.
2. **Vytvorenie prekážky:**
    - Inicializuje objekt `Prekazka` s danými parametrami (šírka, výška, veľkosť políčka, počet prekážok).
    - Generuje náhodné prekážky pomocou metódy `generujNahodnePrekazky()`.
3. **Vytvorenie herného okna (`JFrame`):**
    - Inicializuje objekt `JFrame` s názvom "Snake - Semestrálna práca - Filip Klein".
    - Nastavuje akciu ukončenia programu na zavretie tohto okna.
4. **Vytvorenie hada (`Snake`):**
    - Inicializuje objekt `Snake` s danými parametrami (šírka, výška, rýchlosť, počet prekážok, veľkosť políčka).
5. **Pridanie hada do herného okna a zobrazenie:**
    - Pridáva objekt `Snake` do herného okna (`JFrame`).
    - Balíčky (pack) herné okno, umiestňuje ho na stred obrazovky, zakazuje zmenu veľkosti a nastavuje jeho viditeľnosť.

Celkovo táto metóda inicializuje herné prostredie, vytvára objekty prekážok a hada, a spúšťa hru v novom hernom okne.

restartujHru(Snake snake, Prekazka prekazka)

```
public void restartujHru(Snake snake, Prekazka prekazky) {
    snake.setHlavaHada(snake.generujNahodnePolicko());
    snake.getTeloHada().clear();
    snake.umiestniJedlo();

    prekazky.getPrekazky().clear();
    prekazky.generujNahodnePrekazky(snake.getSirkaPlochy(),
snake.getVyskaPlochy(), snake.getVelkostPolicka(),
snake.getPocetPrekazok());
```

```
    if (prekazky.getPrekazky().isEmpty()) {
        prekazky.generujNahodnePrekazky(snake.getSirkaPlochy(),
snake.getVyskaPlochy(), snake.getVelkostPolicka(),
snake.getPocetPrekazok());
    }

    snake.setKoniecHry(false);
    snake.setHraPozastavena(false);
    snake.getHernyCyklus().start();

    snake.setRychlostX(0);
    snake.setRychlostY(0);
    snake.setCasZastaveny(false);
    snake.setCasOdStartu(System.currentTimeMillis());
}
```

Tento kód implementuje metódu s názvom `restartujHru(Snake snake, Prekazka prekazky)`, ktorá sa stará o reštartovanie hry po prehre hráča. Tu je vysvetlenie toho, ako táto metóda funguje:

1. **Nastavenie nových hodnôt pre hada:**
   - Nastavuje novú polohu hlavy hada volaním metódy `generujNahodnePolicko()` na objekte `snake`.
   - Vyprázdňuje telo hada pomocou `clear()` na získanom liste teloHada.
   - Umisťuje nové jedlo na hraciu plochu volaním `umiestniJedlo()` na objekte `snake`.
2. **Nastavenie nových hodnôt pre prekážky:**
   - Vyprázdňuje existujúce prekážky pomocou `clear()` na získanom liste prekazky.
   - Generuje nové náhodné prekážky volaním `generujNahodnePrekazky()` s aktuálnymi parametrami na objekte `prekazky`.
3. **Overenie existencie prekážok:**
   - Ak sú nové prekážky prázdne (žiadne sa nepodarilo vygenerovať), opätovne volá metódu `generujNahodnePrekazky()` na objekte `prekazky`.
4. **Nastavenie hodnôt pre reštart hry:**
   - Nastavuje rôzne hodnoty objektu `snake` na pôvodné stavy, vrátane zastavenia hry, resetovania rýchlostí, a spustenia herného cyklu.

Celkovo táto metóda umožňuje reštartovať hru po prehre hráča s novým postavením hada, novými prekážkami a nastavením herných premenných.

HraciaPlocha.java

atribúty & konštruktor

```
private int sirkaPlochy;
private int vyskaPlochy;
private int velkostPolicka;
private Snake snake;

public HraciaPlocha(int sirkaPlochy, int vyskaPlochy, int velkostPolicka,
Snake snake) {
    this.sirkaPlochy = sirkaPlochy;
    this.vyskaPlochy = vyskaPlochy;
    this.velkostPolicka = velkostPolicka;
    this.snake = snake;
}
```

Tento kód definuje triedu s názvom `HraciaPlocha`, ktorá má niekoľko atribútov a konštruktor inicializujúci tieto atribúty. Tu je vysvetlenie kódu:

1. **Atribúty triedy:**
    - `sirkaPlochy`: Atribút, ktorý uchováva šírku hracej plochy.
    - `vyskaPlochy`: Atribút, ktorý uchováva výšku hracej plochy.
    - `velkostPolicka`: Atribút, ktorý reprezentuje veľkosť jedného políčka na hracej ploche.
    - `snake`: Atribút, ktorý predstavuje inštanciu triedy `Snake` a je použitý na manipuláciu s hadom vo vnútri tejto hracej plochy.
2. **Konštruktor triedy:**
    - Konštruktor má rovnaký názov ako trieda a slúži na inicializáciu atribútov triedy.
    - Parametre konštruktora sú šírka (`sirkaPlochy`), výška (`vyskaPlochy`), veľkosť políčka (`velkostPolicka`) a inštancia triedy `Snake` (`snake`).
    - Priradzuje hodnoty parametrov k príslušným atribútom triedy.

