MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

Jocul educațional pentru copii numit „TriviaGame”
Educational game for kids called "TriviaGame"

Tehnici și mecanisme de proiectare software

A efectuat: Bou Răzvan, st.TI-202

A verificat: asist. univ., Gaidău Mihai

Chișinău 2023

Cuprins

Introducere **3**
1 Analiza detaliată a domeniului de studiu **4**
2 Modelele implementate în aplicație **5**
2.1 Modelul Singleton 6
2.2 Modelul Abstract Factory 8
2.3 Modelul Bridge 10
2.4 Modelul Proxy 12
2.5 Modelul Strategy 14
2.6 Modelul Command 16
3 Interfața grafică a aplicației de trivia "TriviaGame" pentru copii **18**
Concluzie **25**
Bibliografie **26**

Introducere

Odată cu progresul continuu al erei digitale, jocurile educative pentru copii au devenit din ce în ce mai relevante în procesul lor de învățare. Pe măsură ce tehnologia avansează, este esențial să găsim modalități inovatoare și captivante de a stimula cunoștințele și dezvoltarea cognitivă a copiilor noștri. Jocurile educative oferă o soluție eficientă, interactivă și plină de distracție pentru această nevoie. Copiii sunt în mod natural curioși și deschiși către explorare. Prin intermediul jocurilor educative, putem încuraja și cultiva această curiozitate și dorință de a învăța, într-un mediu sigur și captivant. Ele motivează copiii să interacționeze cu conținutul într-un mod distractiv și plăcut, permițându-le să descopere și să asimileze cunoștințe valoroase.

Aplicația TriviaGame deține un rol semnificativ în această eră a jocurilor educative pentru copii. Aceasta oferă o experiență educațională inovatoare, care combină în mod inteligent elementele de învățare și divertisment. TriviaGame este special concepută pentru copiii în vârstă preșcolară și școlară mică, pentru a se potrivi nevoilor lor specifice de dezvoltare și învățare. Dezvoltarea acestei aplicații a fost ghidată de înțelegerea profunzimii impactului pe care jocurile educative îl pot avea asupra dezvoltării cognitive și intelectuale a copiilor. Prin activitățile interactive și jocurile educative adaptate nivelului de învățare al fiecărui copil, TriviaGame oferă o modalitate eficientă și distractivă de a sprijini dezvoltarea lor în diverse domenii, precum matematica, literatura, științele și multe altele.

Importanța aplicației TriviaGame nu se rezumă doar la furnizarea de cunoștințe și abilități academice. Aceasta contribuie, de asemenea, la dezvoltarea abilităților socio-emoționale ale copiilor, cum ar fi colaborarea, rezolvarea de probleme și gândirea critică. Prin intermediul interfeței intuitive și prietenoase, TriviaGame încurajează copiii să exploreze, să experimenteze și să învețe în mod autonom și plăcut. În concluzie, jocurile educative și dezvoltarea aplicației TriviaGame formează o combinație puternică pentru sprijinirea procesului de învățare al copiilor în această eră digitală. TriviaGame nu numai că îi motivează pe copii să învețe, dar le oferă și instrumentele necesare pentru a-și dezvolta abilitățile și cunoștințele într-un mod distractiv și personalizat. Prin intermediul acestei aplicații, copiii devin actori activi în propriul proces de învățare, pregătindu-se pentru provocările și oportunitățile viitoare.

1 Analiza detaliată a domeniului de studiu

TriviaGame este o aplicație educațională captivantă și inovatoare, special creată pentru copiii din grădiniță și școala primară. Această aplicație desktop din domeniul tehnologiilor informaționale are ca scop să ofere o experiență interactivă și plăcută de învățare, care să stimuleze dezvoltarea copiilor într-un mod distractiv. Este important să modernizăm metodele tradiționale de învățare și să utilizăm tehnologia în beneficiul educației. TriviaGame încurajează interesul și curiozitatea copiilor față de învățare, dezvoltă abilitățile cognitive și oferă o experiență personalizată de învățare.

Există deja câteva sisteme similare sau care prezintă aspecte comune cu TriviaGame. De exemplu, "ABC Learning System" este un sistem online de învățare adresat copiilor, care se concentrează pe dezvoltarea abilităților de citire, scriere și matematică. Cu toate acestea, TriviaGame oferă o experiență personalizată de învățare și acoperă o varietate mai largă de domenii de cunoaștere. De asemenea, "Math Adventure" este o aplicație mobilă care se concentrează exclusiv pe dezvoltarea abilităților matematice la copii, iar "Reading Buddy" se concentrează pe dezvoltarea abilităților de citire și înțelegere a textelor. În comparație, TriviaGame acoperă mai multe domenii, inclusiv matematică, logica și abilitățile cognitive.

Scopul și obiectivele proiectului TriviaGame sunt de a crea o aplicație desktop interactivă și personalizată, care să stimuleze învățarea și să dezvolte abilitățile cognitive la copii. Principalele cerințe funcționale ale sistemului includ o gamă diversificată de activități și jocuri educaționale în diferite domenii de cunoaștere, adaptabilitate la nivelul de învățare și nevoile individuale ale copiilor și o interfață intuitivă și atrăgătoare pentru copii.

Pe viitor, funcționalitățile sistemului și ale subsistemelor vor include și mai multe activități interactive, jocuri educaționale, module de evaluare și monitorizare a progresului, algoritmi de adaptare la nivelul de învățare al copiilor și un sistem de administrare a conturilor și datelor personale.

TriviaGame își propune să stimuleze învățarea la copii prin obiective bine definite. Prin activitățile interactive și jocurile educaționale, aplicația își propune să stimuleze interesul și curiozitatea copiilor față de învățare și să îi motiveze să exploreze diferite domenii de cunoaștere. De asemenea, aplicația îi ajută pe copii să-și dezvolte abilitățile cognitive, cum ar fi gândirea logică, rezolvarea de probleme, atenția concentrată și memoria. Personalizarea experienței de învățare este un aspect distinctiv al TriviaGame, deoarece se adaptează la nivelul și ritmul de învățare al fiecărui copil, oferindu-le conținut și activități potrivite pentru capacitățile și nevoile lor individuale.

TriviaGame este o aplicație educațională care își propune să stimuleze învățarea, să dezvolte abilitățile cognitive, să ofere o experiență de învățare personalizată și captivantă. Prin îndeplinirea acestor obiective, TriviaGame contribuie la formarea copiilor ca indivizi curioși, deschiși la cunoaștere și pregătiți

pentru succesul academic și personal. Aplicația se angajează să fie un partener de încredere pentru părinți și educatori, sprijinind dezvoltarea cognitive, intelectuală și socială a copiilor. TriviaGame oferă o platformă educațională inovatoare și contribuie la dezvoltarea copiilor într-un mod plăcut și eficient.

2 Modelele implementate în aplicație

În aplicația TriviaGame, s-au implementat mai multe șabloane de proiectare pentru a crea o arhitectură solidă și modulară. Aceste șabloane sunt abordări bine-cunoscute și testate în industria software, care oferă soluții elegante și flexibile pentru diverse probleme de proiectare și dezvoltare a aplicației.

Aceste șabloane sunt utilizate în diferite aspecte ale aplicației pentru a îndeplini scopurile și obiectivele sale.

Unul dintre șabloanele utilizate în TriviaGame este Singleton. Acesta este folosit pentru a controla redarea sunetelor în aplicație. Prin crearea unei singure instanțe de redare a sunetelor, se evită suprapunerea sunetelor și se asigură o experiență consistentă pentru utilizatori. De asemenea, se utilizează o listă de tip queue pentru a gestiona ordinea redării întrebărilor, astfel încât o întrebare să fie redată doar după finalizarea celei precedente.

