МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

Тема: Пятнашки

Студент гр. 7303 _____ Никитенко Д.А.

Студент гр. 7303 _____ Шестопалов Р.П.

Студент гр. 7303 _____ Ермолаев Д.В.

Руководитель _____ Ефремов М.А.

Санкт-Петербург

2019

ЗАДАНИЕ
НА УЧЕБНУЮ ПРАКТИКУ

Студент Никитенко Д.А. группы 7303

Студент Шестопалов Р.П. группы 7303

Студент Ермолаев Д.В. группы 7303

Тема практики: выполнение мини-проекта на языке java.

Задание на практику:

Командная итеративная разработка игры «Пятнашки» на Java с графическим интерфейсом.

Алгоритм: A*.

Сроки прохождения практики: 01.07.2019 – 14.07.2019

Дата сдачи отчета: 09.07.2019

Дата защиты отчета: 10.07.2019

Студент _____ Никитенко Д.А.

Студент _____ Шестопалов Р.П.

Студент _____ Ермолаев Д.В.

Руководитель _____ Ефремов М.А.

АННОТАЦИЯ

В ходе выполнения данной работы был реализован алгоритм A* для автоматического поиска решения в игре «Пятнашки». Практическое задание состоит из 4 частей. В первой части расписаны требования к программе (игре) в течении выполнения практического задания. Во второй описано распределение работы на группу студентов и план выполнения работы. В третьей части рассматриваются методы решения задачи и используемые структуры данных. В четвертую часть включено тестирование элементов программы. Также сделано заключение, описаны используемая литература.

SUMMARY

In the course of this work, the A* algorithm was implemented. The practical task consists of 4 parts. The first part describes the requirements for the program during the practical task. The second describes the responsibilities of students for work and the work plan. The third part deals with the methods of implementation of the task and the data structures used. The forth part includes testing of the program, a conclusion, describes the source literature.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель задачи – создание мини-проекта, который реализует алгоритм A* для автоматического решения игры «Пятнашки». В задаче требуется реализовать диаграмму всех использующихся классов, разработать алгоритм решения задачи и интерфейс к алгоритму. Также необходимо связать исходный код с интерфейсом и написать тесты к логике программы. Далее необходимо создать use-case диаграмму для данной программы в нотации UML.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные требования к программе

1.1.1. На рисунке 1 изображена use-case диаграмма. На старте пользователь может начать новую игру или закрыть программу. После выбора «Новой игры» появляется окно с доступными уровнями. Кликнув по картинке с уровнем, пользователь начнет игру. В самом уровне можно решить головоломку самому или при помощи алгоритма (автоматически).

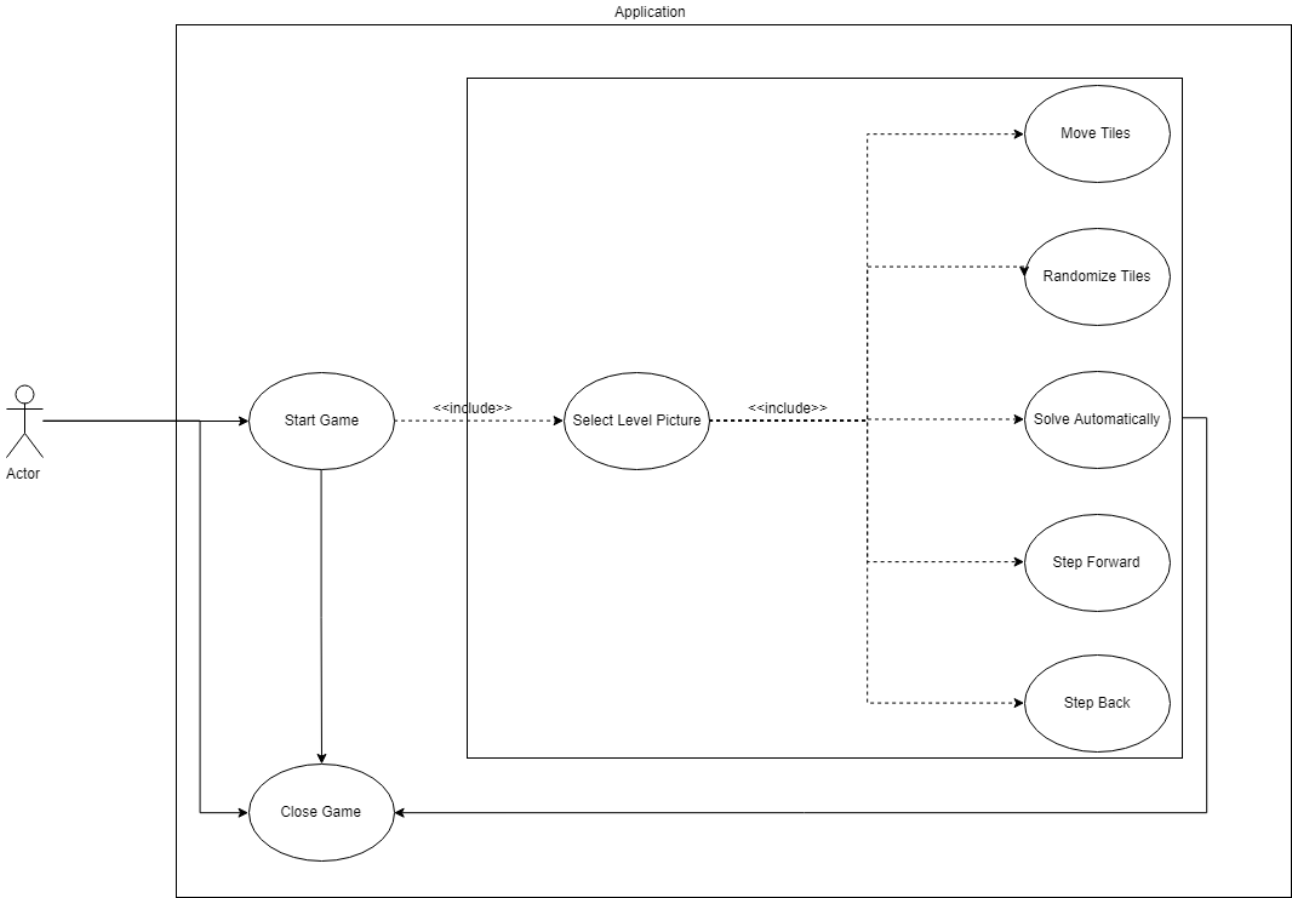

Рисунок 1 – Use-case диаграмма

1.2. Уточнение требований после сдачи прототипа.

1.2.1. Заполнить файл с темой задания

1.2.2. Сделать диаграмму классов.

1.2.3. Исправить недочеты в use-case диаграмме.

1.3. Уточнение требований после сдачи 1-ой версии.

1.3.1. Реализовать сборку проекта с помощью maven.

1.3.2. Сделать unit-тесты для модели.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

до 04.07: use case диаграмма, интерфейс без реализации логики.

до 06.07: прототип с демонстрацией функциональности программы.

до 08.07: рабочая версия со сборкой maven.

до 10.07: исправление недочетов первой версии.

2.2. Распределение ролей в бригаде

Ермолаев Д.В.: базовые классы, логика алгоритма, use-case диаграмма.

Шестопалов Р.П.: контроллер и юнит-тестирование, отчет, сборка на maven.

Никитенко Д.А.: GUI, соединение интерфейса и логики программы.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Использованные структуры данных

На рисунке 2 показана иерархия логики классов. Класс Wrap выполняет роль координатора между UI и состоянием игры. В нем содержиться объект класса Puzzle, с которым идет взаимодействие программы.

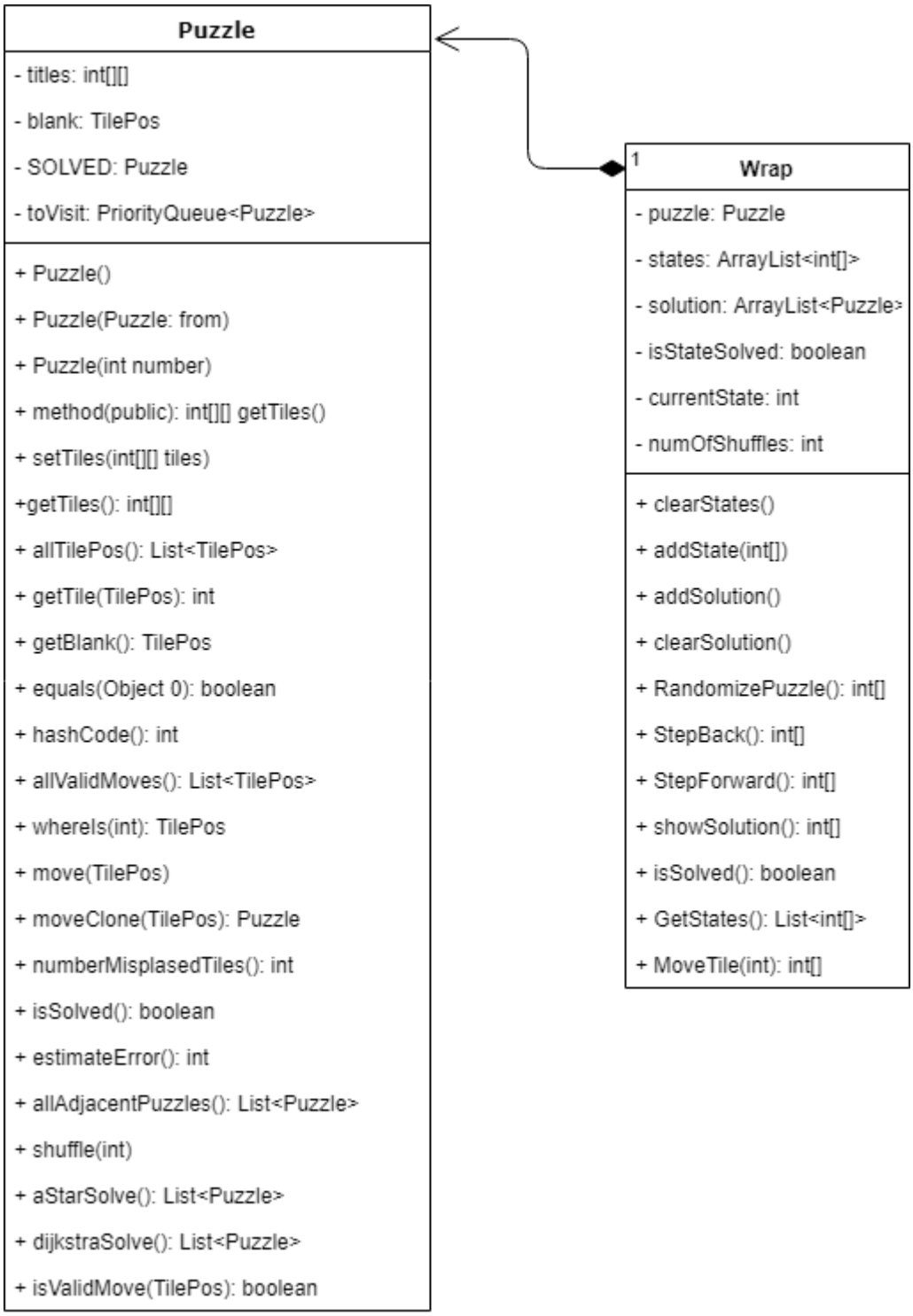

Рисунок 2 – классы логики программы.