Takýto konštruktor umožňuje vytvoriť inštanciu triedy `HraciaPlocha` s definovanými rozmiermi hracej plochy, veľkosťou políčka a priradeným hadom.

vykresli(Graphics g)

mriežka hracieho poľa

```
for (int i = 0; i < this.sirkaPlochy / velkostPolicka; i++) {
    g.setColor(new Color(245, 205, 167));
    g.drawLine(i * velkostPolicka, 0, i * velkostPolicka,
this.vyskaPlochy);
    g.drawLine(0, i * velkostPolicka, this.sirkaPlochy, i *
velkostPolicka);
}
```

Tento kód obsahuje cyklus `for`, ktorý kreslí mriežku na hracej ploche. Tu je vysvetlenie toho, ako tento kód funguje:

1. **Iniciácia cyklu:**
   - `for (int i = 0; i < this.sirkaPlochy / velkostPolicka; i++)`: Inicializuje premennú `i` na 0. Cyklus sa opakuje, pokiaľ `i` je menšie ako šírka plochy delená veľkosťou políčka.
2. **Nastavenie farby:**
   - `g.setColor(new Color(245, 205, 167))`: Nastavuje farbu kreslenia na svetložltú pomocou objektu `Color`. Táto farba bude použitá pre čiary mriežky.
3. **Kreslenie vertikálnych a horizontálnych čiar:**
   - `g.drawLine(i * velkostPolicka, 0, i * velkostPolicka, this.vyskaPlochy)`: Kreslí vertikálne čiary mriežky od hora až na spodok hracej plochy na pozícii `i * velkostPolicka`.
   - `g.drawLine(0, i * velkostPolicka, this.sirkaPlochy, i * velkostPolicka)`: Kreslí horizontálne čiary mriežky zľava doprava na pozícii `i * velkostPolicka`.

Celkovo tento cyklus vytvára efekt mriežky na hracej ploche s vertikálnymi a horizontálnymi čiarami, ktoré majú svetložltú farbu.

hracie pole šachovnicovom štýle

```
for (int i = 0; i < this.sirkaPlochy / velkostPolicka; i++) {
    for (int j = 0; j < this.vyskaPlochy / velkostPolicka; j++) {
        if ((i + j) % 2 == 0) {
            g.setColor(new Color(255, 218, 142));
        } else {
            g.setColor(new Color(250, 213, 138));
        }
        g.fillRect(i * this.velkostPolicka, j * this.velkostPolicka,
this.velkostPolicka, this.velkostPolicka);
    }
}
```

Tento kód vytvára šachovnicový vzor na hracej ploche pomocou dvoch vnorených cyklov `for` a vykresľuje obdĺžniky s rôznymi farbami. Tu je vysvetlenie toho, ako tento kód funguje:

1. **Dva vnorené cykly:**
   - `for (int i = 0; i < this.sirkaPlochy / velkostPolicka; i++)`: Vnútorný cyklus prechádza cez stĺpce hracej plochy.
   - `for (int j = 0; j < this.vyskaPlochy / velkostPolicka; j++)`: Vonkajší cyklus prechádza cez riadky hracej plochy.
2. **Podmienkový príkaz pre farbu obdĺžniku:**

- `if ((i + j) % 2 == 0)`: Ak sú súčet `i` a `j` párny, nastaví farbu obdĺžniku na svetlejší odtieň.
- `else`: V opačnom prípade nastaví farbu obdĺžniku na tmavší odtieň.

3. **Nastavenie farby a vykreslenie obdĺžnikov:**

- `g.setColor(new Color(255, 218, 142))` alebo `g.setColor(new Color(250, 213, 138))`: Nastavuje farbu kreslenia na svetlú alebo tmavú farbu v závislosti od podmienky.
- `g.fillRect(i * this.velkostPolicka, j * this.velkostPolicka, this.velkostPolicka, this.velkostPolicka)`: Kreslí obdĺžnik na hracej ploche na pozícii `[i * velkostPolicka, j * velkostPolicka]` s danou veľkosťou.

Celkovo tento kód vytvára šachovnicový vzor na hracej ploche pomocou svetlých a tmavých obdĺžnikov.

vykreslenie informácií o hre

```
String bodovyText = String.format("Dĺžka hada: %d | Čas v hre: %d s",
this.snake.getTeloHada().size(), this.snake.getCasHry() / 1000);
long casTeraz = System.currentTimeMillis();

g.setColor(Color.blue);
g.setFont(new Font("Arial", Font.BOLD, 16));
g.drawString(bodovyText, this.velkostPolicka - 16, this.velkostPolicka);

//zaistuje to, ze ak hru pausneme, tak sa cas takisto zastavi... pri
opatovnom spusteni hry bude cas rovnaky, ako pred pozastavenim hry
if (this.snake.getHraPozastavena() && !this.snake.getCasZastaveny()) {
    this.snake.setCasKedyPozastavena(casTeraz);
    this.snake.setCasZastaveny(true);
} else if (!this.snake.getHraPozastavena() &&
this.snake.getCasZastaveny()) {
    long casOdSpusteniaPoPozastaveni = (casTeraz -
this.snake.getCasKedyPozastavena());
    if (casOdSpusteniaPoPozastaveni > 0) {
        this.snake.setCasOdStartu(this.snake.getCasOdStartu() +
casOdSpusteniaPoPozastaveni);
    }

    this.snake.setCasZastaveny(false);
}
```

1. **Formátovanie bodového textu:**

- `String.format("Dĺžka hada: %d | Čas v hre: %d s", this.snake.getTeloHada().size(), this.snake.getCasHry() / 1000)`: Vytvára textový reťazec s informáciami o dĺžke hada a čase strávenom v hre.

2. **Nastavenie farby a fontu pre text:**

- `g.setColor(Color.blue)`: Nastavuje farbu textu na modrú.

- `g.setFont(new Font("Arial", Font.BOLD, 16))`: Nastavuje font textu na Arial, tučný, veľkosť 16 bodov.

3. **Vykreslenie bodového textu:**
   - `g.drawString(bodovyText, this.velkostPolicka - 16, this.velkostPolicka)`: Vykresľuje bodový text na hracej ploche na špecifikovanej pozícii.