Șablonul Abstract Factory este utilizat în TriviaGame pentru a crea două fabrici separate de întrebări: una pentru întrebările generale și una pentru întrebările tematice. Aceasta permite aplicației să creeze și să utilizeze întrebări specifice fiecărui tip în funcție de nevoile și preferințele utilizatorilor.

Șablonul Bridge este implementat în aplicație pentru a gestiona generarea și combinarea culorilor și formelor. Prin separarea abstractizării formelor și culorilor, se permite o varietate independentă și o combinație flexibilă a acestora. Acest lucru permite TriviaGame să creeze și să utilizeze întrebări de diferite culori într-un mod modular și extensibil.

Șablonul Proxy este utilizat pentru a simplifica și oferi o interfață simplă pentru verificarea răspunsurilor din joc, în special pentru cele referitoare la culori. Clasa proxy permite efectuarea de operații sau verificări suplimentare înainte sau după apelarea metodelor obiectului real, cum ar fi validarea răspunsurilor.

Șablonul Strategy este aplicat în TriviaGame pentru gestionarea diferitelor tipuri de operații matematice, cum ar fi adunarea și scăderea punctajului. Prin intermediul acestui pattern, se definește o interfață comună pentru diverse strategii de operație, permițând înlocuirea flexibilă a strategiilor în funcție de nevoile utilizatorilor. Acest lucru permite TriviaGame să ofere operații matematice într-un mod modular și flexibil, adaptat nivelului și preferințelor individuale ale jucătorilor.

Șablonul Command este utilizat în TriviaGame pentru inițializarea și gestionarea operațiilor de adunare și scădere a punctajului. Prin intermediul unor obiecte de comandă specifice, se separă solicitarea acțiunii de

implementarea efectivă a acesteia. Astfel, TriviaGame poate gestiona și executa operații matematice într-un mod flexibil și extensibil.

Prin utilizarea acestor șabloane de proiectare, TriviaGame beneficiază de o arhitectură modulară, flexibilă și ușor de întreținut. Aceasta permite aplicației să ofere o experiență interactivă și captivantă pentru jucători, adaptată nivelului și preferințelor individuale. Utilizarea șabloanelor de proiectare contribuie la structurarea și organizarea eficientă a funcționalităților TriviaGame, asigurând astfel succesul și utilizarea optimă a aplicației.

2.1 Modelul Singleton

Pattern-ul Singleton este utilizat în aplicația TriviaGame pentru a asigura existența unei singure instanțe a unei clase în întreaga aplicație. Acesta aduce cu sine beneficii și funcționalități esențiale în cadrul aplicației. În cazul aplicației noastre, Singleton este folosit pentru a gestiona redarea sunetelor. Utilizarea Singleton în TriviaGame aduce mai multe avantaje. În primul rând, acesta garantează consistența și coerența datelor prin evitarea creării mai multor instanțe ale aceleiași clase și gestionarea separată a datelor. În al doilea rând, Singleton facilitează accesul global la instanța clasei, permițând utilizarea ușoară a funcționalităților sale în cadrul aplicației. În plus, Singleton poate oferi funcționalități suplimentare, precum controlul asupra instanței sau gestionarea resurselor.

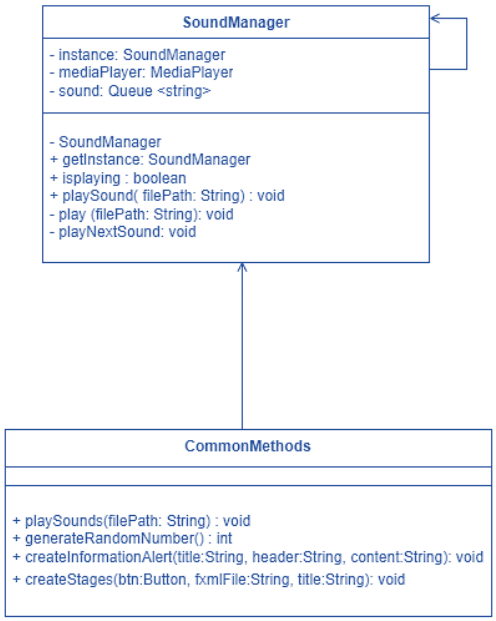

Figura 2.1 – UML-ul șablonului Singleton implementat în aplicația TriviaGame

Clasa SoundManager din diagrama UML de mai jos utilizează pattern-ul Singleton pentru a gestiona redarea sunetelor în aplicația TriviaGame. Clasa SoundManager este inclusă în pachetul "singleton". În codul de mai jos, există o declarație pentru variabila statică "instance" de tip SoundManager, care reprezintă unica instanță a clasei. De asemenea, avem și o variabilă "mediaPlayer" de tip MediaPlayer, care va fi folosită pentru redarea sunetelor, și o coadă "soundQueue" de tip Queue<String>, care va gestiona ordinea de redare a sunetelor.

```
    private static SoundManager instance;
    private MediaPlayer mediaPlayer;
    private Queue<String> soundQueue;
```

Pentru a preveni crearea directă a unor noi instanțe ale clasei, constructorul clasei este declarat privat. Pattern-ul Singleton este implementat în metoda getInstance() prin verificarea existenței unei instanțe și returnarea acesteia, sau crearea unei noi instanțe în cazul în care este necesar.

```
   public static SoundManager getInstance() {
        if (instance == null) {
            instance = new SoundManager();
        }
        return instance;
    }
```

Prin metoda isPlaying() se verifică starea obiectului mediaPlayer pentru a determina dacă un sunet este în curs de redare. Metoda playSound(String filePath) este responsabilă cu redarea sunetului. În cazul în care există deja un sunet în curs de redare, sunetul curent este adăugat în coada soundQueue. În caz contrar, sunetul este redat imediat prin apelul metodei play(String filePath), care primește calea către fișierul audio și creează un obiect Media și MediaPlayer pentru redarea sunetului. De asemenea, se setează un listener pentru evenimentul de încheiere a sunetului, astfel încât când sunetul se termină, se eliberează resursele și se trece la redarea următorului sunet din coadă prin apelul metodei playNextSound(). Metoda privată playNextSound() verifică dacă coada soundQueue nu este goală și redă următorul sunet din coadă prin apelul metodei play(String filePath).

```
    public boolean isPlaying() {
        return mediaPlayer != null && mediaPlayer.getStatus() ==
MediaPlayer.Status.PLAYING;
    }
    public void playSound(String filePath) {
        if (isPlaying()) {
            soundQueue.offer(filePath);
            System.out.println("A sound is already playing. Waiting for it to
finish...");
        } else {
```

```
            play(filePath);
        }
    }
```

Implementarea clasei SoundManager garantează utilizarea unei singure instanțe pentru gestionarea redării sunetelor în aplicația TriviaGame. În plus, este utilizată o coadă pentru a controla ordinea redării sunetelor și a evita suprapunerea lor, asigurând astfel o experiență plăcută și fără întreruperi pentru utilizatori. Această abordare contribuie la menținerea consistenței și coerenței sunetelor în cadrul aplicației, asigurând că fiecare sunet este redat în mod corect și că utilizatorii pot bucura de o experiență audio de calitate în timpul jocului TriviaGame.

2.2 Modelul Abstract Factory

Pattern-ul Abstract Factory este utilizat în cadrul jocului TriviaGame pentru a crea și gestiona diferite tipuri de întrebări într-un mod abstract și consistent. Prin intermediul acestui pattern, se pot crea obiecte dintr-o familie de clase fără a specifica clasele concrete utilizate, oferind o metodă flexibilă de creare a întrebărilor în funcție de nevoile și contextul jocului.