3.2. Основные методы

1) Puzzle() – инициализирует начальное состояние головоломки

2) Puzzle(int) – конструктор с вызовом функции shuffle(int)

3) equals(Object) – переопределенный метод сравнения

4) hashcode() – переопределенный метод хэш функции

5) shuffle(int) – функция, выполняющая N случайный перемещений пустой ячейки пазла

6) isSolved() – метод, который проверяет состояние решения головоломки

7) aStarSolve() – алгоритм A*, возвращающий путь решения от текущего состояния игры

8) dijkstraSolve() – алгоритм Дийкстры, возвращающий путь решения от текущего состояния игры

3.3. Использование интерфейса

На следующих рисунках (рисунок 3, рисунок 4, рисунок5) предствлен интерфейс программы. На рисунке 3 – главное мен. На рисунке 4 – меню выбора уровня. На рисунке 5 – поле с головоломкой

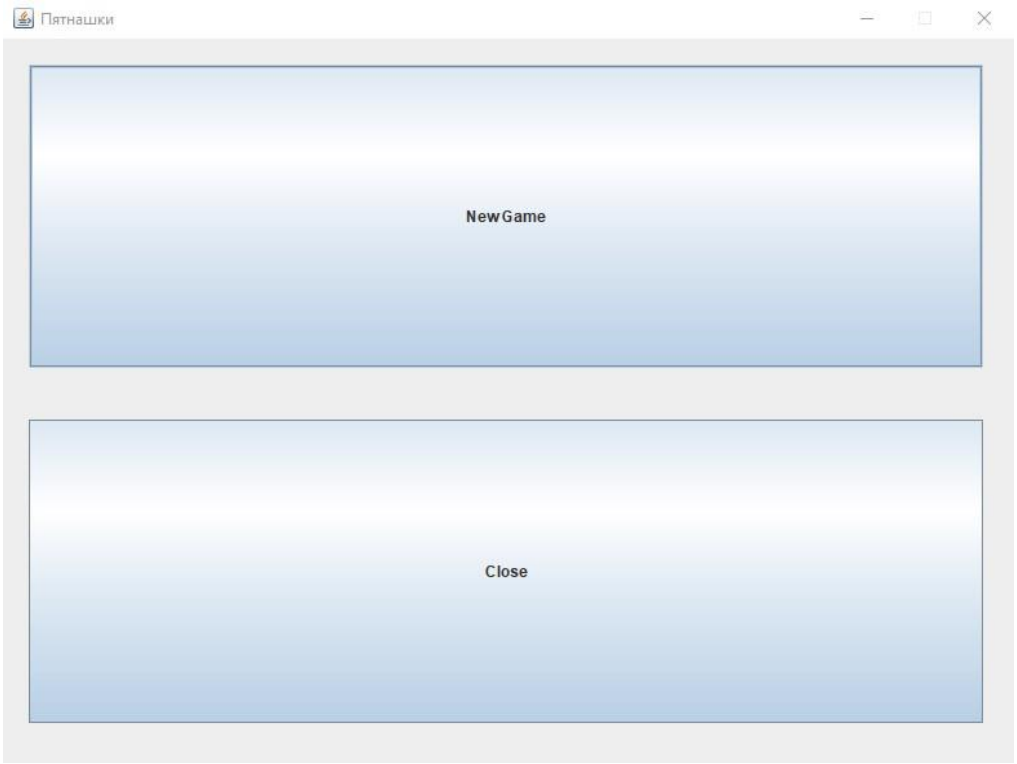

Рисунок 3 – интерфейс стартового меню программы

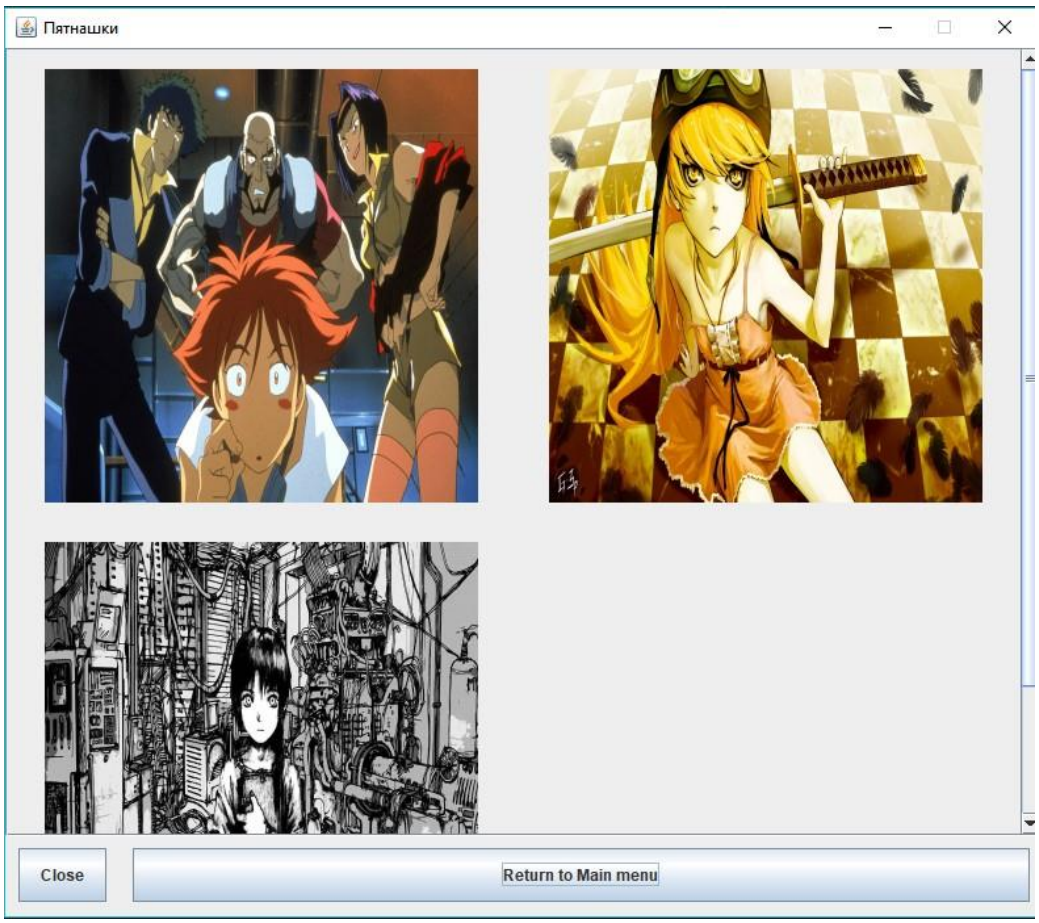

Рисунок 4 – меню выбора картинки

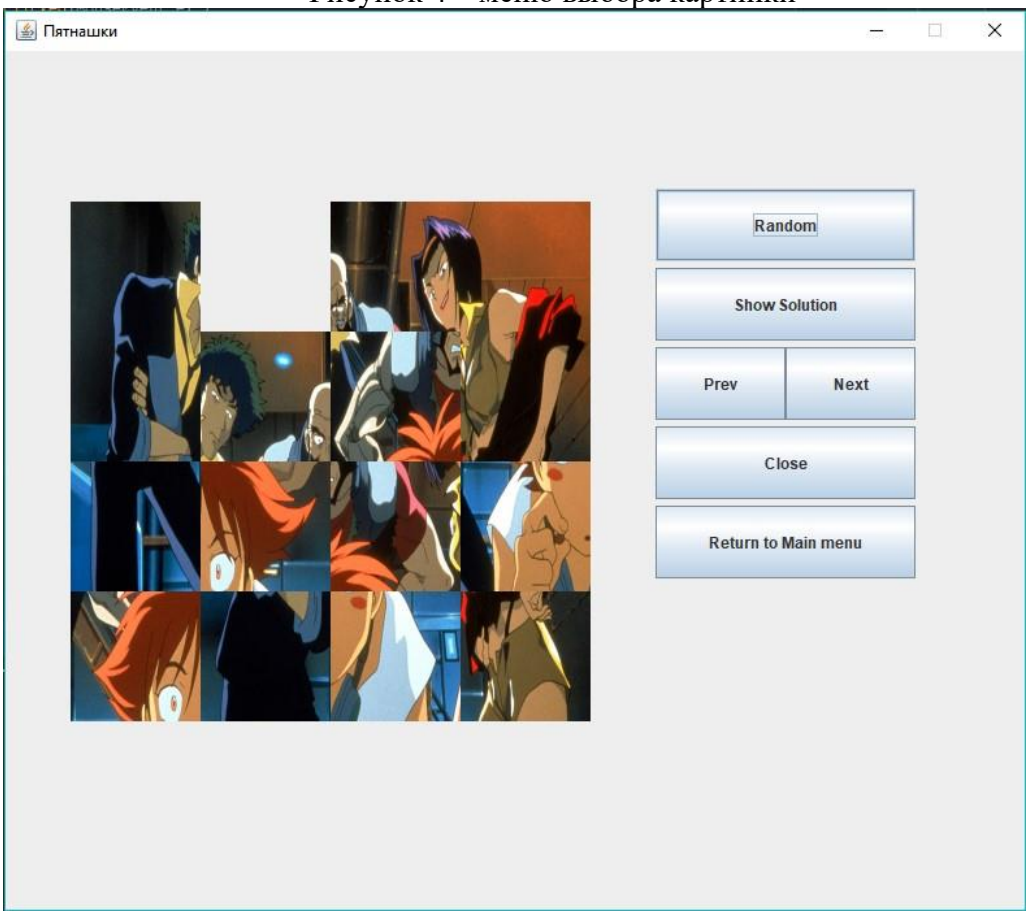

Рисунок 5 – игровое поле

4. ТЕСТИРОВАНИЕ

4.1. Тестирование кода алгоритма

Создается класс PuzzleTest для создания и хранения модульных тестов.

В методе *setUp()* создается объект класса Puzzle для использования во всех модульных тестах.

4 метода move() (move1(), move2() и т.д.) проверяют возможность перемещения ячейки (далее тайлов).

2 метода shuffle() проверяют правильность метода shuffle, т.е. невозможность при перемешивании иконок выставления этих самых иконок сразу в верное положение.

2 метода isValidMove() проверяют доступность перемещения для тайла.

3 метода isNotValidMove() проверяют недоступность перемещения для тайла.

Метод puz() проверяет работу конструктора Puzzle().

2 метода estimateError() проверяют количество неверно расставленных тайлов.

Метод getTile() проверяют верность возвращаемого значения методом getTile().

ЗАКЛЮЧЕНИЕ

В ходе разработки мини-проекта в нотации UML были созданы use-case диаграмма и диаграмма классов, по которым была построена модель решения задачи - класс, хранящий данные о состоянии головоломки и класс, вносящий изменения в эти данные по сигналам от UI. Пользовательский интерфейс "Пятнашек" был реализован при помощи библиотеки Swing. Сборка проекта осуществлялась фреймворком Apache Maven. В итоге после тестирования были получены полноценная реализация игры "Пятнашки" с выбором уровня, а также опыт разработки на языке Java.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кэти Сьерра и Берт Бейтс Изучаем Java: Москва, 2016, 383-428с.

2. Документация Java™ Platform, Standard Edition 7 API Specification [Электронный ресурс] // Copyright © 1993, 2018, Oracle and/or its affiliates. URL: https://docs.oracle.com/javase/7/docs/api/index.html 04.07.2019

3. Руководство по maven – что такое maven [Электронный ресурс], Apache Maven Project. URL: www.apache-maven.ru 03.07.2019

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

MAIN.JAVA

```
IMPORT COM.PRACTICE.BLUETEAM.UI.MAINWINDOW;

IMPORT JAVAX.SWING.*;
IMPORT JAVA.AWT.EVENT.KEYADAPTER;
IMPORT JAVA.AWT.EVENT.KEYEVENT;

PUBLIC CLASS MAIN {
    PUBLIC STATIC VOID MAIN(STRING[] ARGS) {
        MAINWINDOW UI = MAINWINDOW.GETUI();
        UI.SHOWUI();
    }
}
```

PUZZLE JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.ALGO;

IMPORT JAVA.UTIL.ARRAYLIST;
IMPORT JAVA.UTIL.COMPARATOR;
IMPORT JAVA.UTIL.HASHMAP;
IMPORT JAVA.UTIL.LINKEDLIST;
IMPORT JAVA.UTIL.LIST;
IMPORT JAVA.UTIL.PRIORITYQUEUE;
IMPORT JAVA.UTIL.QUEUE;

PUBLIC CLASS PUZZLE {

    PUBLIC CLASS TILEPOS {
        PUBLIC INT X, Y;

        PUBLIC TILEPOS (INT X, INT Y) {
```

```
        THIS.X = X;
        THIS.Y = Y;
    }
}

PRIVATE INT[][] TILES;
PRIVATE TILEPOS BLANK;

PUBLIC PUZZLE () {
    SETTILES(NEW INT[4][4]);
    INT NUM = 0;
    FOR(INT I = 0; I < 4; I++) {
        FOR(INT J = 0; J < 4; J++) {
            GETTILES()[I][J] = NUM;
            NUM++;
        }
    }
    BLANK = NEW TILEPOS(3,3);
}

PUBLIC PUZZLE (PUZZLE FROM) {
    THIS();
    FOR (TILEPOS P: ALLTILEPOS()) {
        GETTILES()[P.X][P.Y] = FROM.GETTILE(P);
    }
    BLANK = FROM.GETBLANK();
}

PUBLIC PUZZLE (INT NUMBER) {
    THIS();
    THIS.SHUFFLE(NUMBER);
}

PUBLIC FINAL STATIC PUZZLE SOLVED = NEW PUZZLE();
```

```
PUBLIC INT[][] GETTILES() {
    RETURN TILES;
}

PUBLIC VOID SETTILES(INT[][] TILES) {
    THIS.TILES = TILES;
    THIS.BLANK = WHEREIS(15);
}

PUBLIC LIST<TILEPOS> ALLTILEPOS() {
    ARRAYLIST<TILEPOS> LIST = NEW ARRAYLIST<TILEPOS>();
    FOR(INT I = 0; I < 4; I++) {
        FOR(INT J = 0; J < 4; J++) {
            LIST.ADD(NEW TILEPOS(I,J));
        }
    }