4. **Správa času:**
   - Ak je hra pozastavená a čas nie je zastavený, nastaví aktuálny čas ako čas pozastavenia a zastaví čas.
   - Ak nie je hra pozastavená a čas je zastavený, pokračuje v čase od momentu, kedy bola hra zastavená. Nastavuje čas od spustenia hry na hodnotu pred zastavením a spúšťa čas.

Celkovo tento kód riadi zobrazenie informácií o stave hry a riadi aj čas v prípade, že je hra pozastavená.

kontroluje sa, či hra je pozastavená & vypísanie hlášok

```
if (!this.snake.getHraPozastavena()) {
    this.snake.setCasHry(casTeraz - this.snake.getCasOdStartu());
    g.drawString(bodovyText, this.velkostPolicka - 16,
this.velkostPolicka);
}

// vypisanie hlasok
if (this.snake.getKoniecHry()) {
    g.setColor(Color.red);
    g.drawString("Koniec hry!", this.velkostPolicka - 16,
this.velkostPolicka * 2);
} else if (this.snake.getHraPozastavena()) {
    g.setColor(Color.black);
    g.drawString("Hra je pozastavená, stlačením klávesy ESC sa vrátite do
hry.", this.velkostPolicka - 16, this.velkostPolicka * 2);
}
```

Tento kód zabezpečuje zobrazenie dodatočných informácií o stave hry a eventuálne vypisuje hlášky, napríklad o konci hry alebo o tom, že hra je pozastavená. Tu je vysvetlenie toho, ako tento kód funguje:

1. **Aktualizácia času hry a výpis bodového textu:**
   - `if (!this.snake.getHraPozastavena())`: Kontroluje, či hra nie je pozastavená. Ak nie je, aktualizuje čas hry a vypisuje bodový text.

2. **Vypisovanie hlášok:**
   - `if (this.snake.getKoniecHry())`: Ak je nastavená podmienka konca hry, vypíše sa hláška "Koniec hry!" v červenej farbe.
   - `else if (this.snake.getHraPozastavena())`: Ak je hra pozastavená, vypíše sa hláška o tom, že hra je pozastavená, a uvádza sa, že klávesa ESC sa dá použiť na návrat do hry.

Celkovo tento kód pridáva do hry dodatočné vizuálne informácie, ako napríklad texty o stave hry a rôzne hlášky pre hráča.

Jedlo.java

umiestniJedlo(int sirkaPlchy, int vyskaPlochy, int velkostPolicka, Prekazka prekazky)

```
public void umiestniJedlo(int sirkaPlochy, int vyskaPlochy, int
velkostPolicka, Prekazka prekazky) {
    Random nahodne = new Random();
    do {
        int x = nahodne.nextInt(sirkaPlochy / velkostPolicka);
        int y = nahodne.nextInt(vyskaPlochy / velkostPolicka);
        this.jedlo = new Policko(x, y);
    } while (this.koliziaJedlaSPrekazkou(prekazky));
}
```

Táto metóda `umiestniJedlo` slúži na umiestnenie jedla na hracej ploche, s respektovaním prípadných prekážok. Tu je vysvetlenie toho, ako kód funguje:

1. **Generovanie náhodných súradníc:**
    - `int x = nahodne.nextInt(sirkaPlochy / velkostPolicka);`: Generuje náhodnú hodnotu `x` medzi 0 a `sirkaPlochy / velkostPolicka`.
    - `int y = nahodne.nextInt(vyskaPlochy / velkostPolicka);`: Generuje náhodnú hodnotu `y` medzi 0 a `vyskaPlochy / velkostPolicka`.
2. **Vytvorenie nového políčka pre jedlo:**
    - `this.jedlo = new Policko(x, y);`: Vytvára novú inštanciu triedy `Policko` s náhodnými súradnicami, ktorá predstavuje umiestnenie jedla.
3. **Overenie kolízie s prekážkami:**
    - `while (this.koliziaJedlaSPrekazkou(prekazky));`: Používa metódu `koliziaJedlaSPrekazkou` na overenie, či nové jedlo nie je umiestnené na políčku obsadenom prekážkou. Ak áno, generuje sa nové jedlo a overuje sa znova.

Celkovo táto metóda zabezpečuje, aby jedlo bolo náhodne umiestnené na hracej ploche a aby sa nevyskytovalo na políčku obsadenom prekážkou.

koliziaJedlaSPrekazkou(Prekazka prekazky)

```
public boolean koliziaJedlaSPrekazkou(Prekazka prekazky) {
    for (Policko prekazka : prekazky.getPrekazky()) {
        if (this.jedlo.getX() == prekazka.getX() && this.jedlo.getY() ==
prekazka.getY()) {
            return true;
```

```
            }
        }
        return false;
    }
```

Metóda `koliziaJedlaSPrekazkou` slúži na overenie, či súradnice aktuálneho jedla kolidujú s nejakou prekážkou na hracej ploche. Metóda prechádza všetky políčka v zozname prekážok a porovnáva súradnice jedla s každou prekážkou. Ak nájde zhodu, vráti `true`, čo indikuje, že dochádza k kolízii jedla s prekážkou. V opačnom prípade vráti `false`.