În cadrul aplicației TriviaGame, se utilizează Abstract Factory pentru a crea două tipuri distincte de fabrici: fabrica de întrebări ușoare și fabrica de întrebări dificile. Aceste fabrici sunt responsabile de crearea și furnizarea diferitelor tipuri de întrebări în funcție de nivelul de dificultate.

Codul prezentat implementează pattern-ul Abstract Factory în jocul TriviaGame, având ca scop gestionarea creării și utilizării întrebărilor. Sunt definite două interfețe: Question (reprezentând întrebarea) și AbstractQuestionFactory (reprezentând fabrica abstractă de întrebări). Interfața Question conține metode precum getQuestionText() pentru a obține textul întrebării și getDifficulty() pentru a obține nivelul de dificultate al întrebării. Interfața AbstractQuestionFactory servește drept interfață marker pentru fabricile de întrebări și include metoda createQuestion() pentru crearea efectivă a întrebărilor.

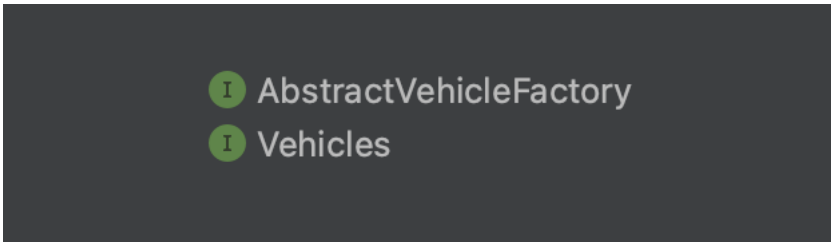

Figura 2.2 – Interfețele „AbstractVehicleFactory” și „Vehicles”

În continuare, cele două secțiuni definesc fabricile concrete de vehicule de urgență. Fabrica de Vehicule de Urgență este implementată în clasa UrgentVehiclesFactoryImpl, care respectă interfața UrgentVehiclesFactory. Această clasă conține o metodă createVehicles(String vehiclesName) care utilizează

o instrucțiune switch pentru a crea obiectele concrete de transport de urgență, precum Poliție, Ambulanță sau Pompieri. Aceeași structură și abordare sunt implementate și în secțiunea de cod responsabilă de crearea vehiculelor.

```
public class UrgentVehicleFactoryImpl implements UrgentVehicleFactory {
    @Override
    public Vehicles createVehicle(String vehicleName) {
        switch (vehicleName) {
            case "fire":
                return new Firefighters();
            case "police":
                return new Police();
            case "ambulance":
                return new Ambulance();
            default:
                throw new RuntimeException("No such vehicle");
        }
    }
}
```

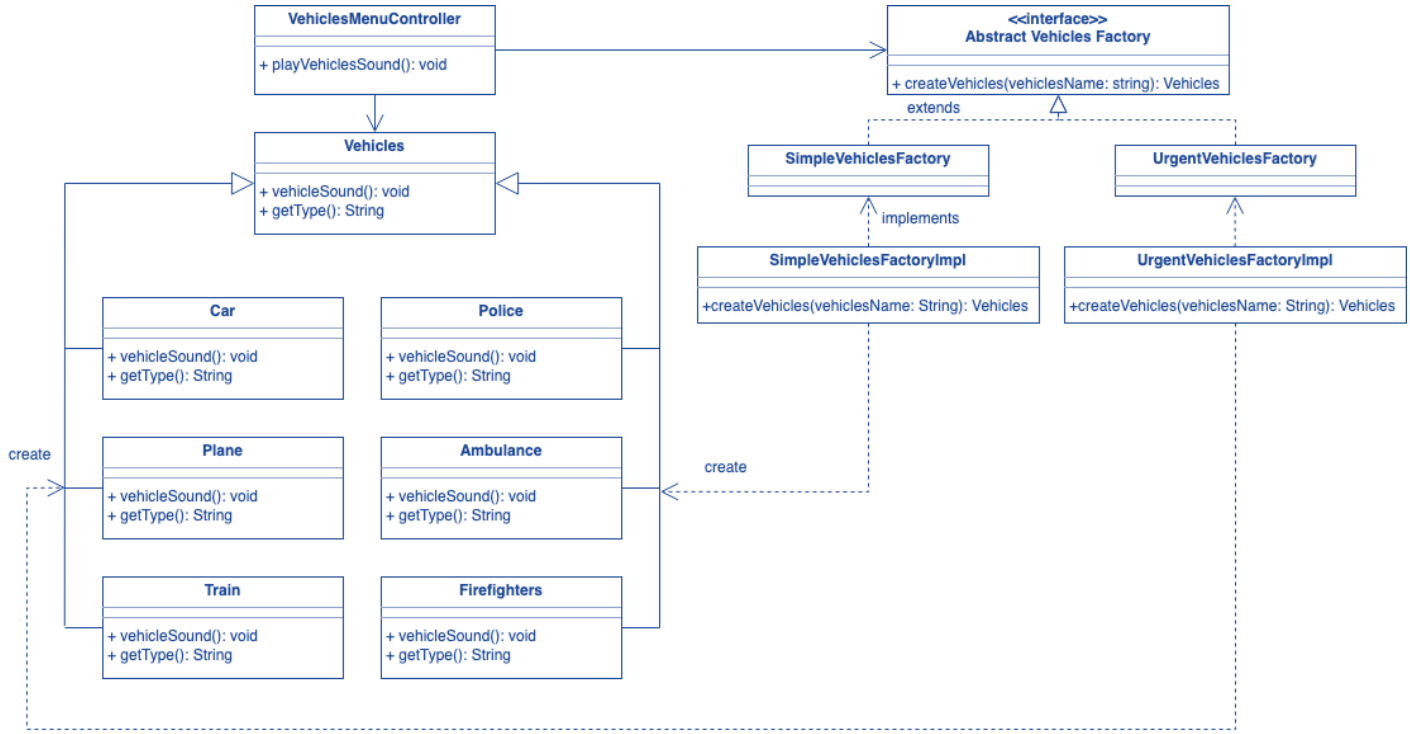

Figura 2.3 – UML-ul șablonului Abstract Factory implementat în aplicația TriviaGame

Următoarea clasă din cod reprezintă o implementare concretă a unui vehicul. Clasa Train implementează interfața Vehicles și conține implementările pentru metodele vehicleSound() și getDetails(). Această clasă redă sunetul specific trenului și returnează un șir de caractere care indică detalii despre vehicul, în acest caz că "Trenul merge pe șine".

```
public class Train implements Vehicles {
    @Override
    public void vehicleSound() {
        String filePath =
"src/main/resources/com/example/rzvnApp/sounds/vehicleSounds/train5sec.mp3";
        playSounds(filePath);
    }

    @Override
    public String getDetails() {
        return "Trenul merge pe șine";
    }
```

Celelalte clase concrete de vehicule au implementări similare, fiecare redând sunetul specific al vehiculului și returnând un șir de caractere care indică tipul vehiculului.

În general, acest cod exemplifică utilizarea șablonului Abstract Factory pentru a crea și utiliza vehicule în cadrul aplicației TriviaGame. Abstract Factory permite crearea de fabrici specializate pentru diferite categorii de vehicule, fie ele de urgență sau obișnuite. Aplicația utilizează aceste fabrici concrete pentru a crea vehicule specifice și pentru a accesa metodele specifice ale acestora, cum ar fi redarea sunetelor și obținerea tipului de vehicul.