    RETURN LIST;
}

PUBLIC INT GETTILE(TILEPOS P) {
    RETURN GETTILES()[P.X][P.Y];
}

PUBLIC TILEPOS GETBLANK() {
    RETURN WHEREIS(15);
}

@OVERRIDE
PUBLIC BOOLEAN EQUALS(OBJECT O) {
    IF(O INSTANCEOF PUZZLE) {
        FOR(TILEPOS P: ALLTILEPOS()) {
            IF( THIS.GETTILE(P) != ((PUZZLE) O).GETTILE(P)) {
                RETURN FALSE;
            }
```

```
            }
            RETURN TRUE;
        }
        RETURN FALSE;
    }

    @OVERRIDE
    PUBLIC INT HASHCODE() {
        INT OUT=0;
        FOR(TILEPOS P: ALLTILEPOS()) {
            OUT= (OUT*4*4) + THIS.GETTILE(P);
        }
        RETURN OUT;
    }

    PUBLIC LIST<TILEPOS> ALLVALIDMOVES() {
        ARRAYLIST<TILEPOS> OUT = NEW ARRAYLIST<TILEPOS>();
        FOR(INT DX = -1; DX < 2; DX++) {
            FOR(INT DY = -1; DY < 2; DY++) {
                TILEPOS TP = NEW TILEPOS(BLANK.X + DX, BLANK.Y +
DY);
                IF( ISVALIDMOVE(TP) ) {
                    OUT.ADD(TP);
                }
            }
        }
        RETURN OUT;
    }

    PUBLIC TILEPOS WHEREIS(INT A) {
        FOR(TILEPOS P: ALLTILEPOS()) {
            IF(GETTILE(P) == A) {
                RETURN P;
            }
        }
```

```
        RETURN NULL;
    }

    PUBLIC BOOLEAN ISVALIDMOVE(TILEPOS P) {
        IF (P == NULL) {
            RETURN FALSE;
        }
        IF( ( P.X < 0) || (P.X >= 4) ) {
            RETURN FALSE;
        }
        IF( ( P.Y < 0) || (P.Y >= 4) ) {
            RETURN FALSE;
        }
        INT DX = BLANK.X - P.X;
        INT DY = BLANK.Y - P.Y;
        IF( (MATH.ABS(DX) + MATH.ABS(DY) != 1 ) || (DX*DY != 0) )
{
            RETURN FALSE;
        }
        RETURN TRUE;
    }

    PUBLIC VOID MOVE(TILEPOS P) {
        IF (ISVALIDMOVE(P)) {
            GETTILES()[BLANK.X][BLANK.Y] = GETTILES()[P.X][P.Y];
            GETTILES()[P.X][P.Y] = 15;
            BLANK = P;
        }
    }

    PUBLIC PUZZLE MOVECLONE(TILEPOS P) {
        PUZZLE OUT = NEW PUZZLE(THIS);
        OUT.MOVE(P);
        RETURN OUT;
    }
```

```
    PUBLIC INT NUMBERMISPLACEDTILES() {
        INT WRONG = 0;
        FOR(INT I = 0; I < 4; I++) {
            FOR(INT J = 0; J < 4; J++) {
                IF( (GETTILES()[I][J] < 15) && ( GETTILES()[I][J]
!= SOLVED.GETTILES()[I][J] ) ){
                    WRONG++;
                }
            }
        }
        RETURN WRONG;
    }

    PUBLIC BOOLEAN ISSOLVED() {
        RETURN NUMBERMISPLACEDTILES() == 0;
    }

    PUBLIC INT ESTIMATEERROR() {
        RETURN THIS.NUMBERMISPLACEDTILES();
    }

    PUBLIC LIST<PUZZLE> ALLADJACENTPUZZLES() {
        ARRAYLIST<PUZZLE> OUT = NEW ARRAYLIST<PUZZLE>();
        FOR (TILEPOS MOVE : ALLVALIDMOVES()) {
            OUT.ADD(MOVECLONE(MOVE));
        }
        RETURN OUT;
    }

    PUBLIC VOID SHUFFLE(INT NUMBER) {
        FOR(INT I = 0; I < NUMBER; I++) {
            LIST<TILEPOS> POSSIBLE = ALLVALIDMOVES();
            INT WHICH =  (INT) (MATH.RANDOM() * POSSIBLE.SIZE());
            TILEPOS MOVE = POSSIBLE.GET(WHICH);
```

```
            THIS.MOVE(MOVE);
        }
    }

    PUBLIC LIST<PUZZLE> ASTARSOLVE() {
        HASHMAP<PUZZLE, PUZZLE> PREDCESSOR = NEW HASHMAP<PUZZLE,
PUZZLE>();
        HASHMAP<PUZZLE, INTEGER> DEPTH = NEW HASHMAP<PUZZLE,
INTEGER>();
        FINAL HASHMAP<PUZZLE, INTEGER> SCORE = NEW HASHMAP<PUZZLE,
INTEGER>();
        COMPARATOR<PUZZLE> COMPARATOR = NEW COMPARATOR<PUZZLE>() {
            @OVERRIDE
            PUBLIC INT COMPARE(PUZZLE A, PUZZLE B) {
                RETURN SCORE.GET(A) - SCORE.GET(B);
            }
        };
        PRIORITYQUEUE<PUZZLE> TOVISIT = NEW
PRIORITYQUEUE<PUZZLE>(10000, COMPARATOR);

        PREDCESSOR.PUT(THIS, NULL);
        DEPTH.PUT(THIS, 0);
        SCORE.PUT(THIS, THIS.ESTIMATEERROR());
        TOVISIT.ADD(THIS);
        INT CNT = 0;
        WHILE (TOVISIT.SIZE() > 0) {
            PUZZLE CANDIDATE = TOVISIT.REMOVE();
            CNT++;
            IF( CANDIDATE.ISSOLVED() ) {
                LINKEDLIST<PUZZLE> SOLUTION = NEW
LINKEDLIST<PUZZLE>();
                PUZZLE BACKTRACE = CANDIDATE;
                WHILE( BACKTRACE != NULL ) {
                    SOLUTION.ADDFIRST(BACKTRACE);
                    BACKTRACE = PREDCESSOR.GET(BACKTRACE);
```

```
                }
                RETURN SOLUTION;
            }
            FOR(PUZZLE FP: CANDIDATE.ALLADJACENTPUZZLES()) {
                IF( !PREDCESSOR.CONTAINSKEY(FP) ) {
                    PREDCESSOR.PUT(FP, CANDIDATE);
                    DEPTH.PUT(FP, DEPTH.GET(CANDIDATE)+1);
                    INT ESTIMATE = FP.ESTIMATEERROR();
                    SCORE.PUT(FP, DEPTH.GET(CANDIDATE) + 1 +
ESTIMATE);
                    TOVISIT.ADD(FP);
                }
            }
        }
        RETURN NULL;
    }

    PUBLIC LIST<PUZZLE> DIJKSTRASOLVE() {
        QUEUE<PUZZLE> TOVISIT = NEW LINKEDLIST<PUZZLE>();
        HASHMAP<PUZZLE,PUZZLE> PREDECESSOR = NEW
HASHMAP<PUZZLE,PUZZLE>();
        TOVISIT.ADD(THIS);
        PREDECESSOR.PUT(THIS, NULL);
        INT CNT = 0;
        WHILE( TOVISIT.SIZE() > 0) {
            PUZZLE CANDIDATE = TOVISIT.REMOVE();
            CNT++;
            IF( CANDIDATE.ISSOLVED() ) {
                LINKEDLIST<PUZZLE> SOLUTION = NEW
LINKEDLIST<PUZZLE>();
                PUZZLE BACKTRACE=CANDIDATE;
                WHILE( BACKTRACE != NULL ) {
                    SOLUTION.ADDFIRST(BACKTRACE);
                    BACKTRACE = PREDECESSOR.GET(BACKTRACE);
                }
```

```
                RETURN SOLUTION;
            }
            FOR(PUZZLE FP: CANDIDATE.ALLADJACENTPUZZLES()) {
                IF( !PREDECESSOR.CONTAINSKEY(FP) ) {
                    PREDECESSOR.PUT(FP,CANDIDATE);
                    TOVISIT.ADD(FP);
                }
            }
        }
        RETURN NULL;
    }

}
```

WRAP.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.ALGO;

IMPORT JAVA.UTIL.ARRAYLIST;
IMPORT JAVA.UTIL.LIST;

PUBLIC CLASS WRAP {

    PRIVATE STATIC PUZZLE PUZZLE = NEW PUZZLE();

    PRIVATE STATIC ARRAYLIST<INT[]> STATES = NEW
ARRAYLIST<INT[]>();
    PRIVATE STATIC ARRAYLIST<PUZZLE> SOLUTION = NULL;
    PRIVATE STATIC BOOLEAN ISSTATESOLVED = FALSE;

    PRIVATE STATIC INT CURRENTSTATE = -1;

    PRIVATE STATIC FINAL INT NUMOFSHUFFLES = 25;

    PUBLIC STATIC INT[] SHOWSOLUTION() {
```

```
    PUZZLE = NEW PUZZLE();
    IF (SOLUTION != NULL) { CLEARSOLUTION(); }
    CLEARSTATES();
    INT[] TEMP = GETTILESARRAY(PUZZLE);
    ADDSTATE(TEMP);
    RETURN TEMP;
}

PRIVATE STATIC INT[] GETTILESARRAY(PUZZLE PUZZLE) {
    INT[] OUT = NEW INT[16];
    INT C = 0;
    FOR (INT I = 0; I < 4; I++) {
        FOR (INT J = 0; J < 4; J++) {
            OUT[C] = PUZZLE.GETTILES()[I][J];
            C++;
        }
    }
    RETURN OUT;
}

PRIVATE STATIC VOID CLEARSTATES() {
    STATES.CLEAR();
    STATES.TRIMTOSIZE();
    CURRENTSTATE = -1;
}

PRIVATE STATIC VOID ADDSTATE (INT[] STATE) {
    STATES.ADD(STATE);
    CURRENTSTATE++;
}

PRIVATE STATIC VOID ADDSOLUTION() {
    WHILE (SOLUTION == NULL) {

        SOLUTION = NEW ARRAYLIST<PUZZLE>(PUZZLE.ASTARSOLVE());
```

```
            IF (SOLUTION == NULL) {
                SOLUTION.TRIMTOSIZE();
                SOLUTION = NEW
ARRAYLIST<PUZZLE>(PUZZLE.DIJKSTRASOLVE());
            }
        }
        ISSTATESOLVED = TRUE;
;    }

    PRIVATE STATIC VOID CLEARSOLUTION() {
        SOLUTION.CLEAR();
        SOLUTION.TRIMTOSIZE();
        SOLUTION = NULL;
        ISSTATESOLVED = FALSE;
    }

    PUBLIC STATIC INT[] RANDOMIZEPUZZLE() {
        PUZZLE = NEW PUZZLE(NUMOFSHUFFLES);
        CLEARSTATES();
        IF (SOLUTION != NULL) {CLEARSOLUTION();}
        INT[] TEMP = GETTILESARRAY(PUZZLE);
        ADDSTATE(TEMP);
        RETURN TEMP;
    }

    PUBLIC STATIC VOID SETPUZZLE (INT[] DATA) {
        IF (SOLUTION != NULL) { CLEARSOLUTION(); }
        CLEARSTATES();
        INT[][] BUFF = NEW INT[4][4];
        INT N = 0;
        FOR (INT I = 0; I < 4; I++) {
            FOR (INT J = 0; J < 4; J++) {
                BUFF[I][J] = DATA[N];
                N++;
```

```
            }
        }
        PUZZLE.SETTILES(BUFF);
        INT[] TMP = GETTILESARRAY(PUZZLE);
        ADDSTATE(TMP);
    }

    PUBLIC STATIC INT[] MOVETILE (INT NUMOFTILE) {
        IF (SOLUTION != NULL) { CLEARSOLUTION(); }
        IF (CURRENTSTATE == STATES.SIZE() - 1){
            IF (!PUZZLE.ISVALIDMOVE(PUZZLE.WHEREIS(NUMOFTILE)))
                RETURN STATES.GET(CURRENTSTATE);
            PUZZLE.MOVE(PUZZLE.WHEREIS(NUMOFTILE));
        } ELSE {
            INT[][] BUFF = NEW INT[4][4];
            INT N = 0;
            INT[] S = STATES.GET(CURRENTSTATE);
            FOR (INT I = 0; I < 4; I++) {
                FOR (INT J = 0; J < 4; J++) {
                    BUFF[I][J] = S[N];
                    N++;
                }
            }
            PUZZLE TEST = NEW PUZZLE();
            TEST.SETTILES(BUFF);
            IF (!TEST.ISVALIDMOVE(TEST.WHEREIS(NUMOFTILE)))
                RETURN STATES.GET(CURRENTSTATE);
            STATES.REMOVEALL(STATES.SUBLIST(CURRENTSTATE + 1,
STATES.SIZE()));
            STATES.TRIMTOSIZE();
            PUZZLE.SETTILES(BUFF);
            PUZZLE.MOVE(PUZZLE.WHEREIS(NUMOFTILE));
        }
        INT[] TEMP = GETTILESARRAY(PUZZLE);
        ADDSTATE(TEMP);
```

```
    RETURN TEMP;
}

PUBLIC STATIC INT[] STEPBACK() {
    IF(CURRENTSTATE > 0) {
        CURRENTSTATE--;
        RETURN STATES.GET(CURRENTSTATE);
    } ELSE {
        RETURN STATES.GET(0);
    }
}

PUBLIC STATIC INT[] STEPFORWARD() {
    IF (CURRENTSTATE < STATES.SIZE() - 1) {
        CURRENTSTATE++;
        RETURN STATES.GET(CURRENTSTATE);
    } ELSE {
        IF (PUZZLE.ISSOLVED()) {
            RETURN STATES.GET(CURRENTSTATE);
        }
        IF (!ISSTATESOLVED) {
            ADDSOLUTION();
        }

        PUZZLE TEMP = NEW PUZZLE(SOLUTION.REMOVE(0));
        WHILE (PUZZLE.EQUALS(TEMP)) {
            TEMP = NEW PUZZLE(SOLUTION.REMOVE(0));
        }
        SOLUTION.TRIMTOSIZE();
        PUZZLE = NEW PUZZLE(TEMP);
        INT[] S = GETTILESARRAY(PUZZLE);
        ADDSTATE(S);
        CURRENTSTATE = STATES.SIZE() - 1;
        RETURN S;
    }
```

```
    }

    PUBLIC STATIC BOOLEAN ISSOLVED() {
        INT[][] BUFF = NEW INT[4][4];
        INT N = 0;
        INT[] S = STATES.GET(CURRENTSTATE);
        FOR (INT I = 0; I < 4; I++) {
            FOR (INT J = 0; J < 4; J++) {
                BUFF[I][J] = S[N];
                N++;
            }
        }
        PUZZLE TEST = NEW PUZZLE();
        TEST.SETTILES(BUFF);
        IF (TEST.ISSOLVED()) {
//            SOLUTION.CLEAR();
//            SOLUTION = NULL;
            RETURN TRUE;
        } ELSE {
            RETURN FALSE;
        }
    }

    PUBLIC STATIC LIST<INT[]> GETSTATES() {
        RETURN STATES;
    }

}
```