vykresli(Graphics g, int velkostPolicka)

```
public void vykresli(Graphics g, int velkostPolicka) {
    try {
        this.jablkoImage =
ImageIO.read(getClass().getResource("foto/jablko.png"));
        g.drawImage(this.jablkoImage, this.jedlo.getX() * velkostPolicka,
this.jedlo.getY() * velkostPolicka, velkostPolicka, velkostPolicka, null);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Metóda `vykresli` slúži na vykreslenie grafického obrázku jablka na hracej ploche. Metóda využíva `ImageIO` na načítanie obrázku jablka z umiestnenia zdrojového súboru. Následne sa tento obrázok vykreslí na hraciu plochu na súradnice, kde sa momentálne nachádza jedlo. V prípade, že načítanie obrázku zlyhá, metóda vypíše informácie o výnimke na konzolu pomocou `e.printStackTrace()`.

Výnimka `IOException` sa tu môže vyskytnúť v prípade, že načítanie obrázku z nejakej príčiny zlyhá.

Manazer.java

keyPressed(KeyEvent e)

```
public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
        this.snake.setHraPozastavena(!this.snake.getHraPozastavena());
    } else if (!this.snake.getHraPozastavena()) {
        Smer novyPohyb = null;
        switch (e.getKeyCode()) {
            case KeyEvent.VK_UP:
            case KeyEvent.VK_W:
                if (this.snake.getRychlostY() != 1) {
                    novyPohyb = Smer.HORE;
                }
                break;
            case KeyEvent.VK_DOWN:
```

```
            case KeyEvent.VK_S:
                if (this.snake.getRychlostY() != -1) {
                    novyPohyb = Smer.DOLE;
                }
                break;
            case KeyEvent.VK_LEFT:
            case KeyEvent.VK_A:
                if (this.snake.getRychlostX() != 1) {
                    novyPohyb = Smer.DOLAVA;
                }
                break;
            case KeyEvent.VK_RIGHT:
            case KeyEvent.VK_D:
                if (this.snake.getRychlostX() != -1) {
                    novyPohyb = Smer.DOPRAVA;
                }
                break;
        }

        if (novyPohyb != null) {
            this.snake.setRychlostX(novyPohyb.getRychlostX());
            this.snake.setRychlostY(novyPohyb.getRychlostY());
        }
    }
}
```

Metóda `keyPressed` sa spúšťa pri stlačení klávesy a zabezpečuje riadenie pohybu hada a pozastavenie hry. Tu je vysvetlenie toho, ako táto metóda funguje:

1. **Pozastavenie hry stlačením klávesy ESC:**
   - `if (e.getKeyCode() == KeyEvent.VK_ESCAPE)`: Kontroluje, či bol stlačený kláves ESC.
   - `this.snake.setHraPozastavena(!this.snake.getHraPozastavena())`: Ak áno, prepína stav pozastavenia hry.

2. **Ovládanie pohybu hada:**
   - `else if (!this.snake.getHraPozastavena())`: Ak hra nie je pozastavená, vykonáva sa ovládanie pohybu hada.
   - `Smer novyPohyb = null;`: Inicializuje sa premenná pre nový smer pohybu hada.
   - **Rozpoznávanie stlačených kláves:**
     - `switch (e.getKeyCode())`: Používa switch pre rozpoznávanie stlačených kláves.
       - V prípade stlačenia šípok alebo kláves W, A, S, D, nastavuje sa nový smer pohybu hada podľa príslušnej klávesy.
   - **Kontrola platnosti nového smeru:**
     - `if (novyPohyb != null)`: Kontroluje, či bol nastavený platný nový smer.

- `this.snake.setRychlostX(novyPohyb.getRychlostX())`: Nastavuje novú rýchlosť hada podľa nového smeru v osi X.
- `this.snake.setRychlostY(novyPohyb.getRychlostY())`: Nastavuje novú rýchlosť hada podľa nového smeru v osi Y.

Týmto spôsobom metóda `keyPressed` riadi pohyb hada a pozastavenie hry v závislosti od stlačených kláves.

vykresliHada(Graphics g)

```
public void vykresliHada(Graphics g) {
    try {
        int velkostPolicka = this.snake.getVelkostPolicka();
        Policko hlavaHada = this.snake.getHlavaHada();

        if (this.snake.getRychlostY() == -1) {
            // pohyb hore
            this.hadikHore =
ImageIO.read(getClass().getResource("foto/hadikHore.png"));
            g.drawImage(this.hadikHore, hlavaHada.getX() * velkostPolicka,
hlavaHada.getY() * velkostPolicka, velkostPolicka, velkostPolicka, null);
        } else if (this.snake.getRychlostY() == 1) {
            // pohyb dole
            this.hadikDole =
ImageIO.read(getClass().getResource("foto/hadikDole.png"));
            g.drawImage(this.hadikDole, hlavaHada.getX() * velkostPolicka,
hlavaHada.getY() * velkostPolicka, velkostPolicka, velkostPolicka, null);
        } else if (this.snake.getRychlostX() == -1) {
            // pohyb vľavo
            this.hadikVlavo =
ImageIO.read(getClass().getResource("foto/hadikVlavo.png"));
            g.drawImage(this.hadikVlavo, hlavaHada.getX() *
velkostPolicka, hlavaHada.getY() * velkostPolicka, velkostPolicka,
velkostPolicka, null);
        } else if (this.snake.getRychlostX() == 1) {
            // pohyb vpravo
            this.hadikVpravo =
ImageIO.read(getClass().getResource("foto/hadikVpravo.png"));
            g.drawImage(this.hadikVpravo, hlavaHada.getX() *
velkostPolicka, hlavaHada.getY() * velkostPolicka, velkostPolicka,
velkostPolicka, null);
        } else {
            //had na zaciatku hry bude otoceny hore
            this.hadikHore =
ImageIO.read(getClass().getResource("foto/hadikHore.png"));
            g.drawImage(this.hadikHore, hlavaHada.getX() * velkostPolicka,
hlavaHada.getY() * velkostPolicka, velkostPolicka, velkostPolicka, null);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Metóda `vykresliHada` sa používa na vykreslenie grafického obrázku hada na hracej ploche. Tu je vysvetlenie toho, ako táto metóda funguje:

Táto metóda načíta obrázky hada pre rôzne smery pohybu (hore, dole, vľavo, vpravo) a potom vykreslí obrázok hlavy hada na hraciu plochu v závislosti od smeru pohybu. Ak nie je určený žiadny smer pohybu (čo sa môže stať na začiatku hry), použije sa obrázok hlavy hada smerujúcej hore. V prípade, že načítanie obrázkov zlyhá, informácie o výnimke sa vypíšu na konzolu pomocou `e.printStackTrace()`.