2.3 Modelul Bridge

În cadrul aplicației TriviaGame, se utilizează șablonul Bridge pentru a gestiona generarea și combinarea culorilor și figurilor, oferind astfel flexibilitate și extensibilitate în utilizarea lor în întreaga aplicație. Acest pattern permite separarea conceptelor de figură și culoare, permițându-le să varieze independent și să fie combinate în moduri diverse, contribuind astfel la diversitatea și adaptabilitatea jocului TriviaGame.

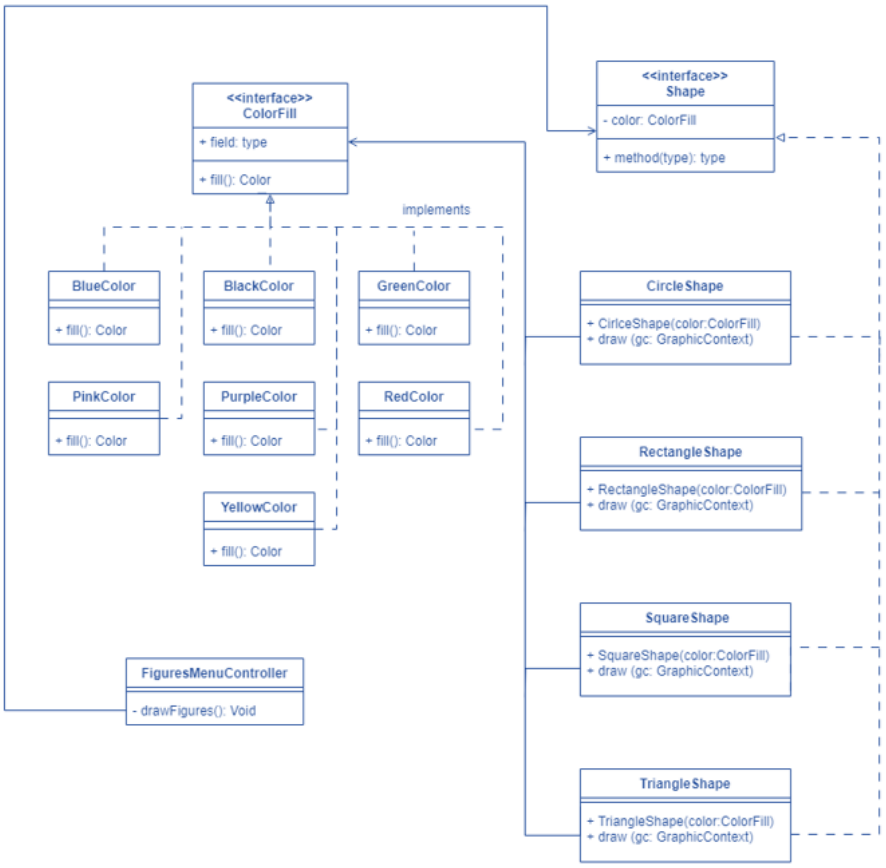

Figura 2.4 – UML-ul șablonului Bridge implementat în aplicația TriviaGame

În cadrul interfeței ColorFill este definită metoda fill(), care returnează o culoare. Această interfață acționează ca o punte între culori și figurile asociate, conform implementării prezentate în codul de mai jos:

```
public interface ColorFill {
    Color fill();
}
```

În fișierele BlackColor, BlueColor și alte fișiere pentru culori, se găsesc clase concrete care implementează interfața ColorFill. Aceste clase oferă implementări specifice și returnează culorile corespunzătoare, cum ar fi Color.BLACK și Color.BLUE, conform cerințelor proiectului.

```
public class BlackColor implements ColorFill {
    @Override
    public Color fill() {
       return Color.BLACK;
    }
}
```

În fișierele CircleShape și RectangleShape se găsesc clasele concrete pentru figuri, care implementează interfața Shape. Aceste clase primesc un obiect de tip ColorFill în constructor și utilizează această culoare pentru a desena forma corespunzătoare în metoda draw(). De exemplu, clasa CircleShape utilizează culoarea specificată pentru a desena un cerc în implementarea sa.

```
public class CircleShape implements Shape{
    private ColorFill color;
    public CircleShape(ColorFill color) {
        this.color = color;
    }
    @Override
    public void draw(GraphicsContext gc){
        gc.setFill(color.fill());
        gc.fillOval(30, 0, 60, 60);
    }
}
```

Prin utilizarea pattern-ului Bridge, figurile și culorile pot fi separate și pot varia independent una de cealaltă. Figurile utilizează o referință către obiecte de tip ColorFill pentru a obține culoarea, evitând astfel dependența directă de clasele concrete de culori.
Această abordare modulară și flexibilă facilitează adăugarea de noi culori sau figuri în aplicația TriviaGame. De exemplu, noi clase de culori, precum MagentaColor sau GrayColor, pot fi adăugate și pot fi combinate cu figuri existente. La fel, noi figuri, cum ar fi PentagonShape sau OctagonShape, pot fi introduse și pot fi combinate cu culorile existente. Prin separarea conceptelor de culoare și formă, pattern-ul Bridge oferă un design mai modular și extensibil pentru aplicația TriviaGame, permițând o combinație flexibilă a culorilor și formelor pe parcursul utilizării lor.

2.4 Modelul Proxy

Scopul șablonului Proxy în cadrul aplicației prezentate este de a oferi un intermediar între client și obiectul real. Proxy-ul acționează ca un substitut sau reprezentant al obiectului real, gestionând accesul la acesta și permițând efectuarea unor operații suplimentare înainte sau după apelarea metodelor obiectului real.

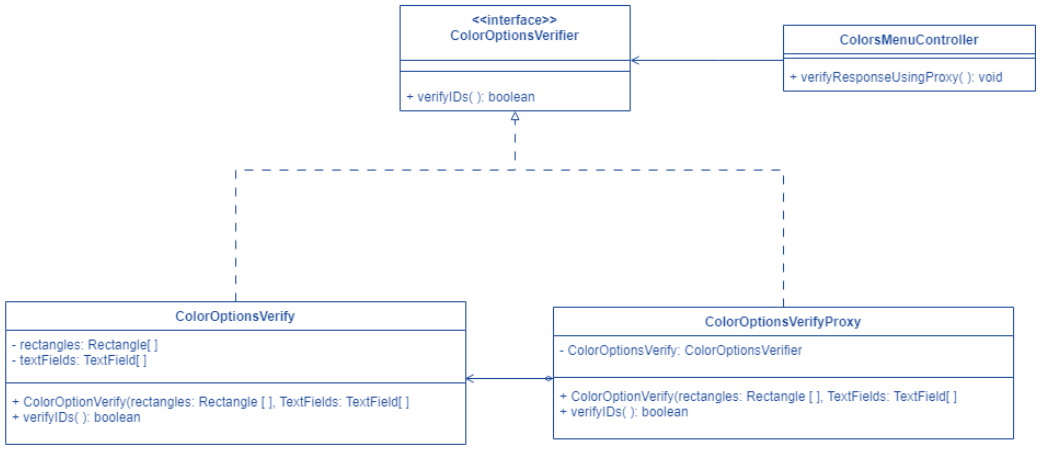

Figura 2.5 – UML-ul șablonului Proxy implementat în aplicația TriviaGame

Clasa ColorOptionsVerifyProxy acționează ca un intermediar între client și obiectul real ColorOptionsVerify prin intermediul interfeței ColorOptionsVerifier. Acest proxy oferă un nivel suplimentar de control sau funcționalitate înainte sau după apelarea metodei verifyIDs(). Prin adăugarea logicii specifice în clasa proxy, se poate efectua o validare sau verificare a datelor de intrare înainte de a apela metoda verifyIDs() pe obiectul real. Aceasta permite proxy-ului să ofere un comportament adițional în comparație cu obiectul real.

```
public class ColorOptionsVerifyProxy implements ColorOptionsVerifier {
    private final ColorOptionsVerifier colorOptionsVerify;
    public ColorOptionsVerifyProxy(Rectangle[] rectangles, TextField[] textFields) {
        colorOptionsVerify = new ColorOptionsVerify(rectangles, textFields);
    }

@Override
    public boolean verifyIDs() {
        return colorOptionsVerify.verifyIDs();
    }
}
```