DATABASE.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.DATABASE;

IMPORT COM.PRACTICE.BLUETEAM.ALGO.PUZZLE;
IMPORT COM.PRACTICE.BLUETEAM.ALGO.WRAP;
```

```
IMPORT COM.PRACTICE.BLUETEAM.GAMESTATE.GAMESTATE;
IMPORT COM.PRACTICE.BLUETEAM.UI.LEVELSELECTWINDOW;
IMPORT COM.PRACTICE.BLUETEAM.UI.LEVELWINDOW;
IMPORT COM.PRACTICE.BLUETEAM.UI.SECRETLEVEL;
IMPORT COM.PRACTICE.BLUETEAM.UI.TILE;

IMPORT JAVAX.IMAGEIO.IMAGEIO;
IMPORT JAVAX.SWING.*;
IMPORT JAVAX.XML.CRYPTO.ALGORITHMMETHOD;
IMPORT JAVAX.XML.CRYPTO.DATA;
IMPORT JAVA.AWT.IMAGE.BUFFEREDIMAGE;
IMPORT JAVA.IO.FILE;
IMPORT JAVA.IO.IOEXCEPTION;
IMPORT JAVA.NET.URL;

// ИСПОЛЬЗУЕТСЯ СИНГЛТОН, ЧТОБЫ НА ПРОТИЖЕНИИ ВСЕГО ПРОЦЕССА ВЕЗДЕ
ИСПОЛЬЗОВАЛАСЬ ОДНА БАЗА ДАННЫХ
PUBLIC CLASS DATABASE {
    PRIVATE STATIC DATABASE OURINSTANCE = NEW DATABASE();
    PUBLIC STATIC DATABASE GETINSTANCE() {
        RETURN OURINSTANCE;
    }

    PUBLIC STATIC BOOLEAN GETISSOLVED(INT INDEX) {
        RETURN ISSOLVED[INDEX];
    }
    PUBLIC STATIC BOOLEAN GETISSOLVED(){
        RETURN ISSECRETSOLVED;
    }
    PRIVATE  STATIC BOOLEAN ISSECRETSOLVED;
    PRIVATE STATIC BOOLEAN[] ISSOLVED;
    // API ДЛЯ РАБОТЫ С АЛГОРИТМОМ
    PUBLIC STATIC VOID RANDOMIZEPUZZLE(INT INDEX) {
        SETTILESORDERONTHESCREEN(WRAP.RANDOMIZEPUZZLE(), INDEX);
        LEVELWINDOWS[INDEX].SETTILES();
```

```
        IF (WRAP.ISSOLVED()) {
            ISSOLVED[INDEX] = TRUE;
        }
        ELSE {
            ISSOLVED[INDEX] = FALSE;
        }
    }

    PUBLIC STATIC VOID NEXTSTEP(INT INDEX) {
        SETTILESORDERONTHESCREEN(WRAP.STEPFORWARD(),INDEX);
        LEVELWINDOWS[INDEX].SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSOLVED[INDEX] = TRUE;
        }
        ELSE {
            ISSOLVED[INDEX] = FALSE;
        }
    }

    PUBLIC STATIC VOID PREVSTEP(INT INDEX) {
        SETTILESORDERONTHESCREEN(WRAP.STEPBACK(),INDEX);
        LEVELWINDOWS[INDEX].SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSOLVED[INDEX] = TRUE;
        }
        ELSE {
            ISSOLVED[INDEX] = FALSE;
        }
    }

    PUBLIC STATIC VOID MOVE (INT NUMBEROFTILE, INT INDEX){
        SETTILESORDERONTHESCREEN(WRAP.MOVETILE(NUMBEROFTILE),
INDEX);
        LEVELWINDOWS[INDEX].SETTILES();
        IF (WRAP.ISSOLVED()) {
```

```
            GAMESTATE.SETISAVAILABLE(INDEX+1);
            LEVELSELECTWINDOW.UPDATE();
            ISSOLVED[INDEX] = TRUE;
        }
        ELSE {
            ISSOLVED[INDEX] = FALSE;
        }
    }

    PUBLIC STATIC VOID AUTOSOLVE() {
        SETSECRETTILESORDERONTHESCREEN(WRAP.SHOWSOLUTION());
        SECRETLEVEL.SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSECRETSOLVED = TRUE;
        }
        ELSE {
            ISSECRETSOLVED = FALSE;
        }
    }

    PUBLIC STATIC VOID AUTOSOLVE(INT INDEX) {
        SETTILESORDERONTHESCREEN(WRAP.SHOWSOLUTION(), INDEX);
        LEVELWINDOWS[INDEX].SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSOLVED[INDEX] = TRUE;
        }
        ELSE {
            ISSOLVED[INDEX] = FALSE;
        }
    }

    PUBLIC STATIC VOID MOVE (INT NUMBEROFTILE){
//
WRAP.SETPUZZLE(CONVERTILESINTOINT(GETTILESORDERONTHESCREEN(INDEX))
);
```

```
SETSECRETTILESORDERONTHESCREEN(WRAP.MOVETILE(NUMBEROFTILE));
        SECRETLEVEL.SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSECRETSOLVED = TRUE;
        }
        ELSE {
            ISSECRETSOLVED = FALSE;
        }
    }

    PUBLIC STATIC VOID RANDOMIZESECRETPUZZLE() {
        SETSECRETTILESORDERONTHESCREEN(WRAP.RANDOMIZEPUZZLE());
        SECRETLEVEL.SETTILES();
    }

    PUBLIC STATIC VOID SECRETNEXTSTEP() {
        SETSECRETTILESORDERONTHESCREEN(WRAP.STEPFORWARD());
        SECRETLEVEL.SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSECRETSOLVED = TRUE;
        }
        ELSE {
            ISSECRETSOLVED = FALSE;
        }
    }

    PUBLIC STATIC VOID SECRETPREVSTEP() {
        SETSECRETTILESORDERONTHESCREEN(WRAP.STEPBACK());
        SECRETLEVEL.SETTILES();
        IF (WRAP.ISSOLVED()) {
            ISSECRETSOLVED = TRUE;
        }
        ELSE {
            ISSECRETSOLVED = FALSE;
```

```
        }
    }

    // КОЛИЧЕСТВО УРОВНЕЙ
    PRIVATE STATIC INT NUMBEROFLEVELS;
    PUBLIC STATIC INT GETNUMBEROFLEVELS() {
        RETURN NUMBEROFLEVELS;
    }
    PUBLIC STATIC VOID SETNUMBEROFLEVELS(INT NUMBEROFLEVELS) {
        DATABASE.NUMBEROFLEVELS = NUMBEROFLEVELS;
    }
    // ТЕКУЩИЙ УРОВЕНЬ
//    PRIVATE STATIC INT CURRENTLEVEL;
//    PUBLIC STATIC INT GETCURRENTLEVEL() {
//        RETURN CURRENTLEVEL;
    //    }
    // ИКОНКИ УРОВНЕЙ
//    PUBLIC STATIC VOID SETCURRENTLEVEL(INT CURRENTLEVEL) {
//        DATABASE.CURRENTLEVEL = CURRENTLEVEL;
//    }
    PRIVATE STATIC IMAGEICON[] LEVELICONS;
    PUBLIC STATIC IMAGEICON[] GETLEVELICONS() {
        RETURN LEVELICONS;
    }
    PUBLIC STATIC VOID SETLEVELICONS(IMAGEICON[] LEVELICONS) {
        DATABASE.LEVELICONS = LEVELICONS;
    }
    // РИСУНКИ ТАЙЛОВ
    PRIVATE STATIC IMAGEICON[][] TILESICONS;
    PUBLIC STATIC IMAGEICON[][] GETTILES() {
        RETURN TILESICONS;
    }
    PUBLIC STATIC IMAGEICON[] GETLEVELTILES(INT LEVELNUMBER) {
        RETURN TILESICONS[LEVELNUMBER];
    }
```

```
    // ПОРЯДОК ТАЙЛОВ
    PRIVATE STATIC TILE[][] TILESORDERONTHESCREEN;
    PUBLIC STATIC TILE[] GETTILESORDERONTHESCREEN(INT I) {
        RETURN TILESORDERONTHESCREEN[I];
    }
    PUBLIC STATIC VOID SETTILESORDERONTHESCREEN(INT[]
TILESORDERONTHESCREEN, INT INDEX) {
        FOR (INT I = 0; I < 16; I++) {
            DATABASE.TILESORDERONTHESCREEN[INDEX][I] = NEW
TILE(TILESORDERONTHESCREEN[I],INDEX);

DATABASE.TILESORDERONTHESCREEN[INDEX][I].SETICON(DATABASE.TILESICO
NS[INDEX][I]);
        }
    }
     //ПОРЯДОК СЕКРЕТНЫХ ТАЙЛОВ
    PRIVATE STATIC TILE[] SECRETTILESORDERONTHESCREEN;
    PUBLIC STATIC TILE[] GETSECRETTILESORDERONTHESCREEN() {
        RETURN SECRETTILESORDERONTHESCREEN;
    }
    PUBLIC STATIC VOID SETSECRETTILESORDERONTHESCREEN(INT[]
TILESORDERONTHESCREEN) {
        FOR (INT I = 0; I < 16; I++) {
            DATABASE.SECRETTILESORDERONTHESCREEN[I] = NEW
TILE(TILESORDERONTHESCREEN[I], 0);

DATABASE.SECRETTILESORDERONTHESCREEN[I].SETICON(DATABASE.SECRETTIL
ESICONS[I]);
        }
    }
    // УРОВНИ
    PRIVATE STATIC LEVELWINDOW[] LEVELWINDOWS;
    PUBLIC STATIC LEVELWINDOW GETLEVELWINDOWS(INT I) {
        RETURN LEVELWINDOWS[I];
    }
```

```
    // СЕКРЕТНЫЙ УРОВЕНЬ
    PRIVATE STATIC SECRETLEVEL SECRETLEVEL;
    PUBLIC STATIC SECRETLEVEL GETSECRETLEVEL() {
        RETURN SECRETLEVEL;
    }
    // РИСУНКИ ТАЙЛОВ СЕКРЕТНОГО УРОВНЯ
    PRIVATE STATIC IMAGEICON[] SECRETTILESICONS;
    PUBLIC STATIC IMAGEICON[] GETSECRETTILES() {
        RETURN SECRETTILESICONS;
    }