Policko.java

```
public class Policko {
    private int x;
    private int y;

    public Policko(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return this.x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return this.y;
    }

    public void setY(int y) {
        this.y = y;
    }
}
```

`Policko` je jednoduchá trieda, ktorá predstavuje jedno políčko na hracej ploche. Každé políčko má svoje súradnice `x` a `y`, a obsahuje metódy na získanie a nastavenie týchto súradníc. Táto trieda slúži na reprezentáciu polohy objektov (napr. hada, jedla, prekážok) na hracej ploche.

Prekazka.java

atribúty && konštruktor

```
private ArrayList<Policko> prekazky = new ArrayList<>();
private Policko hlavaHada;
private int sirkaPlochy;
private int vyskaPlochy;
private int velkostPolicka;
private int pocetPrekazok;

public Prekazka(int sirkaPlochy, int vyskaPlochy, int velkostPolicka, int
pocetPrekazok){
    this.sirkaPlochy = sirkaPlochy;
    this.vyskaPlochy = vyskaPlochy;
    this.velkostPolicka = velkostPolicka;
    this.pocetPrekazok = pocetPrekazok;
    this.prekazky = new ArrayList<>();
    this.generujNahodnePrekazky(this.sirkaPlochy, this.vyskaPlochy,
this.velkostPolicka, this.pocetPrekazok);
}
```

Konštruktor triedy `Prekazka` inicializuje tieto atribúty na základe poskytnutých parametrov. Vytvára sa nový zoznam prekážok a následne sa generujú náhodné prekážky pomocou metódy `generujNahodnePrekazky`.

generujNahodnePrekazky(int sirkaPlochy, int vyskaPlochy, int velkostPolicka, int pocetPrekazok)

```
public void generujNahodnePrekazky(int sirkaPlochy, int vyskaPlochy, int
velkostPolicka, int pocetPrekazok) {
    this.prekazky.clear();
    Random nahodne = new Random();
    for (int i = 0; i < pocetPrekazok; i++) {
        int x;
        int y;
        do {
            x = nahodne.nextInt(sirkaPlochy / velkostPolicka);
            y = nahodne.nextInt(vyskaPlochy / velkostPolicka);
        } while (this.koliziaHadaSPrekazkou(this.hlavaHada));

        this.prekazky.add(new Policko(x, y));
    }
}
```

Metóda `generujNahodnePrekazky` slúži na náhodnú generáciu prekážok na hracej ploche.

V tejto metóde sa postupne generuje požadovaný počet prekážok. Pri generovaní sa používa náhodný generátor na získanie súradníc `x` a `y`. Následne sa skontroluje, či nová prekážka nekolízie s hlavou hada pomocou metódy `koliziaHadaSPrekazkou`. Ak áno, generuje sa nová prekážka, kým nebudú súradnice unikátne. Nakoniec sa nová prekážka pridáva do zoznamu prekazky. Týmto spôsobom sa vytvárajú náhodné a nekolíziu s hadom prekážky na hracej ploche.

koliziaHadaSPrekazkou(Policko hlavaHada)

```
public boolean koliziaHadaSPrekazkou(Policko hlavaHada) {
    for (Policko prekazkaPolicko : this.prekazky) {
        if (Snake.kolizia(hlavaHada, prekazkaPolicko)) {
            return true;
        }
    }
    return false;
}
```

Metóda `koliziaHadaSPrekazkou` slúži na skontrolovanie, či hlava hada koliduje s niektorou z prekážok na hracej ploche.

V tejto metóde sa prechádzajú všetky políčka prekážok v zozname `prekazky`. Pre každú prekážku sa volá metóda `Snake.kolizia(hlavaHada, prekazkaPolicko)`, ktorá zisťuje, či dochádza ku kolízii medzi hlavou hada a aktuálnou prekážkou. Ak áno, metóda vráti `true`, čo znamená, že kolízia bola zistená. Ak žiadna kolízia nebol zistená pre žiadnu prekážku, vráti sa `false`.

vykresli (Graphics g, int velkostPolicka)

```
public void vykresli(Graphics g, int velkostPolicka) {
    for (Policko prekazkaPolicko : this.prekazky) {
        g.setColor(Color.black);
        g.fill3DRect(prekazkaPolicko.getX() * velkostPolicka,
prekazkaPolicko.getY() * velkostPolicka, velkostPolicka, velkostPolicka,
true);
    }
}
```

Metóda `vykresli` slúži na vykreslenie prekážok na hracej ploche. Pre každú prekážku v zozname `prekazky` vykreslí čierny 3D obdĺžnik na príslušných súradniciach. Táto metóda slúži na vizuálne zobrazenie prekážok v hernom prostredí.

Smer.java

```
public enum Smer {
    HORE(0, -1),
    DOLE(0, 1),
    DOLAVA(-1, 0),
    DOPRAVA(1, 0);

    private int rychlostX;
    private int rychlostY;

    Smer (int rychlostX, int rychlostY) {
```

```
        this.rychlostX = rychlostX;
        this.rychlostY = rychlostY;
    }

    public int getRychlostX() {
        return this.rychlostX;
    }

    public int getRychlostY() {
        return this.rychlostY;
    }
}
```

Definícia enumu `Smer` obsahuje štyri hodnoty reprezentujúce základné smery pohybu:

- `HORE`: Predstavuje pohyb smerom nahor.
- `DOLE`: Predstavuje pohyb smerom nadol.
- `DOLAVA`: Predstavuje pohyb smerom doľava.
- `DOPRAVA`: Predstavuje pohyb smerom doprava.