Metoda verifyIDs() este responsabilă de verificarea corectitudinii răspunsurilor introduse de utilizator. Această metodă compară identificatorii pătratelor și textelor corespunzătoare pentru a determina dacă aceștia sunt identici. În cazul în care există cel puțin o pereche de identificatori diferiți, metoda returnează false, altfel returnează true. Astfel, metoda verifyIDs() furnizează un mecanism de validare pentru asigurarea corectitudinii răspunsurilor utilizatorului în contextul aplicației.

```
public boolean verifyIDs() {
        for (int i = 0; i < rectangles.length; i++) {
            String rectangleId = rectangles[i].getId();
            String textFieldId = textFields[i].getText();
            if (!rectangleId.equals(textFieldId)) {
                return false;
            }
        }
        return true;
    }
```

Proxy-ul este utilizat în aplicația TriviaGame pentru a oferi un nivel de abstractizare și control între client și obiectul real. Prin intermediul unei interfețe simple, clientul poate interacționa cu proxy-ul fără a fi conștient de detaliile tehnice sau logica complexă a obiectului real. Aceasta adaugă flexibilitate, control și extensibilitate în interacțiunea cu obiectul real.

Utilizarea unui proxy în aplicație poate aduce multiple beneficii, precum îmbunătățirea securității prin filtrarea și validarea cererilor, optimizarea performanței prin intermediul caching-ului sau gestionarea resurselor prin intermediul unui control mai strict. Proxy-ul oferă un nivel suplimentar de abstracție și control asupra interacțiunii cu obiectul real, permițând adaptarea și îmbunătățirea funcționalității aplicației TriviaGame într-un mod modular și extensibil.

2.5 Modelul Strategy

În aplicația TriviaGame, se utilizează șablonul Strategy pentru definirea și gestionarea tipurilor de întrebări și răspunsuri care sunt utilizate în joc. În interfața QuestionStrategy, este definită metoda generateQuestion() care returnează o întrebare generată aleatoriu.

```
public interface OperationStrategy {
    int execute(int operand, int total);
}
```

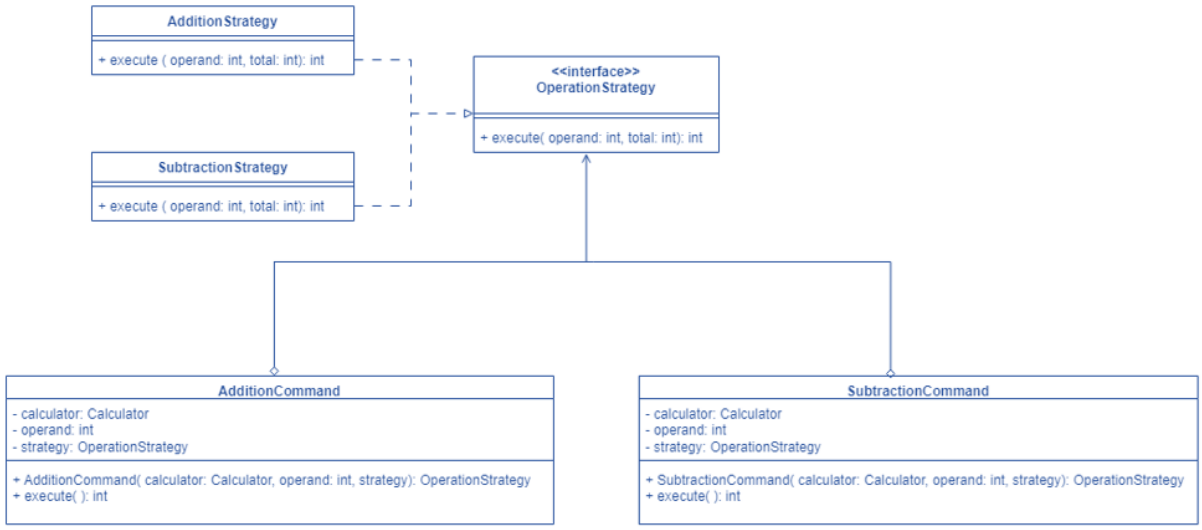

Figura 2.6 – UML-ul șablonului Strategy implementat în aplicația TriviaGame

Clasa AdditionStrategy implementează interfața OperationStrategy și definește metoda execute() pentru operația de adunare. Prin intermediul acestei metode, valoarea operand este adăugată la totalul existent. Clasa SubtractionStrategy, de asemenea, implementează interfața OperationStrategy și definește metoda execute() pentru operația de scădere. Prin intermediul acestei metode, valoarea operand este scăzută din totalul existent.

```
public class SubtractionStrategy implements OperationStrategy {
    @Override
    public int execute(int operand, int total) {
        return total - operand;
    }
}
```

Prin utilizarea acestor clase de strategii și a șablonului Strategy, putem schimba comportamentul de calcul al operațiilor matematice în funcție de strategia selectată. Această abordare oferă o flexibilitate înaltă, permițând schimbarea dinamică a strategiei în timpul execuției. Astfel, putem adăuga și extinde noi strategii de operații în aplicație fără a afecta codul existent. Acest aspect facilitează adaptarea și extinderea funcționalității aplicației TriviaGame într-un mod modular și flexibil.

2.6 Modelul Command

În aplicația TriviaGame, se utilizează șablonul Command pentru a separa logica de executare a comenzilor de obiectul care primește comanda. Acest șablon oferă flexibilitate și extensibilitate în gestionarea comenzilor în timpul execuției, permițând adăugarea și configurarea dinamică a acestora fără a fi direct dependente de implementarea comenzilor sau de obiectul care le primește. În plus, șablonul facilitează operațiile de ștergere și reexecutare a comenzilor într-un mod controlat și ușor de gestionat. Prin utilizarea șablonului Command, aplicația TriviaGame devine modulară și permite o gestionare mai eficientă a comenzilor și a interacțiunilor cu utilizatorul.