    // РИСУНОК СЕКРЕТНОГО УРОВНЯ
    PRIVATE STATIC  IMAGEICON SECRETLEVELIMAGE;
    PUBLIC STATIC IMAGEICON GETSECRETLEVELIMAGE() {
        RETURN SECRETLEVELIMAGE;
    }
    // КОНСТРУКТОР
    PRIVATE DATABASE() {
        NUMBEROFLEVELS = 3;
        //
        // WAITING FOR FUNCTIONAL PART TO GET TILES
        //
        ISSOLVED = NEW BOOLEAN[NUMBEROFLEVELS];
        LEVELICONS = NEW IMAGEICON[NUMBEROFLEVELS];
        TILESICONS = NEW IMAGEICON[NUMBEROFLEVELS][16];
        LEVELWINDOWS = NEW LEVELWINDOW[NUMBEROFLEVELS];
        TILESORDERONTHESCREEN = NEW TILE[NUMBEROFLEVELS][16];
        SECRETTILESORDERONTHESCREEN = NEW TILE[16];
        SECRETTILESICONS = NEW IMAGEICON[16];
        FOR (INT I = 0; I < NUMBEROFLEVELS+1; I++){
            IF (I == NUMBEROFLEVELS) {

                STRING PATH = NEW
STRING("/SECRETLEVEL/IMAGE.JPEG");
                URL URL = THIS.GETCLASS().GETRESOURCE(PATH);
```

```
                SECRETLEVELIMAGE = NEW IMAGEICON(URL);
                FOR (INT J = 0; J < 16; J++) {
                    STRING PATH2 = NEW STRING(
"/SECRETLEVEL/IMAGES/IMAGE_" + (J+1) +".JPG");
                    URL = THIS.GETCLASS().GETRESOURCE(PATH2);
                    SECRETTILESICONS[J] = NEW IMAGEICON(URL);
                }

                SECRETLEVEL = NEW SECRETLEVEL();
            //    RANDOMIZESECRETPUZZLE();
                BREAK;
            }
            STRING PATH = NEW STRING("/LEVEL"+ (I+1)+
"/IMAGE.JPEG");
            URL URL = THIS.GETCLASS().GETRESOURCE(PATH);
            LEVELICONS[I] = NEW IMAGEICON(URL);
            FOR (INT J = 0; J < 16; J++) {
                STRING PATH2 = NEW STRING( "/LEVEL" + (I+1) +
"/IMAGES/IMAGE_" + (J+1) +".JPG");
                URL = THIS.GETCLASS().GETRESOURCE(PATH2);
                TILESICONS[I][J] = NEW IMAGEICON(URL);
            }
        }

        FOR (INT I = 0; I < NUMBEROFLEVELS; I++) {
            LEVELWINDOWS[I] = NEW LEVELWINDOW(I);
            RANDOMIZEPUZZLE(I);
        }
    }
}
```

GAMESTATE.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.GAMESTATE;
```

```
IMPORT COM.PRACTICE.BLUETEAM.DATABASE.DATABASE;

PUBLIC CLASS GAMESTATE {
    PRIVATE STATIC GAMESTATE STATE = NEW GAMESTATE();
    // КОД СЕКРЕТНОГО УРОВНЯ
    PRIVATE STATIC STRING SECRETCODE = "LEAVEMEALONE";
    PRIVATE STATIC INT CORRECTLENGTH = 0;
    PRIVATE STATIC INT CODELENGTH = SECRETCODE.LENGTH();
    // ВВОД КОДА
    PUBLIC STATIC VOID SECRETTYPED(CHAR C) {
        IF (SECRETCODE.CHARAT(CORRECTLENGTH) == C) {
            CORRECTLENGTH++;
        } ELSE {
            CORRECTLENGTH = 0;
        }
    }
    // ПРОВЕРКА КОДА
    PUBLIC STATIC BOOLEAN ISCODEACTIVATED() {
        IF (CORRECTLENGTH == CODELENGTH) {
            CORRECTLENGTH = 0;
            RETURN TRUE;
        } ELSE {
            RETURN FALSE;
        }
    }

    PUBLIC STATIC BOOLEAN GETISAVAILABLE(INT I) {
        RETURN ISAVAILABLE[I];
    }

    PUBLIC STATIC VOID SETISAVAILABLE(INT INDEX) {
        IF (INDEX == ISAVAILABLE.LENGTH)
            RETURN;
        GAMESTATE.ISAVAILABLE[INDEX] = TRUE;
    }
```

```
    // МАССИВ ПРОЙДЕННЫЙ УРОВНЕЙ;
    PRIVATE STATIC BOOLEAN ISAVAILABLE[];
    PRIVATE GAMESTATE() {
        ISAVAILABLE = NEW BOOLEAN[DATABASE.GETNUMBEROFLEVELS()];
        ISAVAILABLE[0] = TRUE;
        FOR (INT I = 1; I < DATABASE.GETNUMBEROFLEVELS(); I++) {
            ISAVAILABLE[I] = FALSE;
        }
    }
}
```

LEVELSELECTWINDOW.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.UI;


IMPORT COM.PRACTICE.BLUETEAM.DATABASE.DATABASE;
IMPORT COM.PRACTICE.BLUETEAM.GAMESTATE.GAMESTATE;

IMPORT JAVAX.SWING.*;
IMPORT JAVAX.XML.CRYPTO.DATA;
IMPORT JAVA.AWT.*;
IMPORT JAVA.AWT.EVENT.KEYADAPTER;
IMPORT JAVA.AWT.EVENT.KEYEVENT;
IMPORT JAVA.AWT.EVENT.MOUSEADAPTER;
IMPORT JAVA.AWT.EVENT.MOUSEEVENT;

PUBLIC CLASS LEVELSELECTWINDOW EXTENDS JFRAME {
    PRIVATE STATIC LEVELSELECTWINDOW SELECTWINDOW = NEW
LEVELSELECTWINDOW();
    // МАССИВ КНОПОК ВЫБОРА УРОВНЯ

    PRIVATE STATIC LEVELBUTTON[] LEVELBUTTONS;
    // КЛАСС КНОПКИ ВЫБОРА УРОВНЯ
```

```
    PRIVATE CLASS LEVELBUTTON EXTENDS JBUTTON{
        PUBLIC INT GETLEVELNUMBER() {
            RETURN LEVELNUMBER;
        }

        PUBLIC VOID SETLEVELNUMBER(INT LEVELNUMBER) {
            THIS.LEVELNUMBER = LEVELNUMBER;
        }

        PRIVATE INT LEVELNUMBER;
        PUBLIC LEVELBUTTON(INT LEVELNUMBER, STRING NAME) {
            SUPER(NAME);
            THIS.LEVELNUMBER = LEVELNUMBER;
            THIS.ADDMOUSELISTENER(MOUSEADAPTER);
        }
        PUBLIC  LEVELBUTTON(){
            SUPER();
            THIS.LEVELNUMBER = 0;
            THIS.ADDMOUSELISTENER(MOUSEADAPTER);
        }
        PUBLIC  LEVELBUTTON(INT LEVELNUMBER){
            SUPER();
            THIS.LEVELNUMBER = LEVELNUMBER;

        }
        MOUSEADAPTER MOUSEADAPTER = NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                SELECTWINDOW.SETVISIBLE(FALSE);

DATABASE.GETLEVELWINDOWS(LEVELNUMBER).SETVISIBLE(TRUE);
                DATABASE.RANDOMIZEPUZZLE(LEVELNUMBER);
                FOCUSRELEASED(E.GETCOMPONENT());
            }
        };
```

```
    }
    // МЕТОД ОТВЕЧАЮЩИЙ ЗА СОЗДАНИЕ МАССИВА КНОПОК
    PRIVATE VOID CREATELEVELBUTTONS(){
        FOR (INT I = 0; I < DATABASE.GETNUMBEROFLEVELS(); ++I) {
            LEVELBUTTONS[I] = NEW LEVELBUTTON(I);
            LEVELBUTTONS[I].SETICON(DATABASE.GETLEVELICONS()[I]);
            LEVELBUTTONS[I].SETBORDERPAINTED(FALSE);
            LEVELBUTTONS[I].SETFOCUSPAINTED(FALSE);
        }
    }

    PUBLIC STATIC VOID UPDATE() {
        FOR (INT I = 0; I < LEVELBUTTONS.LENGTH; I++) {
            IF (!GAMESTATE.GETISAVAILABLE(I)) {
                LEVELBUTTONS[I].SETENABLED(FALSE);
            } ELSE {
                LEVELBUTTONS[I].SETENABLED(TRUE);

LEVELBUTTONS[I].ADDMOUSELISTENER(LEVELBUTTONS[I].MOUSEADAPTER);
            }
        }
    }

    PUBLIC VOID FOCUSRELEASED(COMPONENT COMPONENT)
    {
        COMPONENT.SETFOCUSABLE(FALSE);
        THIS.SETFOCUSABLE(TRUE);
        THIS.REQUESTFOCUS();
    }
    // LISTNER ДЛЯ СЕКРЕТНОГО УРОВНЯ
    // УРОВНИ
    // КНОПКА ЗАКРЫТИЯ ПРИЛОЖЕНИЯ
    PRIVATE JBUTTON CLOSEBUTTON = NEW JBUTTON("CLOSE");
    // КНОПКА ВОЗВРАТА В МЕНЮ
```

```
    PRIVATE JBUTTON RETURNBUTTON = NEW JBUTTON("RETURN TO MAIN
MENU");

    PUBLIC STATIC LEVELSELECTWINDOW GETSELECTWINDOW() {
        RETURN SELECTWINDOW;
    }

    PRIVATE LEVELSELECTWINDOW() {
        //НАСТРОЙКИ ОСНОВНОГО ОКНА
        SUPER("ПЯТНАШКИ");
        LEVELBUTTONS = NEW
LEVELBUTTON[DATABASE.GETNUMBEROFLEVELS()];
        THIS.SETRESIZABLE(FALSE);
        THIS.SETDEFAULTCLOSEOPERATION(EXIT_ON_CLOSE);
        // НАСТРОЙКИ РАЗМЕРА ОКНА
        DIMENSION SCREENSIZE =
TOOLKIT.GETDEFAULTTOOLKIT().GETSCREENSIZE();
        INT SIZEWIDTH = 800;
        INT SIZEHEIGHT = 700;
        INT LOCATIONX = (SCREENSIZE.WIDTH - SIZEWIDTH) / 2;
        INT LOCATIONY = (SCREENSIZE.HEIGHT - SIZEHEIGHT) / 2;
        THIS.SETBOUNDS(LOCATIONX, LOCATIONY, SIZEWIDTH,
SIZEHEIGHT);
        // ИНИЦИАЛИЗАЦИЯ MOUSELISTNER ДЛЯ CLOSEBUTTON
        CLOSEBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                SYSTEM.EXIT(0);
                FOCUSRELEASED(CLOSEBUTTON);
            }
        });
        // ИНИЦИАЛИЗАЦИЯ MOUSELISTNER ДЛЯ RETURNBUTTON
        RETURNBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
```

```
                SELECTWINDOW.SETVISIBLE(FALSE);
                MAINWINDOW.GETUI().SHOWUI();
                FOCUSRELEASED(RETURNBUTTON);
            }
        });
        // ИНИЦИАЛИЗАЦИЯ МАССИВА LEVELBUTTONS
        CREATELEVELBUTTONS();

        //// НАСТРОЙКИ ВНЕШНЕГО ВИДА ОКНА
        //ОСНОВНАЯ ПАНЕЛЬ, НА НЕЙ РАЗМЕЩЕНЫ ДРУГИЕ ПАНЕЛИ
        JPANEL MAINPANEL = NEW JPANEL();
        MAINPANEL.SETLAYOUT(NULL);
        MAINPANEL.SETSIZE(THIS.GETSIZE());

        // ПАНЕЛЬ С КНОПКАМИ
        JPANEL LEVELPANEL = NEW JPANEL();

LEVELPANEL.SETBOUNDS(MAINPANEL.GETX(),MAINPANEL.GETY(),MAINPANEL.G
ETWIDTH() - 10,MAINPANEL.GETHEIGHT() - 100);
        GRIDBAGLAYOUT LAYOUT = NEW GRIDBAGLAYOUT();
        LEVELPANEL.SETLAYOUT(LAYOUT);

        // ПЕРВОНАЧАЛЬНЫЕ НАСТРОЙКИ СЕТКИ
        GRIDBAGCONSTRAINTS GBC = NEW GRIDBAGCONSTRAINTS();
        GBC.FILL = GRIDBAGCONSTRAINTS.HORIZONTAL;
        GBC.IPADX = 0;
        GBC.IPADY = 0;
        GBC.GRIDX = 0;
        GBC.GRIDY = 0;
        GBC.INSETS = NEW INSETS(10,10,10,10);
        //ДОБАВЛЕНИЕ КНОПОК НА ПАНЕЛЬ
        FOR (INT I = 0; I < LEVELBUTTONS.LENGTH; I++) {
             {
                IF (I % 2 == 1) {
                    LEVELPANEL.ADD(LEVELBUTTONS[I], GBC);
```