Kód enumu obsahuje dva atribúty `rychlostX` a `rychlostY`, ktoré určujú, ako veľmi sa zmení poloha objektu (v tomto prípade hada) pri pohybe v danom smere. Konštruktor enumu inicializuje tieto atribúty na základe poskytnutých hodnôt.

Taktiež enum obsahuje metódy `getRychlostX` a `getRychlostY`, ktoré slúžia na získanie rýchlosti pohybu hada v ose X resp. Y pre daný smer.

Snake.java

konštruktor Snake(int sirkaPlochy, int vyskaPlochy, int rychlostHry, int pocetPrekazok, int velkostPolicka)

```
public Snake(int sirkaPlochy, int vyskaPlochy, int rychlostHry, int
pocetPrekazok, int velkostPolicka) {
    //inicializacia atributov
    this.vyskaPlochy = vyskaPlochy;
    this.sirkaPlochy = sirkaPlochy;
    this.pocetPrekazok = pocetPrekazok;
    this.velkostPolicka = velkostPolicka;
    this.zvuk = new Zvuk();
    this.manazer = new Manazer(this);
    this.nahodne = new Random();

    //nastavenie jpanel
    setPreferredSize(new Dimension(this.sirkaPlochy, this.vyskaPlochy));
    addKeyListener(this.manazer);
    setFocusable(true);

    //inicializacia hernych atributov
```

```
        this.hraciaPlocha = new HraciaPlocha(sirkaPlochy, vyskaPlochy,
velkostPolicka, this);
        this.prekazky = new Prekazka(sirkaPlochy, vyskaPlochy, velkostPolicka,
pocetPrekazok);
        this.prekazky.generujNahodnePrekazky(sirkaPlochy, vyskaPlochy,
velkostPolicka, pocetPrekazok);
        this.hlavaHada = this.generujNahodnePolicko();
        this.teloHada = new ArrayList<>();
        this.casOdStartu = System.currentTimeMillis();
        this.casKedyPozastavena = casOdStartu;
        this.casZastaveny = false;
        this.jedlo = new Jedlo();
        this.nahodne = new Random();
        this.jedlo.umiestniJedlo(sirkaPlochy, vyskaPlochy, velkostPolicka,
this.prekazky);
        this.rychlostX = 0;
        this.rychlostY = 0;
        this.hernyCyklus = new Timer(rychlostHry, this);
        this.hernyCyklus.start();
    }
```

Konštruktor triedy `Snake` inicializuje viaceré atribúty pre inštanciu hada a nastavuje herné parametre. Tu je stručný popis časti konštruktora:

- **Inicializácia atribútov**: Nastavujú sa atribúty, ako je šírka a výška hracej plochy, počet prekážok, veľkosť políčka, objekty pre zvuk a náhodné generovanie.
- **Nastavenie JPanel**: Hracia plocha (`JPanel`) je inicializovaná a nastavená na danú šírku a výšku. Taktiež je pridávaný "key listener" pre zachytávanie klávesových udalostí a je nastavovaná ako "focusable", čo umožňuje JPanelu prijímať vstupy.
- **Inicializácia herných atribútov**: Inicializácia rôznych herných atribútov, ako sú hracia plocha, prekážky, hlava hada, tele hada, jedlo, rýchlosti, a ďalšie. Tiež je spúšťaný herný cyklus, ktorý je implementovaný pomocou `Timer` a implementuje rozhranie `ActionListener`.

Celkovo, konštruktor vytvára inštanciu hry so všetkými potrebnými atribútmi a inicializáciami pre správne fungovanie hry hada.

paintComponeent(Graphics g)

```
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        this.vykresli(g);
    }
```

Metóda `paintComponent` je súčasťou metódy pre kreslenie (rendering) v komponente `JPanel`. V tomto prípade táto metóda vykonáva kreslenie grafiky na komponente hada.

- `super.paintComponent(g)`: Volanie nadradenej metódy zabezpečuje, aby sa pred vykresľovaním aktualizoval stav komponenty a vyčistili sa predchádzajúce kresby, čo je dôležité pre správne fungovanie kreslenia.
- `this.vykresli(g)`: Volanie metódy `vykresli` na vykreslenie vizuálnych prvkov hada. Táto metóda zrejme obsahuje kód na vykreslenie hada, prekážok, jedla a ďalších grafických prvkov na komponente.

Celkovo, táto metóda je dôležitá pre správne vizuálne zobrazenie herných prvkov v komponente hada.

vykresli (Graphics g)

```
public void vykresli(Graphics g) {
    this.hraciaPlocha.vykresli(g);
    this.jedlo.vykresli(g, this.velkostPolicka);
    this.manazer.vykresliHada(g);
    this.prekazky.vykresli(g, this.velkostPolicka);

    //telo hada
    g.setColor(new Color(108, 220, 156));
    for (Policko castHada : this.teloHada) {
        g.fillOval(castHada.getX() * this.velkostPolicka, castHada.getY()
* this.velkostPolicka, this.velkostPolicka, this.velkostPolicka);
    }
}
```

Metóda `vykresli` vykonáva kreslenie rôznych prvkov hry na grafický kontext `g`. Tu je krátky popis kódu tejto metódy:

- `this.hraciaPlocha.vykresli(g)`: Vykreslenie hracej plochy na základe jej definície a parametrov.
- `this.jedlo.vykresli(g, this.velkostPolicka)`: Vykreslenie jedla (často reprezentovaného ikonou, obrázkom alebo farbou) na hernú plochu.
- `this.manazer.vykresliHada(g)`: Vykreslenie hada na základe jeho aktuálnej pozície a stavu.
- `this.prekazky.vykresli(g, this.velkostPolicka)`: Vykreslenie prekážok na hernú plochu.
- Vykreslenie tela hada:
  - Nastavenie farby na novú farbu pre tela hada (`new Color(108, 220, 156)`).
  - Pre každý segment hada v tele (`castHada`) sa vykresľuje ovál na hernú plochu.