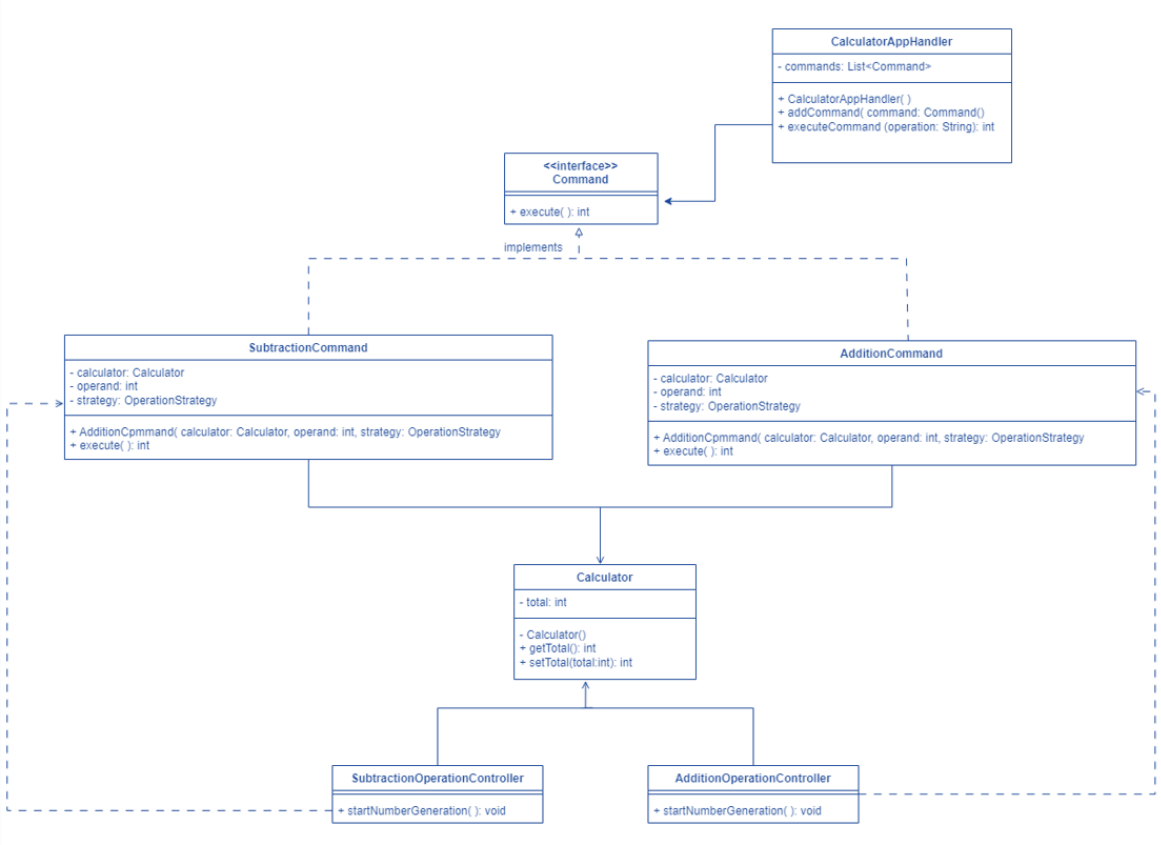

Figura 2.7 – UML-ul șablonului Command implementat în aplicația TriviaGame

Interfața Command definează o metodă execute(), care reprezintă acțiunea executată de o comandă:

```
public interface Command {
    void execute();
}
```

Clasa CalculatorAppHandler servește drept gestionar pentru comenzi și menține o listă de obiecte de tipul Command. Metoda addCommand() adaugă o comandă în lista de comenzi, în timp ce metoda executeCommands() primește un parametru "operation" care indică tipul de operație (adunare sau scădere). În funcție de valoarea parametrului "operation", lista de comenzi este parcursă, iar pentru fiecare comandă se apelează metoda execute(). Rezultatul operațiilor este calculat și returnat. Codul corespunzător acestei clase poate fi găsit în Anexa 1.

Clasa Calculator reprezintă obiectul care primește comenzile. În exemplul dat, clasa Calculator deține o variabilă "total" pentru a stoca rezultatul calculului. Metoda setTotal() actualizează valoarea total și returnează noua valoare.

```
public class Calculator {
  private int total;
  public Calculator() {
      total = 0;
  }
  public int getTotal() {
      return total;
  }
  public int setTotal(int total) {
      this.total = total;
      return total;
  }
```

În aplicația TriviaGame, clasa SubtractCommand și clasa AdditionCommand implementează interfața Command, oferind funcționalitatea de scădere și adunare. Aceste comenzi primesc o instanță a obiectului Calculator, o valoare operand și o strategie de operație (OperationStrategy) în constructorul lor. Metoda execute() apelează metoda execute() a strategiei de operație pentru a obține rezultatul, apoi utilizează metoda setTotal() a calculatorului pentru a actualiza valoarea totală și a returna noua valoare.

Utilizarea șablonului Command în aplicația TriviaGame are ca scop separarea logicii de executare a comenzilor de obiectul care primește comanda și oferirea unei gestionări flexibile a comenzilor în funcție de nevoi și de operațiile solicitate. Acest șablon facilitează implementarea unui sistem robust de înregistrare a comenzilor, permițând reexecutarea sau anularea acestora în viitor. Flexibilitatea oferită de șablonul Command permite integrarea ușoară a funcționalităților complexe, inclusiv compunerea și executarea în lanț a comenzilor, pentru a obține operații mai complexe și modulare în cadrul aplicației TriviaGame.

3 Interfața grafică a aplicației de trivia "TriviaGame" pentru copii

În cadrul dezvoltării aplicațiilor moderne, tehnologiile JavaFX și Maven s-au impus ca soluții populare și eficiente pentru construirea interfețelor grafice și gestionarea dependențelor în proiectele Java. JavaFX, o platformă de dezvoltare Java dedicată creării interfețelor grafice pentru utilizatori (GUI), a câștigat popularitate înlocuind bibliotecile clasice, precum Swing. JavaFX oferă o gamă extinsă de componente grafice, animații, efecte vizuale și funcționalități de interacțiune cu utilizatorul. Este nativ integrat în Java Development Kit (JDK) începând cu versiunea 7, facilitând dezvoltarea și implementarea aplicațiilor JavaFX pe diverse platforme, inclusiv desktop, web și dispozitive mobile.

Interfața grafică a aplicației TriviaGame are un design atrăgător și prietenos. Utilizează culori vii, ilustrații și elemente grafice sugestive pentru a crea o atmosferă captivantă și plină de viață. Meniurile și sistemul de navigare sunt intuitive, permițând utilizatorilor să acceseze cu ușurință diferite secțiuni și funcționalități ale aplicației. Elementele interactive, precum butoanele, facilitează interacțiunea utilizatorului cu conținutul aplicației. Interfața grafică este structurată în secțiuni și ecrane relevante, astfel încât fiecare secțiune să ofere o experiență completă și coerentă.

În general, interfața grafică a aplicației TriviaGame urmărește să ofere utilizatorilor o experiență plăcută, interactivă și ușor de utilizat. Meniul aplicației TriviaGame, prezentat în figura 3.2, este conceput pentru a oferi utilizatorilor acces rapid și facil la diverse funcționalități și opțiuni disponibile în cadrul aplicației. Meniul principal este structurat în categorii distincte, incluzând opțiuni pentru întrebări, moduri de joc, setări și statistici. Această organizare intuitivă permite utilizatorilor să navigheze cu ușurință și să aleagă opțiunile dorite într-un mod eficient și convenabil.

Figura 3.2 – Interfața meniului accesat de utilizatori

Prin selectarea opțiunii de culori, utilizatorii au posibilitatea de a explora și experimenta diferite culori și nuanțe. Paleta de culori oferă o gamă variată de opțiuni, inclusiv nuanțe de roșu, albastru, verde, galben și multe altele. Utilizatorii pot introduce o cifră în casetele de text corespunzătoare, așa cum este prezentat în figura 3.3, iar cifra respectivă va fi aplicată ca text în dreptul culorii. Această interacțiune permite utilizatorilor să personalizeze culorile în funcție de preferințele lor și să obțină rezultate vizuale imediate.

