Koszalin, 30.04.2015 r.

Dokument Detaliczny Projektu
Temat: Aplikacja Sudoku

Zespół projektowy:

Krzysztof Walczak
Marcin Srech
Aleksandra Pietraszewska
Robert Bryłowski
Paweł Świstacz
Jakub Potoniec

Prowadzący:

Dr inż. Walery Susłow

Streszczenie

Niniejszy dokument detaliczny projektu (DDP) opisuje detale pracy zespołu projektowego, który skupia się na stworzeniu aplikacji gry Sudoku w technologii Java. Pierwsza część dokumentu zawiera
wstęp opisujący ogólne założenia projektowe, a także wykorzystywane standardy i konwencje. Druga część opisuje specyfikacje poszczególnych komponentów. Wszystkie zmiany w dokumencie odnotowane będą w historii wersji (zamieszczona poniżej).

Historia dokumentu			
Wersja	**Opis modyfikacji**	**Autor modyfikacji**	**Data**
1.0	Wersja początkowa	Jakub Potoniec	30.04.2015

Spis treści

1. Opis ogólny

1.1 Wstęp

1.1.1 Cel

Niniejszy dokument precyzuje sposób realizowanych prac. Określa założenia projektu, standardy, narzędzia i komponenty wchodzące w skład implementacji, oraz opis realizacji tych komponentów.

1.1.2 Zakres

Założeniem projektu Sudoku jest stworzenie okienkowej aplikacji Sudoku. Aplikacja ma za zadanie umożliwiać użytkownikowi grę w sudoku wykorzystując do tego poprzez tworzenie planszy do gry i sprawdzanie poprawności wypełniania.

System tworzony jest w technologii Java, a do jego obsługi wystarczy komputer wykorzystujący tę technologię.

1.1.3 Definicje

- Gra Sudoku – aplikacja, która jest przedmiotem niniejszego projektu.
- Okno aplikacji – główne okno wyświetlające planszę gry oraz elementy interfejsu. Pojawia się po uruchomieniu aplikacji.
- Plansza gry – obszar głównego okna w którym są przedstawione pola do wypełnienia przez gracza.
- Cyfry gry – cyfry z przedziału od 1 do 9, którymi gracz uzupełnia planszę gry
- Rozpoczęcie nowej gry – wygenerowanie na planszy liczb początkowych.
- Podpowiedź – podświetlenie na planszy miejsc gdzie można umieścić wybraną cyfrę nie łamiąc zasad gry.
- Sprawdzenie gry – podświetlenie błędnych pól z cyframi, które łamią zasady gry.
- Czyszczenie gry – Wyczyszczenie planszy gry z liczb wypełnionych przez użytkownika.

1.1.4 Omówienie

Dokument ten powstał na podstawie specyfikacji wymagań systemowych. Zawiera on definicje standardów, strategii i konwencji które będą przestrzegane podczas realizacji projektu.

2. STANDARDY PROJEKTU, KONWENCJE, PROCEDURY

2.1. Standardy projektowe

Projekt aplikacji powstał w oparciu o paradygmat programowania obiektowego. Dziedzina problemu została przeanalizowana i zaprojektowana zgodnie z jego regułami. Dzięki temu oprogramowanie będzie łatwe w utrzymaniu i rozwoju, przez wzgląd na dostępność wielu narzędzi i wykwalifikowanych programistów.

2.2. Standardy dokumentacyjne

Dokumentacja projektu w sposób jednoznaczny określa jego strukturę logiczną i fizyczną. Wszystkie użyte pojęcia i skróty są wyjaśnione w odpowiednich miejscach. Specyfikacja wymagań jest zgodna ze standardem IEEE 830-1998. Diagramy zawarte w dokumentacji zostały sporządzone wg standardu UML 2.0. Wszystkie dokumenty dotyczące projektu są sporządzone na ustalonym szablonie firmowym. Kody źródłowe zawarte w dokumentacji są pisane czcionką o stałej szerokości.