Celkovo, metóda `vykresli` má na starosti vizuálne zobrazenie rôznych prvkov hry na grafický kontext, čo zahrňuje hraciu plochu, jedlo, hada, prekážky a telo hada.

umiestniJedlo()

```
public void umiestniJedlo() {
    this.jedlo.umiestniJedlo(this.sirkaPlochy, this.vyskaPlochy,
this.velkostPolicka, this.prekazky);
}
```

Metóda `umiestniJedlo` slúži na umiestnenie nového jedla na hernú plochu. Táto metóda volá metódu `umiestniJedlo` na objekte `jedlo`, ktorý reprezentuje jedlo v hre.

Parametre tejto metódy (sirkaPlochy, vyskaPlochy, velkostPolicka, prekazky) sú pravdepodobne použité na určenie vhodnej pozície pre nové jedlo tak, aby sa zabránilo kolízii s prekážkami a inými časťami hry.

Celkovo, táto metóda zabezpečuje, že jedlo je umiestnené na hernú plochu podľa špecifikovaných pravidiel alebo logiky hry.

pohyb()

kolízia hlavyHada a jedla

```
if (kolizia(this.hlavaHada, this.jedlo.getPolicko())) {
    this.teloHada.add(new Policko(this.jedlo.getPolicko().getX(),
this.jedlo.getPolicko().getY()));
    this.umiestniJedlo();
    this.zvuk.zvukJedla();
}
```

Táto podmienka slúži na kontrolu kolízie medzi hlavou hada a jedlom. Ak sa hlava hada dostane na políčko, kde sa nachádza jedlo, vykoná sa nasledujúci kód:

- `this.teloHada.add(new Policko(this.jedlo.getPolicko().getX(), this.jedlo.getPolicko().getY()));`: Pridá nový segment do tela hada na pozíciu jedla. To znamená, že had sa "rozšíri" o jednu časť.
- `this.umiestniJedlo();`: Znovu umiestni nové jedlo na hernú plochu po jeho zjedení.
- `this.zvuk.zvukJedla();`: Spustí zvuk, ktorý indikuje, že had práve zjedol jedlo.

Celkovo, tento blok kódu sa stará o správanie hry po tom, ako had zje jedlo, vrátane aktualizácie tela hada, umiestnenia nového jedla a spustenia zvuku.

časť jedla sa presunie na jej predchádzajúcu pozíciu

```
for (int i = this.teloHada.size() - 1; i >= 0; i--) {
    if (i == 0) {
        this.teloHada.get(i).setX(this.hlavaHada.getX());
        this.teloHada.get(i).setY(this.hlavaHada.getY());
    } else {
        this.teloHada.get(i).setX(this.teloHada.get(i - 1).getX());
```

```
            this.teloHada.get(i).setY(this.teloHada.get(i - 1).getY());
        }
    }
```

Tento blok kódu sa zaoberá aktualizáciou polôh častí tela hada na základe jeho súčasnej polohy a pohybu.

- `for (int i = this.teloHada.size() - 1; i >= 0; i--)`: Cyklus prechádza cez všetky časti tela hada od poslednej (konca) po prvú (hlavu).
- `if (i == 0) { ... }`: Ak sme prišli k hlave hada, aktualizujeme jej polohu na rovnakú ako hlava hada.
- `else { ... }`: Pre ostatné časti tela hada nastavíme ich polohu na polohu predchádzajúcej časti tela. Tým sa dosiahne efekt "posúvania" častí tela za hlavou hada.

Celkovo tento blok kódu zabezpečuje aktualizáciu polôh častí tela hada v smeru jeho pohybu.

kontrola kolízie s okrajmi hracej plochy

```
if (this.hlavaHada.getX() < 0 || this.hlavaHada.getX() >= this.sirkaPlochy
/ this.velkostPolicka || this.hlavaHada.getY() < 0 ||
this.hlavaHada.getY() >= this.vyskaPlochy / this.velkostPolicka) {
    this.koniecHry = true;
}
```

Táto podmienka kontroluje, či hlava hada vyšla mimo hracej plochy. Ak áno, nastaví premennú `koniecHry` na hodnotu `true`, čo znamená, že hra skončila. Podmienka má štyri časti:

1. `this.hlavaHada.getX() < 0`: Kontroluje, či sa hlava hada nachádza za ľavým okrajom hracej plochy.
2. `this.hlavaHada.getX() >= this.sirkaPlochy / this.velkostPolicka`: Kontroluje, či sa hlava hada nachádza za pravým okrajom hracej plochy.
3. `this.hlavaHada.getY() < 0`: Kontroluje, či sa hlava hada nachádza nad horným okrajom hracej plochy.
4. `this.hlavaHada.getY() >= this.vyskaPlochy / this.velkostPolicka`: Kontroluje, či sa hlava hada nachádza pod dolným okrajom hracej plochy.

Ak aspoň jedna z týchto podmienok platí, znamená to, že hlava hada vyšla mimo hracej plochy, a preto je nastavená premenná `koniecHry` na `true`.

pohyb hlavyHada

```
this.hlavaHada.setX(this.hlavaHada.getX() + rychlostX);
this.hlavaHada.setY(this.hlavaHada.getY() + rychlostY);
```

Tieto dva riadky kódu aktualizujú polohu hlavy hada na základe aktuálnych hodnôt rýchlosti (`rychlostX` a `rychlostY`).