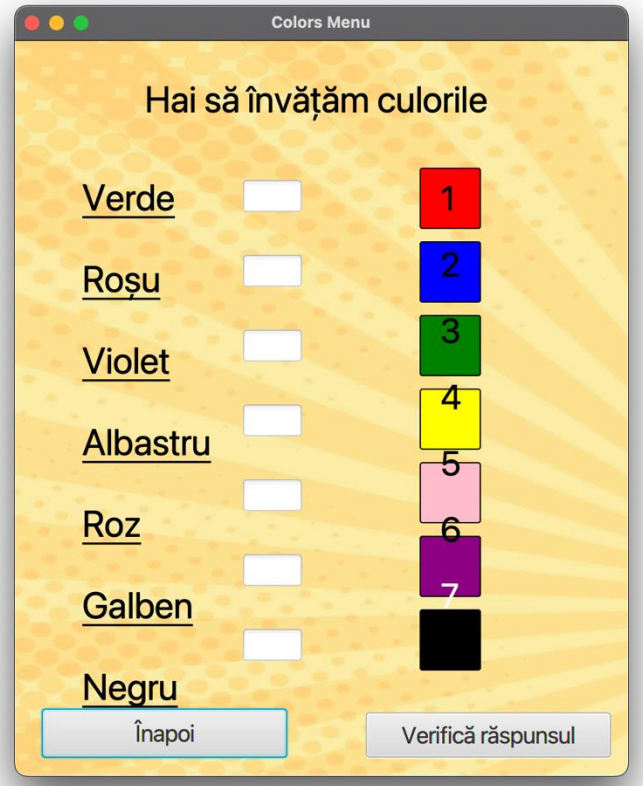

Figura 3.3 – Interfața quizului de culori cu variantele de răspuns introduse în casete

În interfața quizului de figuri geometrice, utilizatorii au posibilitatea să-și testeze cunoștințele despre diferite forme geometrice, cum ar fi cercuri, triunghiuri, pătrate, dreptunghiuri și altele. Fiecare formă geometrică este reprezentată vizual în interfață, iar utilizatorii trebuie să introducă numărul corespunzător acelei forme în caseta de text asociată. Asemenea quiz-ului de culori, acesta oferă utilizatorilor o modalitate interactivă de a se familiariza și de a-și testa cunoștințele despre formele geometrice. Figura 3.4 prezintă un exemplu al acestui quiz.

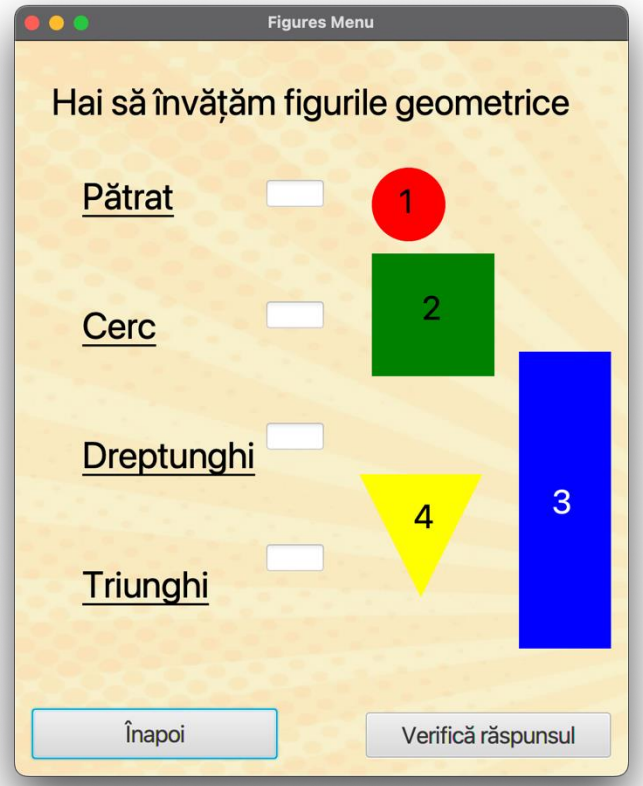

Figura 3.4 – Interfața quizului efectuat pentru învățarea figurilor

Opțiunea matematică din aplicație oferă utilizatorilor posibilitatea de a exersa și de a se familiariza cu conceptele matematice de bază. Aceasta implică selectarea a două tipuri de operații matematice pe care utilizatorul dorește să le practice. În meniul prezentat în figura 3.5, utilizatorul poate alege între adunare și scădere ca tipuri de operații disponibile. Aceasta oferă utilizatorilor o modalitate interactivă de a-și consolida abilitățile matematice și de a exersa operații specifice într-un mod atractiv și prietenos.

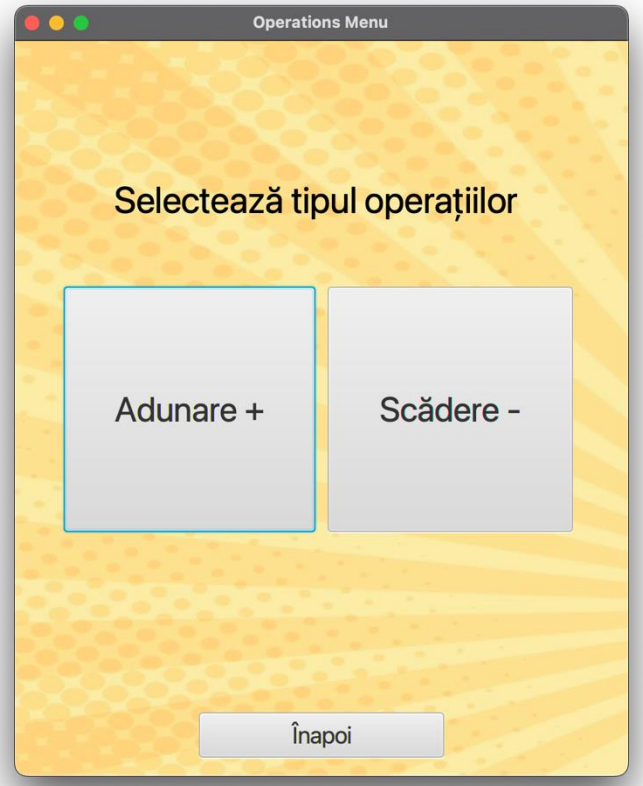

Figura 3.5 – Meniul utilizat de clienți pentru selectarea operațiilor

După ce utilizatorul selectează tipul de operație dorită, o interfață sugestivă este prezentată pentru a efectua operații matematice cu cifre generate aleatoriu. Butonul "Start" din figurile 3.7 și 3.8 inițiază generarea numerelor aleatorii și efectuarea operațiilor matematice în backend-ul aplicației. După finalizarea generării numerelor, rezultatul operațiilor este stocat în spatele interfeței grafice, iar clientul trebuie să introducă propriul răspuns. Apăsând butonul "Verifică răspunsul", răspunsurile primite sunt comparate, iar clientului i se afișează textul corespunzător variantei introduse și se redă un sunet muzical în cazul unui răspuns corect.