```
                    GBC.GRIDX = 0;
                    GBC.GRIDY++;
                } ELSE {
                    LEVELPANEL.ADD(LEVELBUTTONS[I], GBC);
                    GBC.GRIDX++;
                }
                IF (!GAMESTATE.GETISAVAILABLE(I)) {
                    LEVELBUTTONS[I].SETENABLED(FALSE);
                } ELSE {
                    LEVELBUTTONS[I].SETENABLED(TRUE);

LEVELBUTTONS[I].ADDMOUSELISTENER(LEVELBUTTONS[I].MOUSEADAPTER);
                }
            }
        }
        // ОБЕРТКА В JSCROLLPANE
        JSCROLLPANE LEVELSCROLL = NEW JSCROLLPANE(LEVELPANEL);
        LEVELSCROLL.SETBOUNDS(LEVELPANEL.GETBOUNDS());
        LEVELSCROLL.CREATEVERTICALSCROLLBAR();
        LEVELSCROLL.GETVERTICALSCROLLBAR().SETUNITINCREMENT(10);

LEVELSCROLL.SETVERTICALSCROLLBARPOLICY(JSCROLLPANE.VERTICAL_SCROLL
BAR_AS_NEEDED);

        // ДОБАВЛЕНИЕ КНОПОК УРОВНЯ НА MAINPANEL
        MAINPANEL.ADD(LEVELSCROLL,BORDERLAYOUT.CENTER);

        // ПАНЕЛЬ ДЛЯ КНОПОК НАВИГАЦИИ МЕНЮ
        JPANEL MENUPANEL = NEW JPANEL();
        MENUPANEL.SETLAYOUT(NEW GRIDBAGLAYOUT());
        MENUPANEL.SETBOUNDS(LEVELPANEL.GETX(),
LEVELPANEL.GETHEIGHT() -10,
                LEVELPANEL.GETWIDTH(),MAINPANEL.GETHEIGHT() -
LEVELPANEL.GETHEIGHT());
        // НАСТРОЙКА РАСПОЛОЖЕНИЯ КНОПОК
```

```
        GBC.GRIDY = 0;
        GBC.GRIDX = 0;
        GBC.IPADX = 0;
        GBC.IPADY = 15;
        GBC.INSETS = NEW INSETS(0,10,20,10);
        GBC.ANCHOR = GRIDBAGCONSTRAINTS.CENTER;
        MENUPANEL.ADD(CLOSEBUTTON,GBC);
        GBC.GRIDX = 1;
        GBC.WEIGHTX = 2;
        MENUPANEL.ADD(RETURNBUTTON, GBC);
        MAINPANEL.ADD(MENUPANEL);
        THIS.GETCONTENTPANE().ADD(MAINPANEL);
        FOR (INT I = 0; I < LEVELBUTTONS.LENGTH; I++) {
            LEVELBUTTONS[I].SETSIZE(330,330);
            LEVELBUTTONS[I].SETICON(NEW
IMAGEICON(DATABASE.GETLEVELICONS()[I].GETIMAGE().GETSCALEDINSTANCE
(LEVELBUTTONS[I].GETWIDTH(),LEVELBUTTONS[I].GETHEIGHT(),IMAGE.SCAL
E_SMOOTH)));
            LEVELBUTTONS[I].SETCONTENTAREAFILLED(FALSE);
        }
        FOCUSRELEASED(MAINPANEL);
        THIS.ADDKEYLISTENER(NEW KEYADAPTER() {
            @OVERRIDE
            PUBLIC VOID KEYTYPED(KEYEVENT E) {
                SUPER.KEYTYPED(E);
                GAMESTATE.SECRETTYPED(E.GETKEYCHAR());
                IF (GAMESTATE.ISCODEACTIVATED()) {
                    SELECTWINDOW.SETVISIBLE(FALSE);
                    DATABASE.GETSECRETLEVEL().SETVISIBLE(TRUE);
                    DATABASE.RANDOMIZESECRETPUZZLE();

                }
            }
        });
```

```
    }
}
```

LEVELWINDOW.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.UI;

IMPORT COM.PRACTICE.BLUETEAM.ALGO.MAIN;
IMPORT COM.PRACTICE.BLUETEAM.ALGO.PUZZLE;
IMPORT COM.PRACTICE.BLUETEAM.ALGO.WRAP;
IMPORT COM.PRACTICE.BLUETEAM.DATABASE.DATABASE;

IMPORT JAVAX.SWING.*;
IMPORT JAVA.AWT.*;
IMPORT JAVA.AWT.EVENT.ACTIONEVENT;
IMPORT JAVA.AWT.EVENT.ACTIONLISTENER;
IMPORT JAVA.AWT.EVENT.MOUSEADAPTER;
IMPORT JAVA.AWT.EVENT.MOUSEEVENT;
IMPORT JAVA.BEANS.PROPERTYCHANGEEVENT;
IMPORT JAVA.BEANS.PROPERTYCHANGELISTENER;
IMPORT JAVA.UTIL.RANDOM;

PUBLIC CLASS LEVELWINDOW EXTENDS JFRAME
        IMPLEMENTS ACTIONLISTENER, PROPERTYCHANGELISTENER {
    PRIVATE JBUTTON CLOSEBUTTON = NEW JBUTTON("CLOSE");
    PRIVATE JBUTTON RANDOMBUTTON = NEW JBUTTON("RANDOM");
    PRIVATE JBUTTON BUILDBUTTON = NEW JBUTTON("SHOW SOLUTION");

    PUBLIC JBUTTON GETNEXTBUTTON() {
        RETURN NEXTBUTTON;
    }

    PRIVATE JBUTTON NEXTBUTTON = NEW JBUTTON("NEXT");
    PRIVATE JBUTTON PREVBUTTON = NEW JBUTTON("PREV");
```

```
    PRIVATE JBUTTON RETURNBUTTON = NEW JBUTTON("RETURN TO MAIN
MENU");
    PRIVATE JPANEL TILESPANEL = NEW JPANEL();
    PRIVATE INT LEVELNUMBER;
    PRIVATE TILE[] TILES;

    PRIVATE TASK TASK;
    PRIVATE JPROGRESSBAR PROGRESSBAR;
    CLASS TASK EXTENDS SWINGWORKER<VOID, VOID> {
        /*
         * MAIN TASK. EXECUTED IN BACKGROUND THREAD.
         */
        @OVERRIDE
        PUBLIC VOID DOINBACKGROUND() {
            PROGRESSBAR.SETVISIBLE(TRUE);
            RANDOM RANDOM = NEW RANDOM();
            NEXTBUTTON.SETENABLED(FALSE);
            INT PROGRESS = 0;
            //INITIALIZE PROGRESS PROPERTY.
            SETPROGRESS(0);

            WHILE (PROGRESS < 100) {
                //SLEEP FOR UP TO ONE SECOND.
                TRY {
                    THREAD.SLEEP(RANDOM.NEXTINT(100));
                } CATCH (INTERRUPTEDEXCEPTION IGNORE) {}
                //MAKE RANDOM PROGRESS.
                PROGRESS += RANDOM.NEXTINT(50);
                SETPROGRESS(MATH.MIN(PROGRESS, 100));
            }

            RETURN NULL;
        }

        /*
```

```
         * EXECUTED IN EVENT DISPATCHING THREAD
         */
        @OVERRIDE
        PUBLIC VOID DONE() {
            DATABASE.NEXTSTEP(LEVELNUMBER);
            NEXTBUTTON.SETENABLED(TRUE);
            PROGRESSBAR.SETVISIBLE(FALSE);
            IF (DATABASE.GETISSOLVED(LEVELNUMBER))
                NEXTBUTTON.SETENABLED(FALSE);
            ELSE
                NEXTBUTTON.SETENABLED(TRUE);
            SETTILES();
            SETCURSOR(NULL); //TURN OFF THE WAIT CURSOR
        }
    }

    PUBLIC VOID SETTILES() {
        FOR (INT I = 0; I < 16; I++)
        {
            TILES[I] = NEW
TILE(DATABASE.GETTILESORDERONTHESCREEN(LEVELNUMBER)[I].NUMBEROFTIL
E, LEVELNUMBER);

TILES[I].SETICON(DATABASE.GETLEVELTILES(LEVELNUMBER)[DATABASE.GETT
ILESORDERONTHESCREEN(LEVELNUMBER)[I].NUMBEROFTILE]);
            TILES[I].SETBORDERPAINTED(FALSE);
            TILES[I].SETFOCUSPAINTED(FALSE);

        }
        TILESPANEL.REMOVEALL();
        FOR (INT I = 0; I < 16; I++) {
            TILESPANEL.ADD(TILES[I]);
            TILES[I].SETICON(NEW
IMAGEICON(DATABASE.GETLEVELTILES(LEVELNUMBER)[TILES[I].NUMBEROFTIL
```

```
E].GETIMAGE().GETSCALEDINSTANCE(TILESPANEL.GETWIDTH()/4,
TILESPANEL.GETHEIGHT()/4, IMAGE.SCALE_SMOOTH)));
        }
    }
    PUBLIC LEVELWINDOW(INT LEVELNUMBER) {

        SUPER("ПЯТНАШКИ");
        THIS.LEVELNUMBER = LEVELNUMBER;
        SETRESIZABLE(FALSE);
        SETDEFAULTCLOSEOPERATION(EXIT_ON_CLOSE);
        DIMENSION SCREENSIZE =
TOOLKIT.GETDEFAULTTOOLKIT().GETSCREENSIZE();
        TILES = NEW TILE[16];

        PROGRESSBAR = NEW JPROGRESSBAR(0,100);
        PROGRESSBAR.SETVALUE(0);
        PROGRESSBAR.SETSTRINGPAINTED(TRUE);
        PROGRESSBAR.SETVISIBLE(FALSE);



        INT SIZEWIDTH = 800;
        INT SIZEHEIGHT = 700;
        INT LOCATIONX = (SCREENSIZE.WIDTH - SIZEWIDTH) / 2;
        INT LOCATIONY = (SCREENSIZE.HEIGHT - SIZEHEIGHT) / 2;
        THIS.SETBOUNDS(LOCATIONX, LOCATIONY, SIZEWIDTH,
SIZEHEIGHT);
        JPANEL MAINPANEL = NEW JPANEL();

        TILESPANEL = NEW JPANEL();
        THIS.SETCONTENTPANE(MAINPANEL);
        MAINPANEL.SETLAYOUT(NULL);
        MAINPANEL.SETSIZE(THIS.GETSIZE());
        PROGRESSBAR.SETBOUNDS(MAINPANEL.GETWIDTH()/5,
MAINPANEL.GETHEIGHT()/10, 200, 30);
```

```
        MAINPANEL.ADD(PROGRESSBAR);
        TILESPANEL.SETLAYOUT(NEW GRIDLAYOUT(4,4,0,0));

TILESPANEL.SETLOCATION(MAINPANEL.GETWIDTH()/16,MAINPANEL.GETHEIGHT
()/6 );
        TILESPANEL.SETSIZE(400, 400);
        MAINPANEL.ADD(TILESPANEL);
        JPANEL BUTTONSPANEL = NEW JPANEL();
        BUTTONSPANEL.SETLOCATION(TILESPANEL.GETWIDTH() + 100,
TILESPANEL.GETY() -10);
        BUTTONSPANEL.SETSIZE(200,300);
        BUTTONSPANEL.SETLAYOUT(NEW GRIDLAYOUT(5,1,5,5));
        BUTTONSPANEL.ADD(RANDOMBUTTON);
        BUTTONSPANEL.ADD(BUILDBUTTON);
        JPANEL NEXTPREVPANEL = NEW JPANEL();
        NEXTPREVPANEL.SETLAYOUT(NEW GRIDLAYOUT(1,2,0,0));
        NEXTPREVPANEL.ADD(PREVBUTTON);
        NEXTPREVPANEL.ADD(NEXTBUTTON);
        BUTTONSPANEL.ADD(NEXTPREVPANEL);
        BUTTONSPANEL.ADD(CLOSEBUTTON);
        BUTTONSPANEL.ADD(RETURNBUTTON);
        MAINPANEL.ADD(BUTTONSPANEL);
        BUILDBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                DATABASE.AUTOSOLVE(LEVELNUMBER);
                IF (DATABASE.GETISSOLVED(LEVELNUMBER))
                    NEXTBUTTON.SETENABLED(FALSE);
                ELSE
                    NEXTBUTTON.SETENABLED(TRUE);
            }
        });
        NEXTBUTTON.SETACTIONCOMMAND("START");
        NEXTBUTTON.ADDACTIONLISTENER(THIS);
        PREVBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
```

```
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                DATABASE.PREVSTEP(LEVELNUMBER);
                IF (DATABASE.GETISSOLVED(LEVELNUMBER))
                    NEXTBUTTON.SETENABLED(FALSE);
                ELSE
                    NEXTBUTTON.SETENABLED(TRUE);
            }
        });
        CLOSEBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSEPRESSED(MOUSEEVENT E) {
                SYSTEM.EXIT(0);
            }
        });
        RETURNBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {

DATABASE.GETLEVELWINDOWS(LEVELNUMBER).SETVISIBLE(FALSE);
                MAINWINDOW.GETUI().SETVISIBLE(TRUE);
            }
        });
        RANDOMBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                DATABASE.RANDOMIZEPUZZLE(LEVELNUMBER);
                IF (DATABASE.GETISSOLVED(LEVELNUMBER))
                    NEXTBUTTON.SETENABLED(FALSE);
                ELSE
                    NEXTBUTTON.SETENABLED(TRUE);
            }
        });

    }
```

```
    @OVERRIDE
    PUBLIC VOID ACTIONPERFORMED(ACTIONEVENT E) {
        SETCURSOR(CURSOR.GETPREDEFINEDCURSOR(CURSOR.WAIT_CURSOR));
        //INSTANCES OF JAVAX.SWING.SWINGWORKER ARE NOT REUSUABLE,
SO
        //WE CREATE NEW INSTANCES AS NEEDED.
        TASK = NEW TASK();

        TASK.ADDPROPERTYCHANGELISTENER(THIS);
        TASK.EXECUTE();

    }

    @OVERRIDE
    PUBLIC VOID PROPERTYCHANGE(PROPERTYCHANGEEVENT EVT) {
        IF ("PROGRESS" == EVT.GETPROPERTYNAME()) {
            INT PROGRESS = (INTEGER) EVT.GETNEWVALUE();
            PROGRESSBAR.SETVALUE(PROGRESS);
        }
    }
}
```