- `this.hlavaHada.getX() + rychlostX`: Pripočíta hodnotu `rychlostX` k aktuálnej x-ovej pozícii hlavy hada, čím sa posunie horizontálne.
- `this.hlavaHada.getY() + rychlostY`: Pripočíta hodnotu `rychlostY` k aktuálnej y-ovej pozícii hlavy hada, čím sa posunie vertikálne.

Tieto operácie simulujú pohyb hada na hracej ploche na základe aktuálnych hodnôt rýchlosti v horizontálnom a vertikálnom smere.

kontrola kolízie hlavyHada s telom

```
for (Policko castHada : this.teloHada) {
    if (kolizia(this.hlavaHada, castHada)) {
        this.koniecHry = true;
    }
}
```

Tento blok kódu kontroluje, či hlava hada koliduje s niektorým z členov jeho tela. Pre každý segment tela hada sa kontroluje kolízia s hlavou hada. Ak sa zistí kolízia, nastavuje sa premenná `koniecHry` na hodnotu `true`, čo znamená koniec hry.

V praxi to znamená, že ak hlava hada narazí do svojho tela, hra sa skončí. To je jedna z podmienok pre ukončenie hry v prípade samozrážky hada so sebou.

kontrola kolízie hlavyHada s prekážkou

```
if (this.prekazky.koliziaHadaSPrekazkou(this.hlavaHada)) {
    this.koniecHry = true;
}
```

Táto podmienka kontroluje, či hlava hada koliduje s nejakou prekážkou na hracej ploche. Ak sa táto podmienka vyhodnotí ako `true`, nastavuje sa premenná `koniecHry` na hodnotu `true`, čo znamená koniec hry.

V praxi to znamená, že ak hlava hada narazí do nejakej prekážky, hra sa ukončí. Táto podmienka zabezpečuje, aby had nemohol prejsť cez prekážky, a tým zvyšuje obtiažnosť hry.

kontrola kolízie jedla s prekážkou

```
if (this.jedlo.koliziaJedlaSPrekazkou(this.prekazky)) {
    this.umiestniJedlo();
}
```

Táto podmienka kontroluje, či nové umiestnenie jedla koliduje s nejakou prekážkou na hracej ploche. Ak sa táto podmienka vyhodnotí ako `true`, volá sa metóda `umiestniJedlo()`, ktorá zabezpečí, že jedlo bude umiestnené na novej pozícii, ktorá nekoliduje s prekážkami.

Tento postup je dôležitý pre správne fungovanie hry, aby jedlo bolo vždy umiestnené na bezpečnom mieste, kde ho had môže zjesť, a zároveň aby to bolo v súlade s pravidlami hry.

kolizia(Object objekt1, Object objekt2)

```
public static boolean kolizia(Object objekt1, Object objekt2) {
    if (objekt1 instanceof Policko && objekt2 instanceof Policko) {
        Policko policko1 = (Policko)objekt1;
        Policko policko2 = (Policko)objekt2;
        return policko1.getX() == policko2.getX() && policko1.getY() ==
policko2.getY();
    }
    return false;
}
```

Táto metóda slúži na kontrolu kolízií medzi dvoma objektami. Ak sú oba objekty inštanciami triedy `Policko`, metóda porovná ich súradnice a vráti `true`, ak majú rovnaké súradnice (tj. sú na rovnakej pozícii). Inak vráti `false`. Metóda je navrhnutá tak, aby pracovala so všeobecnými objektami, ale je špecificky implementovaná pre prípad, keď objekty sú inštanciami triedy `Policko`.

actionPerformed(ActionEvent e)

```
public void actionPerformed(ActionEvent e) {
    this.pohyb();
    repaint();

    //kontrola ci skoncila hra
    if (this.koniecHry) {
        Aplikacia aplikacia = new Aplikacia();
        aplikacia.prehralSi(this, this.prekazky);
    }
}
```

V tejto metóde `actionPerformed` sa volá metóda `pohyb()`, ktorá zabezpečuje pohyb hada, a následne sa volá metóda `repaint()`, ktorá vyvolá opätovné vykreslenie komponentu. Po týchto krokoch sa kontroluje, či hra skončila. Ak `koniecHry` má hodnotu `true`, vytvorí sa nová inštancia triedy `Aplikacia` a zavolá sa jej metóda `prehralSi(this, this.prekazky)`, kde this predstavuje aktuálnu inštanciu triedy `Snake`.

keyPressed (KeyEvent e)

```
public void keyPressed(KeyEvent e) {
    this.manazer.keyPressed(e);
```

```
}
```

V tejto metóde `keyPressed` sa volá metóda `keyPressed` z objektu `manazer` (ktorý je typu `Manazer`), a ako parameter sa jej odovzdáva parameter `e` z metódy `keyPressed` tejto triedy. Táto metóda sa používa na spracovanie stlačených kláves.

generujNahodnePolicko()

```
public Policko generujNahodnePolicko() {
    return new Policko(this.nahodne.nextInt(this.sirkaPlochy /
velkostPolicka), this.nahodne.nextInt(this.vyskaPlochy / velkostPolicka));
}
```

Metóda `generujNahodnePolicko` vracia nový objekt typu `Policko`, ktorý je inicializovaný náhodnými hodnotami pre svoje súradnice X a Y. Tieto hodnoty sú generované pomocou inštancie `Random` s názvom `nahodne`. Konkrétne hodnoty sú generované tak, že sú náhodné indexy v rozsahu od 0 do určených hodnôt (počet polí v šírke a výške hracej plochy, deleno veľkosťou políčka).