2.3. Konwencje nazwowe

Przy projektowaniu aplikacji przyjęliśmy następujące konwencje dotyczące:

a) nazw klas:

- niedopuszczalne polskie znaki i słowa (nazwa w języku angielskim);
- nazwa rozpoczyna się wielką literą i każde następne w niej słowo również (np. UpdateAction);
- brak prefiksów;

b) nazw pól i metod w klasach

- nazwa w języku angielskim;
- niedopuszczalne polskie znaki;
- nazwa nie musi rozpoczynć się wielką literą, ale każde następne w niej słowo już tak (np. newGame);
- nazwa pola/metody musi określać zastosowanie

c) nazw zmiennych lokalnych

- dopuszczalne krótkie nazwy
- nazwa rozpoczyna się małą literą

d) klamry i wcięcia

- klamra otwierająca i zamykająca w osobnej linii,
- wcięcie w kodzie na każdym poziomie zagnieżdżenia

przykład:

```
public void checkGame() {
    selectedNumber = 0;
    for (int y = 0; y < 9; y++) {
        for (int x = 0; x < 9; x++)
            //jakiś kod
    }
    //jakiś kod
}
```

e) reszty kodu

- kod samodokumentujący się (intuicyjne nazwy zmiennych)

2.4. Standardy programistyczne

Aplikacja zostanie wykonana i będzie działać na platformie NetBeans. Do zbudowania architektury aplikacji posłuży framework NetBeans, opierający swoje działanie na wzorcu projektowym MVC. Pozwoli to stworzyć przejrzysty, łatwy do utrzymania kod, z rozdzieloną warstwą logiki biznesowej i prezentacji.

2.5. Narzędzia rozwijania oprogramowania

Środowiskiem RAD w którym powstanie aplikacja jest
NetBeans IDE 8.0.1,
Ponadto wykorzystane zostaną:

- serwer GitHub;

narzędzie do projektowania – aplikacje online ze stron creately.com, glify.com

- edytor tekstu - Microsoft Word 2003

3. Specyfikacja klas i metod

Klasa ButtonController

actionPerformed(ActionEvent e) – Kontrola przycisków

Klasa SudokuController

mousePressed(MouseEvent e) – Sprawdzenie które pole zostało wciśnięte
mouseClicked(MouseEvent e) – Obsługa zdarzeń myszy
mouseEntered(MouseEvent e) – Obsługa zdarzeń myszy
mouseExited(MouseEvent e) – Obsługa zdarzeń myszy
mouseReleased(MouseEvent e) – Obsługa zdarzeń myszy

Klasa Game

Game() – Konstruktor
Restart() – Zerowanie planszy
newGame() – Generowanie nowej gry
checkGame() – Porównanie wyboru gracza z tablicą rozwiązań
setHelp(boolean help) – Pomoc
setSelectedNumber(int selectedNumber) – Obsługa liczby wybranej przez gracza
getSelectedNumber() – Zwraca wybraną liczbę
isHelp() – Sprawdza czy pomoc jest włączona
isSelectedNumberCandidate(int x, int y) – Sprawdza czy liczba pasuje do pozycji
setNumber(int x, int y, int number) – Wpisuje liczbę do danej pozycji
getNumber(int x, int y) – Zwraca ‘x’ oraz ‘y’ pozycji
isCheckValid(int x, int y) – Sprawdza poprawność pozycji
isPossibleX(int[][] game, int y, int number) – Sprawdza poprawność na osi ‘x’

isPossibleY(int[][] game, int x, int number) – Sprawdza poprawność na osi ‘y’
isPossibleBlock(int[][] game, int x, int y, int number) – Sprawdza poprawność pozycji
getNextPossibleNumber(int[][] game, int x, int y, List<Integer> numbers) – Zwraca następną możliwą liczbę do generacji
generateSolution(int[][] game, int index) – Generowanie rozwiązania
generateGame(int[][] game) – Generowanie gry na podstawie rozwiązania
generateGame(int[][] game, List<Integer> positions) – Usunięcie losowych liczb
isValid(int[][] game) – Sprawdzenie poprawności gry
isValid(int[][] game, int index, int[] numberOfSolutions) – Sprawdza ilość rozwiązań
copy(int[][] game) – Tworzy kopię tablicy

Klasa ButtonPanel

ButtonPanel() – Tworzenie GUI
update(Observable o, Object arg) – Czyszczenie zaznaczenia przycisków
setController(ButtonController buttonController) – Przypisanie akcji do przycisków

Klasa Field

Field(int x, int y) – Rysuje linie planszy oraz ustawia czcionkę
setNumber(int number, boolean userInput) - Ustawia kolor liczb na planszy
getFieldX() – Zwraca ‘x’
getFieldY() – Zwraca ‘y’

Klasa Sudoku:

main(String[] args) – Klasa main

Klasa SudokuPanel:

SudokuPanel() – Graficzne generowanie pola gry
update(Observable o, Object arg) – Kontrola gry
setGame(Game game) – Ustawianie koloru tła pól oraz wpisywanie liczb
setGameCheck(Game game) – Obsługa zmiany koloru tła pól podczas sprawdzania
setCandidates(Game game) – Obsługa zmiany koloru tła pól przy używaniu pomocy
setController(SudokuController sudokuController) – Interfejs słuchacza myszy