MAINWINDOW.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.UI;

IMPORT COM.PRACTICE.BLUETEAM.DATABASE.DATABASE;
IMPORT COM.PRACTICE.BLUETEAM.GAMESTATE.GAMESTATE;

IMPORT JAVAX.SWING.*;
IMPORT JAVA.AWT.*;
IMPORT JAVA.AWT.EVENT.KEYADAPTER;
IMPORT JAVA.AWT.EVENT.KEYEVENT;
IMPORT JAVA.AWT.EVENT.MOUSEADAPTER;
```

```
IMPORT JAVA.AWT.EVENT.MOUSEEVENT;


PUBLIC CLASS MAINWINDOW EXTENDS  JFRAME {
    PUBLIC STATIC LEVELSELECTWINDOW GETSELECTWINDOW() {
        RETURN SELECTWINDOW;
    }
    PRIVATE STATIC LEVELSELECTWINDOW SELECTWINDOW;
    PRIVATE STATIC MAINWINDOW UI = NULL;
    PRIVATE JBUTTON NEWGAME = NEW JBUTTON("NEW GAME");
    PRIVATE JBUTTON CLOSE = NEW JBUTTON("CLOSE");

    PUBLIC STATIC MAINWINDOW GETUI() {
        IF (UI == NULL)
            UI = NEW MAINWINDOW();
        RETURN UI;
    }

    PUBLIC VOID SHOWUI() {
        UI.SETVISIBLE(TRUE);
    }
    PUBLIC VOID CLOSEUI() { UI.SETVISIBLE(FALSE); }
    PRIVATE MAINWINDOW() {
        SUPER("ПЯТНАШКИ");
        SELECTWINDOW = LEVELSELECTWINDOW.GETSELECTWINDOW();
        THIS.SETRESIZABLE(FALSE);
        THIS.SETDEFAULTCLOSEOPERATION(EXIT_ON_CLOSE);
        DIMENSION SCREENSIZE =
TOOLKIT.GETDEFAULTTOOLKIT().GETSCREENSIZE();
        INT SIZEWIDTH = 800;
        INT SIZEHEIGHT = 600;
        INT LOCATIONX = (SCREENSIZE.WIDTH - SIZEWIDTH) / 2;
        INT LOCATIONY = (SCREENSIZE.HEIGHT - SIZEHEIGHT) / 2;
        THIS.SETBOUNDS(LOCATIONX, LOCATIONY, SIZEWIDTH,
SIZEHEIGHT);
```

```
        NEWGAME.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                UI.SETVISIBLE(FALSE);
                SELECTWINDOW.SETVISIBLE(TRUE);
            }
        });
        CLOSE.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSEPRESSED(MOUSEEVENT E) {
                SYSTEM.EXIT(0);
            }
        });
        JPANEL MAINPANEL = NEW JPANEL();
        JPANEL BUTTONSPANEL = NEW JPANEL();
        THIS.SETCONTENTPANE(MAINPANEL);
        MAINPANEL.SETLAYOUT(NULL);
        MAINPANEL.SETSIZE(THIS.GETSIZE());
        BUTTONSPANEL.SETLAYOUT(NULL);
        BUTTONSPANEL.SETLOCATION(MAINPANEL.GETX() +
10,MAINPANEL.GETY() + 20);
        BUTTONSPANEL.SETSIZE(MAINPANEL.GETWIDTH() - 10,
MAINPANEL.GETHEIGHT() - 50);
        BUTTONSPANEL.ADD(NEWGAME);
        BUTTONSPANEL.ADD(CLOSE);
        NEWGAME.SETSIZE(BUTTONSPANEL.GETWIDTH() - 50,
BUTTONSPANEL.GETHEIGHT()/2 - 40);
        CLOSE.SETSIZE(BUTTONSPANEL.GETWIDTH() - 50,
BUTTONSPANEL.GETHEIGHT()/2 - 40);
        NEWGAME.SETLOCATION(BUTTONSPANEL.GETX(),0);

CLOSE.SETLOCATION(BUTTONSPANEL.GETX(),BUTTONSPANEL.GETHEIGHT()/2);
        MAINPANEL.ADD(BUTTONSPANEL);
    }
}
```

SECRETLEVEL.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.UI;

IMPORT COM.PRACTICE.BLUETEAM.DATABASE.DATABASE;

IMPORT JAVAX.SWING.*;
IMPORT JAVAX.XML.CRYPTO.DATA;
IMPORT JAVA.AWT.*;
IMPORT JAVA.AWT.EVENT.ACTIONEVENT;
IMPORT JAVA.AWT.EVENT.ACTIONLISTENER;
IMPORT JAVA.AWT.EVENT.MOUSEADAPTER;
IMPORT JAVA.AWT.EVENT.MOUSEEVENT;
IMPORT JAVA.BEANS.PROPERTYCHANGEEVENT;
IMPORT JAVA.BEANS.PROPERTYCHANGELISTENER;
IMPORT JAVA.UTIL.RANDOM;

PUBLIC CLASS SECRETLEVEL EXTENDS JFRAME IMPLEMENTS ACTIONLISTENER,
PROPERTYCHANGELISTENER {
    PRIVATE JBUTTON CLOSEBUTTON = NEW JBUTTON("CLOSE");
    PRIVATE JBUTTON RANDOMBUTTON = NEW JBUTTON("RANDOM");
    PRIVATE JBUTTON BUILDBUTTON = NEW JBUTTON("SHOW SOLUTION");

    PUBLIC JBUTTON GETNEXTBUTTON() {
        RETURN NEXTBUTTON;
    }

    PRIVATE JBUTTON NEXTBUTTON = NEW JBUTTON("NEXT");
    PRIVATE JBUTTON PREVBUTTON = NEW JBUTTON("PREV");
    PRIVATE JBUTTON RETURNBUTTON = NEW JBUTTON("RETURN TO MAIN
MENU");
    PRIVATE JPANEL TILESPANEL = NEW JPANEL();

    PRIVATE INT LEVELNUMBER;
    PRIVATE TILE[] TILES;
```

```
PRIVATE TASK TASK;
PRIVATE JPROGRESSBAR PROGRESSBAR;
CLASS TASK EXTENDS SWINGWORKER<VOID, VOID> {
    /*
     * MAIN TASK. EXECUTED IN BACKGROUND THREAD.
     */
    @OVERRIDE
    PUBLIC VOID DOINBACKGROUND() {
        PROGRESSBAR.SETVISIBLE(TRUE);
        RANDOM RANDOM = NEW RANDOM();
        NEXTBUTTON.SETENABLED(FALSE);
        INT PROGRESS = 0;
        //INITIALIZE PROGRESS PROPERTY.
        SETPROGRESS(0);

        WHILE (PROGRESS < 100) {
            //SLEEP FOR UP TO ONE SECOND.
            TRY {
                THREAD.SLEEP(RANDOM.NEXTINT(100));
            } CATCH (INTERRUPTEDEXCEPTION IGNORE) {}
            //MAKE RANDOM PROGRESS.
            PROGRESS += RANDOM.NEXTINT(50);
            SETPROGRESS(MATH.MIN(PROGRESS, 100));
        }
        RETURN NULL;
    }

    /*
     * EXECUTED IN EVENT DISPATCHING THREAD
     */
    @OVERRIDE
    PUBLIC VOID DONE() {
        DATABASE.SECRETNEXTSTEP();
        NEXTBUTTON.SETENABLED(TRUE);
```

```
            PROGRESSBAR.SETVISIBLE(FALSE);
            IF (DATABASE.GETISSOLVED())
                NEXTBUTTON.SETENABLED(FALSE);
            ELSE
                NEXTBUTTON.SETENABLED(TRUE);
            SETTILES();
            SETCURSOR(NULL); //TURN OFF THE WAIT CURSOR
        }
    }
    PUBLIC VOID SETTILES() {
        FOR (INT I = 0; I < 16; I++)
        {
            TILES[I] = NEW
TILE(DATABASE.GETSECRETTILESORDERONTHESCREEN()[I].NUMBEROFTILE,
LEVELNUMBER);

TILES[I].SETICON(DATABASE.GETSECRETTILES()[DATABASE.GETSECRETTILES
ORDERONTHESCREEN()[I].NUMBEROFTILE]);
            TILES[I].SETBORDERPAINTED(FALSE);
            TILES[I].SETFOCUSPAINTED(FALSE);
        }
        TILESPANEL.REMOVEALL();
        FOR (INT I = 0; I < 16; I++) {
            TILESPANEL.ADD(TILES[I]);
            TILES[I].SETICON(NEW
IMAGEICON(DATABASE.GETSECRETTILES()[TILES[I].NUMBEROFTILE].GETIMAG
E().GETSCALEDINSTANCE(TILESPANEL.GETWIDTH()/4,
TILESPANEL.GETHEIGHT()/4, IMAGE.SCALE_SMOOTH)));
        }
    }

    PUBLIC SECRETLEVEL() {
        SUPER("ПЯТНАШКИ");

        SETRESIZABLE(FALSE);
```

```
        SETDEFAULTCLOSEOPERATION(EXIT_ON_CLOSE);
        DIMENSION SCREENSIZE =
TOOLKIT.GETDEFAULTTOOLKIT().GETSCREENSIZE();
        TILESPANEL = NEW JPANEL();
        LEVELNUMBER = 4;
        TILES = NEW TILE[16];

        PROGRESSBAR = NEW JPROGRESSBAR(0,100);
        PROGRESSBAR.SETVALUE(0);
        PROGRESSBAR.SETSTRINGPAINTED(TRUE);
        PROGRESSBAR.SETVISIBLE(FALSE);

        INT SIZEWIDTH = 800;
        INT SIZEHEIGHT = 700;
        INT LOCATIONX = (SCREENSIZE.WIDTH - SIZEWIDTH) / 2;
        INT LOCATIONY = (SCREENSIZE.HEIGHT - SIZEHEIGHT) / 2;
        THIS.SETBOUNDS(LOCATIONX, LOCATIONY, SIZEWIDTH,
SIZEHEIGHT);
        JPANEL MAINPANEL = NEW JPANEL();
        THIS.SETCONTENTPANE(MAINPANEL);
        MAINPANEL.SETLAYOUT(NULL);
        MAINPANEL.SETSIZE(THIS.GETSIZE());
        PROGRESSBAR.SETBOUNDS(MAINPANEL.GETWIDTH()/5,
MAINPANEL.GETHEIGHT()/10, 200, 30);
        MAINPANEL.ADD(PROGRESSBAR);
        TILESPANEL.SETLAYOUT(NEW GRIDLAYOUT(4,4,0,0));

TILESPANEL.SETLOCATION(MAINPANEL.GETWIDTH()/16,MAINPANEL.GETHEIGHT
()/6 );
        TILESPANEL.SETSIZE(400, 400);
        MAINPANEL.ADD(TILESPANEL);
        JPANEL BUTTONSPANEL = NEW JPANEL();
        BUTTONSPANEL.SETLOCATION(TILESPANEL.GETWIDTH() + 100,
TILESPANEL.GETY() -10);
        BUTTONSPANEL.SETSIZE(200,300);
```

```
BUTTONSPANEL.SETLAYOUT(NEW GRIDLAYOUT(5,1,5,5));
BUTTONSPANEL.ADD(RANDOMBUTTON);
BUTTONSPANEL.ADD(BUILDBUTTON);
JPANEL NEXTPREVPANEL = NEW JPANEL();
NEXTPREVPANEL.SETLAYOUT(NEW GRIDLAYOUT(1,2,0,0));
NEXTPREVPANEL.ADD(PREVBUTTON);
NEXTPREVPANEL.ADD(NEXTBUTTON);
BUTTONSPANEL.ADD(NEXTPREVPANEL);
BUTTONSPANEL.ADD(CLOSEBUTTON);
BUTTONSPANEL.ADD(RETURNBUTTON);
MAINPANEL.ADD(BUTTONSPANEL);
BUILDBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
    @OVERRIDE
    PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
        DATABASE.AUTOSOLVE();
        IF (DATABASE.GETISSOLVED())
            NEXTBUTTON.SETENABLED(FALSE);
        ELSE
            NEXTBUTTON.SETENABLED(TRUE);
    }
});
NEXTBUTTON.SETACTIONCOMMAND("START");
NEXTBUTTON.ADDACTIONLISTENER(THIS);

CLOSEBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
    @OVERRIDE
    PUBLIC VOID MOUSEPRESSED(MOUSEEVENT E) {
        SYSTEM.EXIT(0);
    }
});
RETURNBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
    @OVERRIDE
    PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
        DATABASE.GETSECRETLEVEL().SETVISIBLE(FALSE);
        MAINWINDOW.GETUI().SETVISIBLE(TRUE);
```

```
            }
        });
        RANDOMBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                DATABASE.RANDOMIZESECRETPUZZLE();
                IF (DATABASE.GETISSOLVED())
                    NEXTBUTTON.SETENABLED(FALSE);
                ELSE
                    NEXTBUTTON.SETENABLED(TRUE);
            }
        });

        PREVBUTTON.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
            @OVERRIDE
            PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                DATABASE.SECRETPREVSTEP();
                IF (DATABASE.GETISSOLVED())
                    NEXTBUTTON.SETENABLED(FALSE);
                ELSE
                    NEXTBUTTON.SETENABLED(TRUE);
            }
        });
    }

    @OVERRIDE
    PUBLIC VOID ACTIONPERFORMED(ACTIONEVENT E) {
        SETCURSOR(CURSOR.GETPREDEFINEDCURSOR(CURSOR.WAIT_CURSOR));
        //INSTANCES OF JAVAX.SWING.SWINGWORKER ARE NOT REUSUABLE,
SO
        //WE CREATE NEW INSTANCES AS NEEDED.
        TASK = NEW TASK();

        TASK.ADDPROPERTYCHANGELISTENER(THIS);
        TASK.EXECUTE();
```

```
    }

    @OVERRIDE
    PUBLIC VOID PROPERTYCHANGE(PROPERTYCHANGEEVENT EVT) {
        IF ("PROGRESS" == EVT.GETPROPERTYNAME()) {
            INT PROGRESS = (INTEGER) EVT.GETNEWVALUE();
            PROGRESSBAR.SETVALUE(PROGRESS);
        }
    }
}
```