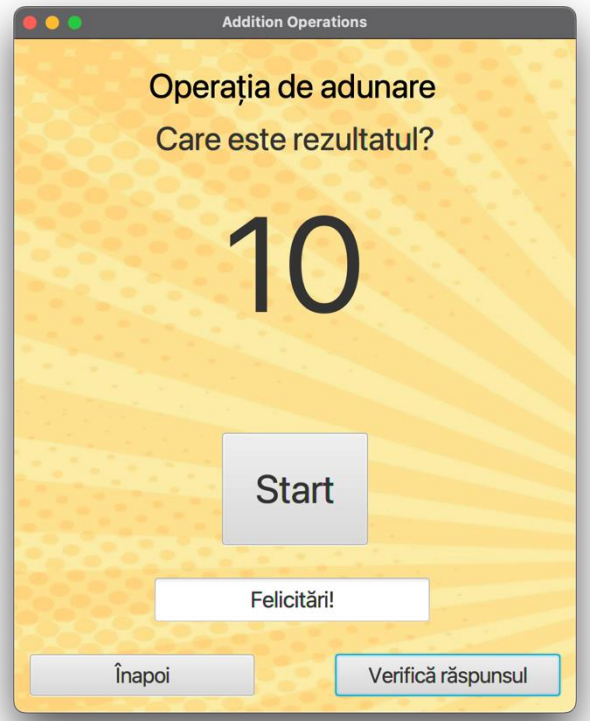

Figura 3.7 – Reprezintă interfața operației de adunare a numerelor

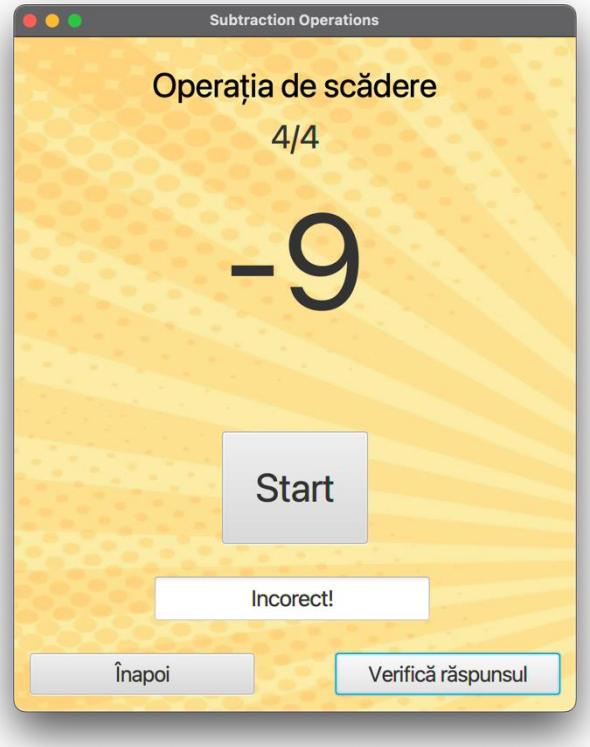

Figura 3.8 – Reprezintă interfața operației de scădere a numerelor

Prin selectarea opțiunii "Vehicule", utilizatorii pot explora și învăța despre lumea vehiculelor. Interfața afișează tipul de transport selectat: "Transport de urgență" sau "Transport simplu", precum și sunetele asociate fiecărui tip de transport. Prin apăsarea butoanelor corespunzătoare transportului din figura 3.9, se redau sunetele specifice pentru fiecare tip de transport.

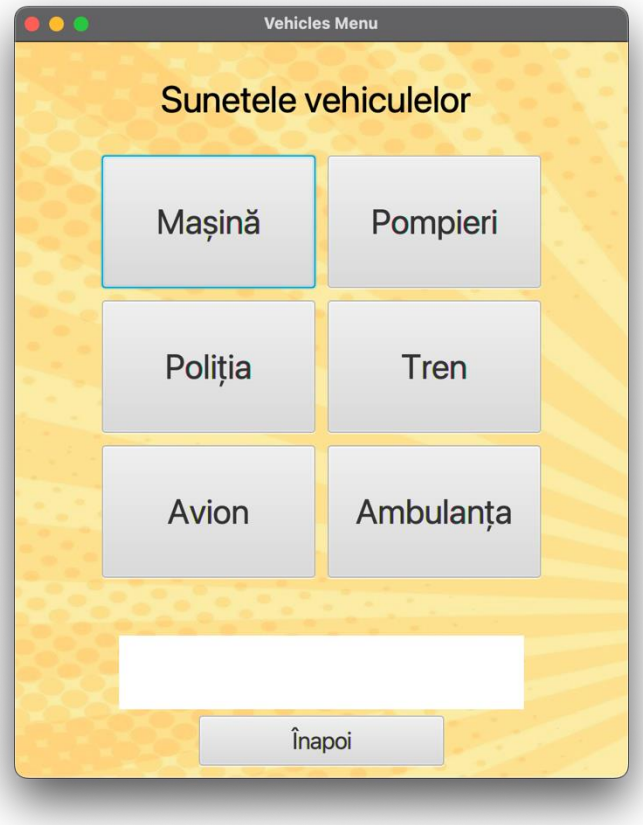

Figura 3.9 – Interfața meniului de vehicule

Concluzie

TriviaGame este o aplicație educațională captivantă și inovatoare, creată special pentru copiii din grădiniță și școala primară. Această aplicație desktop utilizează tehnologia informațională pentru a oferi o experiență interactivă și plăcută de învățare, care să stimuleze dezvoltarea copiilor într-un mod distractiv. TriviaGame se diferențiază de alte sisteme sau aplicații similare prin acoperirea unei varietăți mai largi de domenii de cunoaștere și prin personalizarea experienței de învățare în funcție de nivelul și nevoile fiecărui copil.

Aplicația are ca obiectiv principal să stimuleze interesul și curiozitatea copiilor față de învățare, să dezvolte abilitățile cognitive și să ofere o experiență personalizată de învățare. Prin intermediul activităților interactive și jocurilor educaționale, TriviaGame motivează copiii să exploreze diferite domenii de cunoaștere și îi ajută să-și dezvolte gândirea logică, abilitățile de rezolvare a problemelor, atenția concentrată și memoria.

Pe viitor, se dorește extinderea funcționalităților aplicației prin adăugarea de activități interactive suplimentare, jocuri educaționale, module de evaluare și monitorizare a progresului, algoritmi de adaptare la nivelul de învățare al copiilor și un sistem de administrare a conturilor și datelor personale.

TriviaGame contribuie la formarea copiilor ca indivizi curioși, deschiși la cunoaștere și pregătiți pentru succesul academic și personal. Aplicația se angajează să fie un partener de încredere pentru părinți și educatori, sprijinind dezvoltarea cognitive, intelectuală și socială a copiilor. Prin intermediul acestei platforme educaționale inovatoare, TriviaGame contribuie la dezvoltarea copiilor într-un mod plăcut și eficient.

Bibliografie

1. Introducere în JavaFX. Disponibil: https://www.jetbrains.com/help/idea/javafx.html
2. Aplicații educaționale pentru copii. Disponibil: https://www.verywellfamily.com/best-educational-apps-for-kids-4842950
3. Ghid pentru crearea aplicațiilor educaționale pentru copii. Disponibil: https://yellow.systems/blog/how-to-create-an-educational-app
4. Tutorial în JavaFX. Disponibil: https://www.javatpoint.com/javafx-tutorial
5. Implementarea șablonului Singleton. Disponibil: https://www.javatpoint.com/singleton-design-pattern-in-java
6. Implementarea șablonului Abstract Factory. Disponibil: https://reactiveprogramming.io/blog/en/design-patterns/abstract-factory
7. Utilitatea șablonului Bridge. Disponibil: https://www.geeksforgeeks.org/bridge-design-pattern/
8. Implementarea șablonului Bridge. Disponibil: https://www.javatpoint.com/bridge-pattern
9. Utilizarea șablonului Proxy. Disponibil: https://www.tutorialspoint.com/design_pattern/proxy_pattern.htm
10. Implementarea șablonului Strategy. Disponibil: https://refactoring.guru/design-patterns/strategy
11. Diferența dintre șablonul Command și Strategy. Disponibil: https://interviewnoodle.com/what-exactly-is-the-command-pattern-and-how-is-it-different-from-the-strategy-pattern-6fc432f94ba6