TILE.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.UI;

IMPORT COM.PRACTICE.BLUETEAM.DATABASE.DATABASE;

IMPORT JAVAX.SWING.*;
IMPORT JAVAX.XML.CRYPTO.DATA;
IMPORT JAVA.AWT.EVENT.MOUSEADAPTER;
IMPORT JAVA.AWT.EVENT.MOUSEEVENT;

PUBLIC CLASS TILE EXTENDS JBUTTON {
    // ЯВЛЯЕТСЯ ЛИ ПОСЛЕДНИМ ТАЙЛОМ
    PRIVATE BOOLEAN LAST;
    PUBLIC BOOLEAN ISLAST() {
        RETURN LAST;
    }
    PUBLIC VOID SETLAST(BOOLEAN LAST) {
        THIS.LAST = LAST;
    }

    PUBLIC INT GETNUMBEROFTILE() {
        RETURN NUMBEROFTILE;
```

```
    }

    // НОМЕР ТАЙЛА
    INT NUMBEROFTILE;
    INT LEVELNUMBER;
    // КОНСТРУКТОР
    PUBLIC TILE(INT NUMBEROFTILE, INT LEVELNUMBER) {
        THIS.LAST = FALSE;
        THIS.NUMBEROFTILE = NUMBEROFTILE;
        IF (NUMBEROFTILE == 15)
            THIS.LAST = TRUE;
        ELSE
            THIS.LAST = FALSE;

        SETSIZE(50,50);
        IF (THIS.LAST)
            SETVISIBLE(FALSE);
        ELSE
            SETVISIBLE(TRUE);
        IF (LEVELNUMBER == DATABASE.GETNUMBEROFLEVELS() + 1) {
            THIS.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
                @OVERRIDE
                PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                    DATABASE.MOVE(NUMBEROFTILE);
                    DATABASE.GETSECRETLEVEL().SETTILES();
                    IF (DATABASE.GETISSOLVED())

DATABASE.GETSECRETLEVEL().GETNEXTBUTTON().SETENABLED(FALSE);
                    ELSE

DATABASE.GETSECRETLEVEL().GETNEXTBUTTON().SETENABLED(TRUE);
                }
            });
        } ELSE {
            THIS.ADDMOUSELISTENER(NEW MOUSEADAPTER() {
```

```
                @OVERRIDE
                PUBLIC VOID MOUSECLICKED(MOUSEEVENT E) {
                    DATABASE.MOVE(NUMBEROFTILE, LEVELNUMBER);

DATABASE.GETLEVELWINDOWS(LEVELNUMBER).SETTILES();
                    IF (DATABASE.GETISSOLVED(LEVELNUMBER))

DATABASE.GETLEVELWINDOWS(LEVELNUMBER).GETNEXTBUTTON().SETENABLED(F
ALSE);
                    ELSE

DATABASE.GETLEVELWINDOWS(LEVELNUMBER).GETNEXTBUTTON().SETENABLED(T
RUE);
                }
            });
        }
    }
}
```

PUZZLETEST.JAVA

```
PACKAGE COM.PRACTICE.BLUETEAM.ALGO;

IMPORT ORG.JUNIT.BEFORE;
IMPORT ORG.JUNIT.TEST;

IMPORT STATIC ORG.JUNIT.ASSERT.*;

PUBLIC CLASS PUZZLETEST {
    PUBLIC PUZZLE TEMP;

    PUBLIC CLASS TILEPOS {
        PUBLIC INT X, Y;

        PUBLIC TILEPOS (INT X, INT Y) {
```

```
        THIS.X = X;
        THIS.Y = Y;
    }
}

PRIVATE INT[][] TILES;
PRIVATE PUZZLE.TILEPOS BLANK;
@BEFORE
PUBLIC VOID SETUP(){
    TEMP = NEW PUZZLE();
}

@TEST
PUBLIC VOID MOVE1() {
    PUZZLE BUF = NEW PUZZLE(TEMP);
    BUF.MOVE(BUF.WHEREIS(16));
    ASSERTEQUALS(BUF, TEMP);
}

@TEST
PUBLIC VOID MOVE2() {
    PUZZLE BUF = NEW PUZZLE(TEMP);
    BUF.MOVE(BUF.WHEREIS(-1));
    ASSERTEQUALS(BUF, TEMP);
}

@TEST
PUBLIC VOID MOVE3() {
    PUZZLE BUF = NEW PUZZLE(TEMP);
    BUF.MOVE(BUF.WHEREIS(11));
    ASSERTNOTEQUALS(BUF, TEMP);
}

@TEST
PUBLIC VOID MOVE4() {
```

```
    PUZZLE BUF = NEW PUZZLE(TEMP);
    BUF.MOVE(BUF.WHEREIS(14));
    ASSERTNOTEQUALS(BUF, TEMP);
}

@TEST
PUBLIC VOID SHUFFLE1() {
    PUZZLE BUF = NEW PUZZLE(TEMP);
    BUF.SHUFFLE(3);
    ASSERTNOTSAME(BUF, TEMP);
}

@TEST
PUBLIC VOID SHUFFLE2() {
    PUZZLE BUF = NEW PUZZLE(TEMP);
    BUF.SHUFFLE(25);
    ASSERTNOTSAME(BUF, TEMP);
}

@TEST
PUBLIC VOID ISVAILDMOVE1(){
    ASSERTTRUE(TEMP.ISVALIDMOVE(TEMP.WHEREIS(14)));
}

@TEST
PUBLIC VOID ISVAILDMOVE2(){
    ASSERTTRUE(TEMP.ISVALIDMOVE(TEMP.WHEREIS(11)));
}
@TEST
PUBLIC VOID ISNOTVALIDMOVE1(){
    ASSERTFALSE(TEMP.ISVALIDMOVE(TEMP.WHEREIS(-6)));

}
@TEST
PUBLIC VOID ISNOTVALIDMOVE2(){
```

```
        ASSERTFALSE(TEMP.ISVALIDMOVE(TEMP.WHEREIS(25)));
    }
    @TEST
    PUBLIC VOID ISNOTVALIDMOVE3(){
        ASSERTFALSE(TEMP.ISVALIDMOVE(TEMP.WHEREIS(4)));
    }
    @TEST
    PUBLIC VOID PUZ(){
        ASSERTEQUALS(PUZZLE.SOLVED, NEW PUZZLE());
    }
    @TEST
    PUBLIC VOID  ESTIMATEERROR1(){
        ASSERTEQUALS(0, TEMP.ESTIMATEERROR());
    }
    @TEST
    PUBLIC VOID  ESTIMATEERROR2(){
        TEMP.SHUFFLE(5);
        ASSERTNOTEQUALS(0, TEMP.ESTIMATEERROR());
    }
    @TEST
    PUBLIC VOID GETTILE(){
        ASSERTEQUALS(15, TEMP.GETTILE(TEMP.WHEREIS(15)));
    }
}
```

POM.XML

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<PROJECT XMLNS="HTTP://MAVEN.APACHE.ORG/POM/4.0.0"
         XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
         XSI:SCHEMALOCATION="HTTP://MAVEN.APACHE.ORG/POM/4.0.0
HTTP://MAVEN.APACHE.ORG/XSD/MAVEN-4.0.0.XSD">
    <MODELVERSION>4.0.0</MODELVERSION>

    <GROUPID>MAVENTEST</GROUPID>
```

```
<ARTIFACTID>MAVENTEST</ARTIFACTID>
<VERSION>1.0-SNAPSHOT</VERSION>
<PACKAGING>JAR</PACKAGING>

<DEPENDENCIES>
    <DEPENDENCY>
        <GROUPID>JUNIT</GROUPID>
        <ARTIFACTID>JUNIT</ARTIFACTID>
        <VERSION>4.12</VERSION>
        <SCOPE>TEST</SCOPE>
    </DEPENDENCY>

</DEPENDENCIES>

<BUILD>
    <PLUGINS>
        <PLUGIN>
            <GROUPID>ORG.APACHE.MAVEN.PLUGINS</GROUPID>
            <ARTIFACTID>MAVEN-COMPILER-PLUGIN</ARTIFACTID>
            <VERSION>3.1</VERSION>
            <CONFIGURATION>
                <SOURCE>1.8</SOURCE>
                <TARGET>1.8</TARGET>
            </CONFIGURATION>
        </PLUGIN>
        <PLUGIN>
            <!-- BUILD AN EXECUTABLE JAR -->
            <GROUPID>ORG.APACHE.MAVEN.PLUGINS</GROUPID>
            <ARTIFACTID>MAVEN-JAR-PLUGIN</ARTIFACTID>
            <VERSION>3.1.0</VERSION>
            <CONFIGURATION>
                <ARCHIVE>
                    <MANIFEST>
                        <ADDCLASSPATH>TRUE</ADDCLASSPATH>
```

```
<CLASSPATHPREFIX>LIB/</CLASSPATHPREFIX>
                            <MAINCLASS>MAIN</MAINCLASS>
                        </MANIFEST>
                    </ARCHIVE>
                </CONFIGURATION>
            </PLUGIN>
        </PLUGINS>
        <RESOURCES>
            <RESOURCE>

<DIRECTORY>SRC/MAIN/JAVA/COM/PRACTICE/BLUETEAM/RESOURCES</DIRECTOR
Y>
            </RESOURCE>
        </RESOURCES>
    </BUILD>

</PROJECT>
```